TRIÈST: Counting Local and Global Triangles in Fully-Dynamic Streams with Fixed Memory Size

Lorenzo De Stefani
Brown University
Providence, RI, USA
lorenzo@cs.brown.edu

Alessandro Epasto*
Google
New York, NY, USA
aepasto@google.com

Matteo Riondato*
Two Sigma Investments
New York, NY, USA
matteo@twosigma.com

Eli Upfal
Brown University
Providence, RI, USA
eli@cs.brown.edu

"Ogni lassada xe persa"[1] – Proverb from Trieste, Italy.

ABSTRACT

We present TRIÈST, a suite of one-pass streaming algorithms to compute unbiased, low-variance, high-quality approximations of the global and local (i.e., incident to each vertex) number of triangles in a fully-dynamic graph represented as an adversarial stream of edge insertions and deletions.

Our algorithms use reservoir sampling and its variants to exploit the user-specified memory space at all times. This is in contrast with previous approaches, which require hard-to-choose parameters (e.g., a fixed sampling probability) and offer no guarantees on the amount of memory they use. We analyze the variance of the estimations and show novel concentration bounds for these quantities.

Our experimental results on very large graphs demonstrate that TRIÈST outperforms state-of-the-art approaches in accuracy and exhibits a small update time.

1. INTRODUCTION

Exact computation of characteristic quantities of Web-scale networks is often impractical or even infeasible due to the humongous size of these graphs. It is natural in these cases to resort to *efficient-to-compute approximations* of these quantities, which, when of sufficiently high quality, can be used as proxies for the exact values.

In addition to being huge, many interesting networks are *fully-dynamic* and can be represented as a *stream* whose elements are edges/nodes insertions and deletions occurring in an *arbitrary* (even adversarial) order. Characteristic quantities in these graphs are *intrinsically volatile*, hence there is limited added value in maintaining them exactly. The goal is rather to keep track, *at all times*, of a high-quality approximation of these quantities. For efficiency, the algorithms should *aim at exploiting the available memory space as much as possible* and they should *require only one pass over the stream*.

We introduce TRIÈST, a suite of *sampling-based, one-pass algorithms for adversarial fully-dynamic streams to approximate the global number of triangles and the local number of triangles incident to each vertex*. Mining local and global triangles is a fundamental primitive with many applications (e.g., community detection [4], topic mining [10], spam/anomaly detection [3, 27], ego-networks mining [12] and protein interaction networks analysis [29].)

Many previous works on triangle estimation in streams also employ sampling (see Sect. 3), but they usually require the user to specify *in advance* an *edge sampling probability* p that is fixed for the entire stream. This approach presents several significant drawbacks. First, choosing a p that allows to obtain the desired approximation quality requires to know or guess a number of properties of the input (e.g., the size of the stream). Second, a fixed p implies that the sample size grows with the size of the stream, which is problematic when the stream size is not known in advance: if the user specifies a large p, the algorithm may run out of memory, while for a smaller p it will provide a suboptimal estimation. Third, even assuming to be able to compute a p that ensures (in expectation) full use of the available space, the memory would be fully utilized only at the end of the stream, and the estimations computed throughout the execution would be suboptimal.

Contributions. We address all the above issues by taking a significant departure from the fixed-probability, independent edge sampling approach taken even by state-of-the-art methods [27]. Specifically:

- We introduce TRIÈST (*TRI*angle *E*stimation from *ST*reams), a suite of *one-pass streaming algorithms* to approximate, at each time instant, the global and local number of triangles in a *fully-dynamic* graph stream (i.e., a sequence of edges additions and deletions in arbitrary order) using a *fixed amount of memory*. This is the first contribution that enjoys all these properties. TRIÈST only requires the user to specify *the amount of available memory*, an interpretable parameter that is definitively known to the user.
- Our algorithms maintain a sample of edges: they use the *reservoir sampling* [37] and *random pairing* [14] sampling

*Work partially done at Brown University.
[1]Any missed chance is lost forever.

KDD '16, August 13–17, 2016, San Francisco, CA, USA

ISBN 978-1-4503-4232-2/16/08…$15.00

DOI: http://dx.doi.org/10.1145/2939672.2939771

schemes to exploit the available memory as much as possible. To the best of our knowledge, ours is the first application of these techniques to subgraph counting in fully-dynamic, arbitrarily long, adversarially ordered streams. We present an analysis of the unbiasedness and of the variance of our estimators, and establish strong concentration results for them. The use of reservoir sampling and random pairing requires additional sophistication in the analysis, as the presence of an edge in the sample is *not independent* from the concurrent presence of another edge. Hence, in our proofs we must consider the complex dependencies in events involving sets of edges. The gain is worth the effort: we prove that the variance of our algorithms is smaller than that of state-of-the-art methods [27], and this is confirmed by our experiments.

- We conduct an extensive experimental evaluation of TRIÈST on very large graphs, some with billions of edges, comparing the performances of our algorithms to those of existing state-of-the-art contributions [18, 27, 32]. *Our algorithms significantly and consistently reduce the average estimation error by up to* 90% w.r.t. the state of the art, both in the global and local estimation problems, while using the same amount of memory. Our algorithms are also extremely scalable, showing update times in the order of hundreds of microseconds for graphs with billions of edges.

Due to space constraints, the proofs and additional experimental results can be found in the extended version [9].

2. PRELIMINARIES

We study the problem of counting global and local triangles in a fully-dynamic undirected graph as an arbitrary (adversarial) stream of edge insertions and deletions.

Formally, for any (discrete) time instant $t \geq 0$, let $G^{(t)} = (V^{(t)}, E^{(t)})$ be the graph observed up to and including time t. At time $t = 0$ we have $V^{(t)} = E^{(t)} = \emptyset$. For any $t > 0$, at time $t+1$ we receive an element $e_{t+1} = (\bullet, (u, v))$ from a stream, where $\bullet \in \{+, -\}$ and u, v are two distinct vertices. The graph $G^{(t+1)} = (V^{(t+1)}, E^{(t+1)})$ is obtained by *inserting a new edge or deleting an existing edge* as follows:

$$E^{(t+1)} = \begin{cases} E^{(t)} \cup \{(u,v)\} \text{ if } \bullet = \text{“}+\text{”} \\ E^{(t)} \setminus \{(u,v)\} \text{ if } \bullet = \text{“}-\text{”} \end{cases}.$$

If u or v do not belong to $V^{(t)}$, they are added to $V^{(t+1)}$. Nodes are deleted from $V^{(t)}$ when they have degree zero.

Edges can be added and deleted in the graph in an arbitrary adversarial order, i.e., as to cause the worst outcome for the algorithm, but we assume that the adversary has *no access to the random bits* used by the algorithm. We assume that *all operations have effect*: if $e \in E^{(t)}$ (resp. $e \notin E^{(t)}$), $(+, e)$ (resp. $(-, e)$) can not be on the stream at time $t+1$.

Given a graph $G^{(t)} = (V^{(t)}, E^{(t)})$, a *triangle* in $G^{(t)}$ is a *set* of three vertices $\{u, v, w\} \subseteq V^{(t)}$ such that $\{(u, v), (v, w), (w, u)\} \subseteq E^{(t)}$. We refer to the vertices forming a triangle as its *corners*. We denote with $\Delta^{(t)}$ the set of *all* triangles in $G^{(t)}$, and, for any vertex $u \in V^{(t)}$, with $\Delta_u^{(t)}$ the subset of $\Delta^{(t)}$ containing all and only the triangles that have u as a corner.

Problem definition. We study the *Global* (resp. *Local*) Triangle Counting Problem in Fully-dynamic Streams, which requires to compute, at each time $t \geq 0$ an estimation of $|\Delta^{(t)}|$ (resp. for each $u \in V$ an estimation of $|\Delta_u^{(t)}|$).

3. RELATED WORK

The literature on triangle counting is extremely rich, including exact algorithms, graph sparsifiers [35, 36], complex-valued sketches [20, 28], and MapReduce algorithms [30, 31, 33]. Here we restrict the discussion to the works most related to ours, i.e., to those presenting algorithms for counting or approximating the number of triangles from data streams. We refer to the survey by Latapy [25] for an in-depth discussion of other works.

Many previous contributions presented algorithms for more restricted (i.e., less generic) settings than ours, or for which the constraints on the computation are more lax [2, 6, 19, 22]. For example, Becchetti et al. [3] and Kolountzakis et al. [21] present algorithms for approximate triangle counting from *static* graphs by performing multiple passes over the input. Pavan et al. [32] and Jha et al. [18] propose algorithms for approximating only the global number of triangles from *edge-insertion-only* streams. Kutzkov and Pagh [23] present a one-pass algorithm for fully-dynamic graphs, but the triangle count estimation is (expensively) computed only at the end of the stream and the algorithm requires, in the worst case, more memory than what is needed to store the entire graph. Ahmed et al. [1] apply the sampling-and-hold approach to insertion-only graph stream mining to obtain, only at the end of the stream and using non-constant space, an estimation of many network measures including triangles.

None of these works has *all* the features offered by TRIÈST: performs a single pass over the data, handles fully-dynamic streams, uses a fixed amount of memory space, requires a single interpretable parameter, and returns an estimation at each time instant. Furthermore, our experimental results show that we outperform the algorithms from [18, 32] on insertion-only streams.

Lim and Kang [27] present an algorithm for insertion-only streams that is based on independent edge sampling with a fixed probability. Since the memory is not fully utilized during most of the stream, the variance of the estimate is large. Our approach handles fully-dynamic streams and makes better use of the available memory space at each time instant, resulting in a better estimation, as shown by our analytical and experimental results.

Vitter [37] presents a detailed analysis of the reservoir sampling scheme and discusses methods to speed up the algorithm by reducing the number of calls to the random number generator. Random Pairing [14] is an extension of reservoir sampling to handle fully-dynamic streams with insertions and deletions. Cohen et al. [8] generalize and extend the Random Pairing approach to the case where the elements on the stream are key-value pairs, where the value may be negative (and less than -1). In our settings, where the value is not less than -1 (for an edge deletion), these generalizations do not apply and the algorithm presented by Cohen et al. [8] reduces essentially to Random Pairing.

4. ALGORITHMS

We present TRIÈST, a suite of three novel algorithms for approximate global and local triangle counting from edge streams. The first two work on insertion-only streams, while the third can handle fully-dynamic streams where edge deletions are allowed.

Parameters. Our algorithms keep an edge sample $\mathcal{S}$ of up to M edges from the stream, where M is a positive integer

parameter. For ease of presentation, we realistically assume $M \geq 6$. In Sect. 1 we motivated the design choice of only requiring M as a parameter and remarked on its advantages over using a fixed sampling probability p. Our algorithms are designed to use the available space as much as possible.

Counters. TRIÈST algorithms keep *counters* to compute the estimations of the global and local number of triangles. They *always* keep one global counter τ for the estimation of the global number of triangles. Only the global counter is needed to estimate the total triangle count. To estimate the local triangle counts, the algorithms keep a set of local counters τ_u for a subset of the nodes $u \in V^{(t)}$. The local counters are created on the fly as needed, and *always* destroyed as soon as they have a value of 0. Hence our algorithms use $O(M)$ space (with one exception, see Sect. 4.2).

Notation. For any $t \geq 0$, let $G^{\mathcal{S}} = (V^{\mathcal{S}}, E^{\mathcal{S}})$ be the subgraph of $G^{(t)}$ containing all and only the edges in the current sample $\mathcal{S}$. We denote with $\mathcal{N}_u^{\mathcal{S}}$ the *neighborhood* of u in $G^{\mathcal{S}}$: $\mathcal{N}_u^{\mathcal{S}} = \{v \in V^{(t)} \ : \ (u,v) \in \mathcal{S}\}$ and with $\mathcal{N}_{u,v}^{\mathcal{S}} = \mathcal{N}_u^{\mathcal{S}} \cap \mathcal{N}_v^{\mathcal{S}}$ the *shared neighborhood* of u and v in $G^{\mathcal{S}}$.

4.1 A first algorithm – TRIÈST-BASE

We first present TRIÈST-BASE, which works on insertion-only streams and uses standard reservoir sampling [37] to maintain the edge sample $\mathcal{S}$:

- If $t \leq M$, then the edge $e_t = (u,v)$ on the stream at time t is deterministically inserted in $\mathcal{S}$.
- If $t > M$, TRIÈST-BASE flips a biased coin with heads probability M/t. If the outcome is heads, it chooses an edge $(w,z) \in \mathcal{S}$ uniformly at random, removes (w,z) from $\mathcal{S}$, and inserts (u,v) in $\mathcal{S}$. Otherwise, $\mathcal{S}$ is not modified.

After each insertion (resp. removal) of an edge (u,v) from $\mathcal{S}$, TRIÈST-BASE calls the procedure UPDATECOUNTERS that increments (resp. decrements) τ, τ_u and τ_v by $|\mathcal{N}_{u,v}^{\mathcal{S}}|$, and τ_c by one, for each $c \in \mathcal{N}_{u,v}^{\mathcal{S}}$.

The pseudocode for TRIÈST-BASE is presented in Alg. 1.

Algorithm 1 TRIÈST-BASE

```
Input: Insertion-only edge stream Σ, integer M ≥ 6
S ← ∅, t ← 0, τ ← 0
for each element (+, (u, v)) from Σ do
    t ← t + 1
    if SampleEdge((u, v), t) then
        S ← S ∪ {(u, v)}
        UpdateCounters(+, (u, v))

function SampleEdge((u, v), t)
    if t ≤ M then
        return True
    else if FlipBiasedCoin(M/t) = heads then
        (u', v') ← random edge from S
        S ← S \ {(u', v')}
        UpdateCounters(−, (u', v'))
        return True
    return False

function UpdateCounters((•, (u, v)))
    N^S_{u,v} ← N^S_u ∩ N^S_v
    for all c ∈ N^S_{u,v} do
        τ ← τ • 1
        τ_c ← τ_c • 1
        τ_u ← τ_u • 1
        τ_v ← τ_v • 1
```

Estimation. For any $t \geq 0$, let $\xi^{(t)} = \max\left\{1, \frac{t(t-1)(t-2)}{M(M-1)(M-2)}\right\}$. Let $\tau^{(t)}$ (resp. $\tau_u^{(t)}$) be the value of the counter τ at the end of time step t (i.e., after the edge on the stream at time t has been processed by TRIÈST-BASE) (resp. the value of the counter τ_u at the end of time step t if there is such a counter, 0 otherwise). When queried at the end of time t, TRIÈST-BASE returns $\xi^{(t)}\tau^{(t)}$ (resp. $\xi^{(t)}\tau_u^{(t)}$) as the estimation for the global (resp. local for $u \in V^{(t)}$) triangle count.

Analysis.

THEOREM 1. *We have*

$$\xi^{(t)}\tau^{(t)} = \tau^{(t)} = |\Delta^{(t)}| \text{ if } t \leq M$$
$$\mathbb{E}\left[\xi^{(t)}\tau^{(t)}\right] = |\Delta^{(t)}| \text{ if } t > M \ .$$

The TRIÈST-BASE estimations are not only *unbiased in all cases*, but actually *exact for* $t \leq M$, i.e., for $t \leq M$, they are the true global/local number of triangles in $G^{(t)}$.

We now analyze the variance of the estimation returned by TRIÈST-BASE for $t > M$ (the variance is 0 for $t \leq M$.) Let $t \geq 0$. For any $u \in V^{(t)}$, let $r_u^{(t)}$ be the number of *unordered* pairs of distinct triangles from $\Delta_u^{(t)}$ sharing an edge.[1] Similarly, let $r^{(t)} = \frac{1}{3}\sum_{u \in V^{(t)}} r_u^{(t)}$ be the *total* number of unordered pairs of distinct triangles from $\Delta^{(t)}$ sharing an edge. We also define $w^{(t)} = \binom{|\Delta^{(t)}|}{2} - r^{(t)}$ as the number of unordered pairs of distinct triangles that do not share any edge, and analogously for $w_u^{(t)}$.

THEOREM 2. *For any* $t > M$, *let* $f(t) = \xi^{(t)} - 1$,

$$g(t) = \xi^{(t)}\frac{(M-3)(M-4)}{(t-3)(t-4)} - 1$$

and

$$h(t) = \xi^{(t)}\frac{(M-3)(M-4)(M-5)}{(t-3)(t-4)(t-5)} - 1 \ (\leq 0).$$

We have:

$$\mathrm{Var}\left[\xi(t)\tau^{(t)}\right] = |\Delta^{(t)}|f(t) + r^{(t)}g(t) + w^{(t)}h(t). \quad (1)$$

In our proofs, we carefully account for the fact that, as we use reservoir sampling [37], the presence of an edge a in $\mathcal{S}$ is *not independent* from the concurrent presence of another edge b in $\mathcal{S}$. This is not the case for samples built using fixed-probability independent edge sampling. When computing the variance, we must consider not only pairs of triangles that share an edge (as for independent edge sampling approaches), but also pairs of triangles sharing no edge, as their respective presences in the sample are not independent events. The gain is worth the additional sophistication needed in the analysis, because the contribution to the variance by triangles not sharing edges is *non-positive* ($h(t) \leq 0$), i.e., it reduces the variance. A comparison of the variance of our estimator with that obtained with a fixed-probability independent edge sampling approach, is discussed below.

Let $h^{(t)}$ denote the maximum number of triangles sharing a single edge in $G^{(t)}$. The following concentration theorem relies on *1.* a result by Hajnal and Szemerédi [15] on graph coloring, *2.* a novel concentration result for fixed-probability independent edge sampling, and *3.* a Poisson-approximation-like result on probabilities of general events under reservoir sampling w.r.t. their probabilities under independent edge sampling. These ingredients are then combined to obtain the following result. The details can be found in our extended online version [9].

[1] Two distinct triangles can share at most one edge.

THEOREM 3. *Let $t \geq 0$ and assume $|\Delta^{(t)}| > 0$.*[2] *For any $\varepsilon, \delta \in (0,1)$, let*

$$\Phi = \sqrt[3]{8\varepsilon^{-2}\frac{3h^{(t)}+1}{|\Delta^{(t)}|}\ln\left(\frac{(3h^{(t)}+1)e}{\delta}\right)} \ .$$

If

$$M \geq \max\left\{t\Phi\left(1+\frac{1}{2}\ln^{2/3}(t\Phi)\right), 12\varepsilon^{-1}+e^2, 25\right\},$$

then $|\xi^{(t)}\tau^{(t)} - |\Delta^{(t)}|| < \varepsilon|\Delta^{(t)}|$ with probability $> 1-\delta$.

Results analogous to those in Thms. 1, 2, and 3 hold for the local triangle count for any $u \in V^{(t)}$, replacing the global quantities with the corresponding local ones.

Comparison with fixed-probability approaches. We now compare the variance of TRIÈST-BASE to the variance of the fixed probability sampling approach MASCOT-C [27], which samples edges *independently* with a fixed probability p and uses $p^{-3}|\Delta_{\mathcal{S}}|$ as the estimate for the global number of triangles at time t. As shown by Lim and Kang [27, Lemma 2], the variance of this estimator is

$$\mathrm{Var}[p^{-3}|\Delta_{\mathcal{S}}|] = |\Delta^{(t)}|\bar{f}(p) + r^{(t)}\bar{g}(p),$$

where $\bar{f}(p) = p^{-3} - 1$ and $\bar{g}(p) = p^{-1} - 1$.

Assume that we give MASCOT-C the additional information that the stream has finite length T, and assume we run MASCOT-C with $p = M/T$ so that the expected sample size at the end of the stream is M.[3] Let $\mathbb{V}^{(t)}_{\mathrm{fix}}$ be the resulting variance of the MASCOT-C estimator at time t, and let $\mathbb{V}^{(t)}$ be the variance of our estimator at time t (see (1)). For $t \leq M$, $\mathbb{V}^{(t)} = 0$, hence $\mathbb{V}^{(t)} \leq \mathbb{V}^{(t)}_{\mathrm{fix}}$.

For $M < t < T$, we can show the following result.

LEMMA 1. *Let $0 < \alpha < 1$ be a constant. For any constant $M > \max(\frac{8\alpha}{1-\alpha}, 42)$ and any $t \leq \alpha T$ we have $\mathbb{V}^{(t)} < \mathbb{V}^{(t)}_{\mathrm{fix}}$.*

For example, if we set $\alpha = 0.99$ and run TRIÈST-BASE with $M \geq 400$ and MASCOT-C with $p = M/T$, we have that TRIÈST-BASE has strictly smaller variance than MASCOT-C for 99% of the stream.

What about $t = T$? The difference between the definitions of $\mathbb{V}^{(t)}_{\mathrm{fix}}$ and $\mathbb{V}^{(t)}$ is in the presence of $\bar{f}(M/T)$ instead of $f(t)$ (resp. $\bar{g}(M/T)$ instead of $g(t)$) as well as the additional term $w^{(t)}h(M,t) \leq 0$ in our $\mathbb{V}^{(t)}$. Let $M(T)$ be an arbitrary slowly increasing function of T. For $T \to \infty$ we can show that $\lim_{T\to\infty}\frac{\bar{f}(M(T)/T)}{f(T)} = \lim_{T\to\infty}\frac{\bar{g}(M(T)/T)}{g(T)} = 1$, hence, *informally*, $\mathbb{V}^{(T)} \to \mathbb{V}^{(T)}_{\mathrm{fix}}$, for $T \to \infty$.

A similar discussion also holds for the method we present in Sect. 4.2, and explains the results of our experimental evaluations, which show that our algorithms have strictly lower (empirical) variance than fixed probability approaches for most of the stream.

Update time. The time to process an element of the stream is dominated by the computation of the shared neighborhood $\mathcal{N}_{u,v}$ in UPDATECOUNTERS. A MERGESORT-based algorithm for the intersection requires $O(\deg(u) + \deg(v))$ time, where the degrees are w.r.t. the graph $G_{\mathcal{S}}$. By storing the neighborhood of each vertex in a Hash Table (resp. an AVL tree), the update time can be reduced to $O(\min\{\deg(v), \deg(u)\})$ (resp. amortized time $O(\min\{\deg(v), \deg(u)\} + \log\max\{\deg(v), \deg(u)\})$).

[2] If $|\Delta^{(t)}| = 0$, our algorithms correctly estimate 0 triangles.

[3] We are giving MASCOT-C a significant advantage: if only space M were available, we should run MASCOT-C with a sufficiently smaller $p' < p$, otherwise there would be a constant probability that MASCOT-C would run out of memory.

4.2 Improved insertion algorithm – TRIÈST-IMPR

TRIÈST-IMPR is a variant of TRIÈST-BASE with small modifications that result in higher-quality (i.e., lower variance) estimations. The changes are:

1. UPDATECOUNTERS is called *unconditionally for each element on the stream*, before the algorithm decides whether or not to insert the edge into $\mathcal{S}$. W.r.t. the pseudocode in Alg. 1, this change corresponds to moving the call to UPDATECOUNTERS on line 6 to *before* the **if** block. MASCOT [27] uses a similar idea, but TRIÈST-IMPR is significantly different as TRIÈST-IMPR allows edges to be removed from the sample, since it uses reservoir sampling.
2. TRIÈST-IMPR *never* decrements the counters when an edge is removed from $\mathcal{S}$. W.r.t. the pseudocode in Alg. 1, we remove the call to UPDATECOUNTERS on line 13.
3. UPDATECOUNTERS performs a *weighted* increase of the counters using $\eta^{(t)} = \max\{1, (t-1)(t-2)/(M(M-1))\}$ as weight. W.r.t. the pseudocode in Alg. 1, we replace "1" with $\eta^{(t)}$ on lines 19–22 (given change 2 above, all the calls to UPDATECOUNTERS have $\bullet = +$).

Counters. If we are interested only in estimating the global number of triangles in $G^{(t)}$, TRIÈST-IMPR needs to maintain only the counter τ and the edge sample $\mathcal{S}$ of size M, so it still requires space $O(M)$. If instead we are interested in estimating the local triangle counts, at any time t TRIÈST maintains (non-zero) local counters *only* for the nodes u such that at least one triangle with a corner u has been detected by the algorithm up until time t. The number of such nodes may be greater than $O(M)$, but this is the price to pay to obtain estimations with lower variance (Thm. 5).

Estimation. When queried for an estimation, TRIÈST-IMPR returns the value of the corresponding counter, unmodified.

Analysis.

THEOREM 4. *We have $\tau^{(t)} = |\Delta^{(t)}|$ if $t \leq M$ and $\mathbb{E}\left[\tau^{(t)}\right] = |\Delta^{(t)}|$ if $t > M$.*

As in TRIÈST-BASE, the estimations by TRIÈST-IMPR are *exact* at time $t \leq M$ and unbiased for $t > M$.

We now show an *upper bound* to the variance of the TRIÈST-IMPR estimations for $t > M$. The proof relies on a very careful analysis of the covariance of two triangles which depends on the order of arrival of the edges in the stream (which we assume to be adversarial). Let $z^{(t)}$ be the number of unordered pairs (λ, γ) of distinct triangles in $G^{(t)}$ that share an edge g and are such that g is neither the last edge of λ on the stream nor the last edge of γ on the stream. For any node $u \in V^{(t)}$, let $z^{(t)}_u$ be similarly defined, but considering only the triangles incident to u.

THEOREM 5. *Then, for any time $t > M$, we have*

$$\mathrm{Var}\left[\tau^{(t)}\right] \leq |\Delta^{(t)}|(\eta^{(t)} - 1) + z^{(t)}\frac{t-1-M}{M} \ .$$

For the sake of clarity, in Thm. 5, we chose not to present a stricter but more complex bound involving triangles that do not share any edge, which, as in Thm. 2, would add a *non-positive* term to the variance (i.e., reduce the variance).

The following result relies on Chebyshev's inequality and Thm. 5.

THEOREM 6. *Let $t \geq 0$ and assume $|\Delta^{(t)}| > 0$. For any $\varepsilon, \delta \in (0,1)$, if*

$$M > \max\left\{\sqrt{\frac{2(t-1)(t-2)}{\delta\varepsilon^2|\Delta^{(t)}|+2} + \frac{1}{4}} + \frac{1}{2}, \frac{2z^{(t)}(t-1)}{\delta\varepsilon^2|\Delta^{(t)}|^2+2z^{(t)}}\right\}$$

then $|\tau^{(t)} - |\Delta^{(t)}|| < \varepsilon|\Delta^{(t)}|$ with probability $> 1-\delta$.

In Thms. 5 and 6, it is possible to replace the value $z^{(t)}$ with the more interpretable $r^{(t)}$, which is agnostic to the order of the edges on the stream but gives a looser upper bound to the variance.

Results analogous to those in Thms. 4, 5, and 6 hold for the local triangle count for any $u \in V^{(t)}$, replacing the global quantities with the corresponding local ones.

4.3 Fully-dynamic algorithm – TRIÈST-FD

TRIÈST-FD computes unbiased estimates of the global and local triangle counts in a *fully-dynamic stream where edges are inserted/deleted in any arbitrary, adversarial order*. It is based on *random pairing* (RP) [14], a sampling scheme that extends reservoir sampling and can handle deletions. The idea behind the RP scheme is that edge deletions seen on the stream will be "compensated" by future edge insertions. Following RP, TRIÈST-FD keeps a counter d_i (resp. d_o) to keep track of the number of uncompensated edge deletions involving an edge e that was (resp. was *not*) in $\mathcal{S}$ at the time the deletion for e was on the stream.

When an edge deletion for an edge $e \in E^{(t-1)}$ is on the stream at the beginning of time step t, then, if $e \in \mathcal{S}$ at this time, TRIÈST-FD removes e from $\mathcal{S}$ (effectively decreasing the number of edges stored in the sample by one) and increases d_i by one. Otherwise, it just increases d_o by one. When an edge insertion for an edge $e \notin E^{(t-1)}$ is on the stream at the beginning of time step t, if $d_\text{i} + d_\text{o} = 0$, then TRIÈST-FD follows the standard reservoir sampling scheme. If $|\mathcal{S}| < M$, then e is deterministically inserted in $\mathcal{S}$ without removing any edge from $\mathcal{S}$ already in $\mathcal{S}$, otherwise it is inserted in $\mathcal{S}$ with probability M/t, replacing an uniformly-chosen edge already in $\mathcal{S}$. If instead $d_\text{i} + d_\text{o} > 0$, then e is inserted in $\mathcal{S}$ with probability $d_\text{i}/(d_\text{i}+d_\text{o})$; since it must be $d_\text{i} > 0$, then it must be $|\mathcal{S}| < M$ and no edge already in $\mathcal{S}$ needs to be removed. In any case, after having handled the eventual insertion of e into $\mathcal{S}$, the algorithm decreases d_i by 1 if e was inserted in $\mathcal{S}$, otherwise it decreases d_o by 1. TRIÈST-FD also keeps track of $s^{(t)} = |E^{(t)}|$ by appropriately incrementing or decrementing a counter by 1 depending on whether the element on the stream is an edge insertion or deletion. The pseudocode for TRIÈST-FD is presented in Alg. 2 where the UPDATECOUNTERS procedure is the one from Alg. 1.

Estimation. We denote as $M^{(t)}$ the size of $\mathcal{S}$ at the end of time t (we always have $M^{(t)} \leq M$). For any time t, let $d_\text{i}^{(t)}$ and $d_\text{o}^{(t)}$ be the value of the counters d_i and d_o at the end of time t respectively, and let $\omega^{(t)} = \min\{M, s^{(t)} + d_\text{i}^{(t)} + d_\text{o}^{(t)}\}$. Define

$$\kappa^{(t)} = 1 - \sum_{j=0}^{2} \binom{s^{(t)}}{j}\binom{d_\text{i}^{(t)}+d_\text{o}^{(t)}}{\omega^{(t)}-j} \Big/ \binom{s^{(t)}+d_\text{i}^{(t)}+d_\text{o}^{(t)}}{\omega^{(t)}} \ .$$

Algorithm 2 TRIÈST-FD

Input: Fully Dynamic edge stream Σ, integer $M \geq 6$
$\mathcal{S} \leftarrow \emptyset$, $d_\text{i} \leftarrow 0$, $d_\text{o} \leftarrow 0$, $t \leftarrow 0$, $s \leftarrow 0$
for each element $(\bullet, (u,v))$ from Σ **do**
  $t \leftarrow t+1$
  $s \leftarrow s \bullet 1$
  if $\bullet = +$ **then**
    if SAMPLEEDGE(u,v) **then**
      UPDATECOUNTERS$(+, (u,v))$
  else if $(u,v) \in \mathcal{S}$ **then**
    UPDATECOUNTERS$(-, (u,v))$
    $\mathcal{S} \leftarrow \mathcal{S} \setminus \{(u,v)\}$
    $d_\text{i} \leftarrow d_\text{i} + 1$
  else $d_\text{o} \leftarrow d_\text{o} + 1$

function SAMPLEEDGE(u,v)
  if $d_\text{o} + d_\text{i} = 0$ **then**
    if $|\mathcal{S}| < M$ **then**
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{(u,v)\}$
      return True
    else if FLIPBIASEDCOIN$(\frac{M}{t})$ = heads **then**
      Select (z,w) uniformly at random from $\mathcal{S}$
      UPDATECOUNTERS$(-, (z,w))$
      $\mathcal{S} \leftarrow (\mathcal{S} \setminus \{(z,w)\}) \cup \{(u,v)\}$
      return True
  else if FLIPBIASEDCOIN$\left(\frac{d_\text{i}}{d_\text{i}+d_\text{o}}\right)$ = heads **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(u,v)\}$
    $d_\text{i} \leftarrow d_\text{i} - 1$
    return True
  else
    $d_\text{o} \leftarrow d_\text{o} - 1$
    return False

For any three positive integers a, b, c s.t. $a \leq b \leq c$, define

$$\psi_{a,b,c} = \prod_{i=0}^{a-1} \frac{c-i}{b-i} \ .$$

When queried at the end of time t, for an estimation of the global number of triangles, TRIÈST-FD returns

$$\rho^{(t)} = \begin{cases} 0 \text{ if } M^{(t)} < 3 \\ \frac{\tau^{(t)}}{\kappa^{(t)}}\psi_{3,M^{(t)},s^{(t)}} = \frac{\tau^{(t)}}{\kappa^{(t)}}\frac{s^{(t)}(s^{(t)}-1)(s^{(t)}-2)}{M^{(t)}(M^{(t)}-1)(M^{(t)}-2)} \text{ othw.} \end{cases}$$

When estimating $|\Delta_u^{(t)}|$ for $u \in V^{(t)}$, the definition for $\rho_u^{(t)}$ uses $\tau_u^{(t)}$ and has the additional condition that $\rho_u^{(t)} = 0$ if there is no counter τ_u. TRIÈST-FD can keep track of $\kappa^{(t)}$ during the execution, each update of $\kappa^{(t)}$ taking time $O(1)$. Hence the time to return the estimations is still $O(1)$.

Analysis. Let t^* be the first $t \geq M+1$ such that $|E^{(t)}| = M+1$, if such a time step exists (otherwise $t^* = +\infty$).

THEOREM 7. *We have $\rho^{(t)} = |\Delta^{(t)}|$ for all $t < t^*$, and $\mathbb{E}\left[\rho^{(t)}\right] = |\Delta^{(t)}|$ for $t \geq t^*$.*

The proof relies on properties of RP and on the definitions of $\kappa^{(t)}$ and $\rho^{(t)}$.

THEOREM 8. *Let $t > t^*$ s.t. $|\Delta^{(t)}| > 0$ and $s^{(t)} \geq M$. Suppose we have $d^{(t)} = d_o^{(t)} + d_i^{(t)} \leq \alpha s^{(t)}$ total unpaired deletions at time t, with $0 \leq \alpha < 1$. If $M \geq \frac{1}{2\sqrt{\alpha'-\alpha}}7\ln s^{(t)}$ for some $\alpha < \alpha' < 1$, we have:*

$$\text{Var}\left[\rho^{(t)}\right] \leq (\kappa^{(t)})^{-2}|\Delta^{(t)}|\left(\psi_{3,M(1-\alpha'),s^{(t)}} - 1\right) + (\kappa^{(t)})^{-2}2 + (\kappa^{(t)})^{-2}r^{(t)}\left(\psi^2_{3,M(1-\alpha'),s^{(t)}}\psi^{-1}_{5,M(1-\alpha'),s^{(t)}} - 1\right)$$

The following result relies on Chebyshev's inequality and on Thm. 8.

THEOREM 9. *Let $t \geq t^*$ s.t. $|\Delta^{(t)}| > 0$ and $s^{(t)} \geq M$. Let $d^{(t)} = d_o^{(t)} + d_i^{(t)} \leq \alpha s^{(t)}$ for some $0 \leq \alpha < 1$. For any $\varepsilon, \delta \in (0,1)$, if for some $\alpha < \alpha' < 1$*

$$M > \max \left\{ \frac{1}{\sqrt{a' - \alpha}} 7 \ln s^{(t)}, \right.$$
$$(1-\alpha')^{-1} \left(\sqrt[3]{\frac{2s^{(t)}(s^{(t)}-1)(s^{(t)}-2)}{\delta \varepsilon^2 |\Delta^{(t)}| (\kappa^{(t)})^2 + 2\frac{|\Delta^{(t)}|-2}{|\Delta^{(t)}|}}} + 2 \right),$$
$$\left. \frac{(1-\alpha')^{-1}}{3} \left(\frac{r^{(t)} s^{(t)}}{\delta \varepsilon^2 |\Delta^{(t)}|^2 (\kappa^{(t)})^{-2} + 2r^{(t)}} \right) \right\}$$

then $|\rho^{(t)} - |\Delta^{(t)}|| < \varepsilon |\Delta^{(t)}|$ with probability $> 1 - \delta$.

Results analogous to those in Thms. 7, 8, and 9 hold for the local triangle count for any $u \in V^{(t)}$, replacing the global quantities with the corresponding local ones.

5. EXPERIMENTAL EVALUATION

We evaluated TRIÈST on several real-world graphs with up to a billion edges. The algorithms were implemented in C++, and ran on the Brown University CS department cluster.[4] Each run employed a single core and used at most 4 GB of RAM. We report here only a subset of the results. Additional details are available in the extended online version. The code is available from http://bigdata.cs.brown.edu/triangles.html.

Datasets. We created the streams from the following publicly available graphs (properties in Table 1).

Patent (Co-Aut.) and Patent (Cit.). The *Patent (Co-Aut.)* and *Patent (Cit.)* graphs are obtained from a dataset of ≈ 2 million U.S. patents granted between '75 and '99 [16]. In *Patent (Co-Aut.)*, the nodes represent inventors and there is an edge with timestamp t between two co-inventors of a patent if the patent was granted in year t. In *Patent (Cit.)*, nodes are patents and there is an edge (a, b) with timestamp t if patent a cites b and a was granted in year t.

LastFm. The LastFm graph is based on a dataset [7, 34] of ≈ 20 million last.fm song listenings, ≈ 1 million songs and ≈ 1000 users. There is a node for each song and an edge between two songs if ≥ 3 users listened to both on day t.

Yahoo! Answers. The Yahoo! Answers graph is obtained from a sample of ≈ 160 million answers to ≈ 25 millions questions posted on Yahoo! Answers [38]. An edge connects two users at time $max(t_1, t_2)$ if they both answered the same question at times t_1, t_2 respectively. We removed 6 outliers questions with more than 5000 answers.

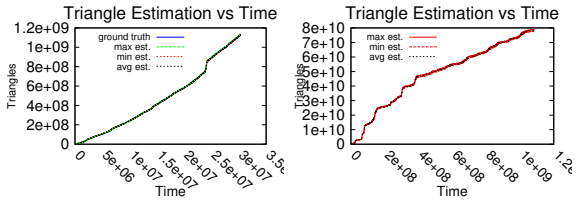
Twitter. This is a snapshot [5, 24] of the Twitter followers/following network with ≈ 41 million nodes and ≈ 1.5 billions edges. We do not have time information for the edges, hence we assign a random timestamp to the edges (of which we ignore the direction).

Ground truth. To evaluate the accuracy of our algorithms, we computed the *ground truth* for our smaller graphs (i.e., the exact number of global and local triangles for each time step), using an exact algorithm. The entire current graph is stored in memory and when an edge u, v is inserted (or deleted) we update the current count of local and global triangles by checking how many triangles are completed (or broken). As exact algorithms are not scalable, computing the exact triangle count is feasible only for small graphs such as Patent (Co-Aut.), Patent (Cit.) and LastFm. Table 1 reports the exact total number of triangles at the end of the stream for those graphs (and an estimate for the larger ones using TRIÈST-IMPR with $M = 1000000$).

[4] https://cs.brown.edu/about/system/services/hpc/grid/

Graph	$\|V\|$	$\|E\|$	$\|E_u\|$	$\|\Delta\|$
Patent (Co-Aut.)	1,162,227	3,660,945	2,724,036	3.53×10^6
Patent (Cit.)	2,745,762	13,965,410	13,965,132	6.91×10^6
LastFm	681,387	43,518,693	30,311,117	1.13×10^9
Yahoo! Answers	2,432,573	1.21×10^9	1.08×10^9	7.86×10^{10}
Twitter	41,652,230	1.47×10^9	1.20×10^9	3.46×10^{10}

Table 1: Properties of the dynamic graph streams analyzed. $|V|$, $|E|$, $|E_u|$, $|\Delta|$ refer respectively to the number of nodes appearing in the graph, the number of edge addition events, the number of distinct edges additions, and the maximum number of triangles in the graph (for Yahoo! Answers and Twitter estimated with TRIÈST-IMPR $M = 1000000$, otherwise computed exactly with the naïve algorithm).



(a) LastFm (b) Yahoo! Answers

Figure 1: Estimation by TRIÈST-IMPR of the global number of triangles over time. Our estimations have very small error and variance: the ground truth is indistinguishable from max/min point-wise estimation over ten runs.

5.1 Insertion-only case

We now evaluate TRIÈST on insertion-only streams and compare its performances with those of state-of-the-art approaches [18, 27, 32], showing that TRIÈST has an average estimation error significantly smaller than these methods both for the global and local estimation problems, while using the same amount of memory.

Estimation of the global number of triangles. Starting from an empty graph we add one edge at a time, in timestamp order. Figure 1 illustrates the evolution, over time, of the estimation computed by TRIÈST-IMPR with $M =$ 1,000,000. For smaller graphs for which the ground truth can be computed exactly, the curve of the exact count is practically indistinguishable from our estimation showing the precision of the method. Our estimators have very small variance even on the very large Yahoo! Answers graph (point-wise max/min estimation over ten runs is almost coincident with the average estimation). These results show that TRIÈST-IMPR is very accurate even when storing less than a 0.001 fraction of the total edges of the graph.

Comparison with the state of the art. We compare quantitatively with three state-of-the-art methods: MAS-

Graph	Impr.	p	Max. MAPE MASCOT	Max. MAPE TRIÈST	Avg. MAPE MASCOT	Avg. MAPE TRIÈST	Change
Patent (Cit.)	N	0.01	0.9231	0.2583	0.6517	0.1811	-72.2%
	Y	0.01	0.1907	0.0363	0.1149	0.0213	-81.4%
	N	0.1	0.0839	0.0124	0.0605	0.0070	-88.5%
	Y	0.1	0.0317	0.0037	0.0245	0.0022	-91.1%
LastFm	N	0.01	0.1525	0.0185	0.0627	0.0118	-81.2%
	Y	0.01	0.0273	0.0046	0.0141	0.0034	-76.2%
	N	0.1	0.0075	0.0028	0.0047	0.0015	-68.1%
	Y	0.1	0.0048	0.0013	0.0031	0.0009	-72.1%

Table 2: Global triangle estimation MAPE for TRIÈST and MASCOT. The rightmost column shows the reduction in terms of the avg. MAPE obtained by using TRIÈST. Rows with Y in column "Impr." refer to improved algorithms (TRIÈST-IMPR and MASCOT-I) while those with N to basic algorithms (TRIÈST-BASE and MASCOT-C).

p	Avg. Pearson MASCOT-I	Avg. Pearson TRIÈST	Avg. Pearson Change	Avg. ε Err MASCOT-I	Avg. ε Err TRIÈST	Avg. ε Err Change
0.1	0.99	1.00	+1.18%	0.79	0.30	-62.02%
0.05	0.97	1.00	+2.48%	0.99	0.47	-52.79%
0.01	0.85	0.98	+14.28%	1.35	0.89	-34.24%

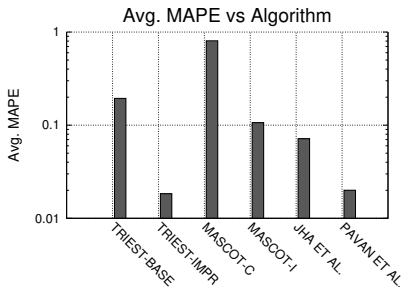
Table 3: Comparison of the quality of the local triangle estimations in LastFM between TRIÈST-IMPR and MASCOT-I). We outperform MASCOT-I using the same amount of memory.

COT [27], JHA ET AL. [18] and PAVAN ET AL. [32]. MASCOT is a suite of local triangle counting methods (but provides also a global estimation). The other two are global triangle counting approaches. None of these can handle fully-dynamic streams, in contrast with TRIÈST-FD. We first compare the three methods to TRIÈST for the global triangle counting estimation. MASCOT comes in two memory efficient variants: the basic MASCOT-C variant and an improved MASCOT-I variant.[5] Both variants sample edges with fixed probability p, so there is no guarantee on the amount of memory used during the execution. To ensure fairness of comparison, we devised the following experiment. First, we run both MASCOT-C and MASCOT-I for $\ell = 10$ times with a fixed p using the same random bits for the two algorithms run-by-run (i.e. the same coin tosses used to select the edges) measuring each time the number of edges M_i' stored in the sample at the end of the stream (by construction this the is same for the two variants run-by-run). Then, we run our algorithms using $M = M_i'$ (for $i \in [\ell]$). We do the same to fix the size of the edge memory for JHA ET AL. [18] and PAVAN ET AL. [32].[6] This way, all algorithms use the same amount of memory for storing edges (run-by-run).

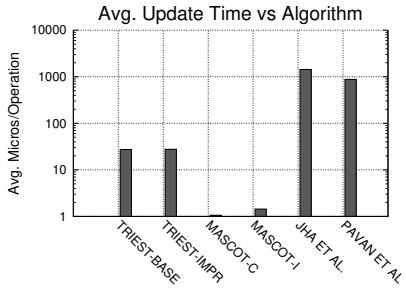
We use the *MAPE* (Mean Average Percentage Error) to assess the accuracy of the global triangle estimators over time. The MAPE measures the average percentage of the prediction error with respect to the ground truth, and is widely used in the prediction literature [17]. For $t = 1, \ldots, T$, let $\overline{\Delta}^{(t)}$ be the estimator of the number of triangles at time t, the MAPE is defined as $\frac{1}{T}\sum_{t=1}^{T}\left|\frac{|\Delta^{(t)}|-\overline{\Delta}^{(t)}}{|\Delta^{(t)}|}\right|$.[7]

[5]In the original work [27], this variant had no suffix and was simply called MASCOT. We add the -I suffix to avoid confusion. The variant MASCOT-A can be forced to store the entire graph with probability 1 (using an adversarial edge order) so we do not consider it here.

[6]More precisely, we use $M_i'/2$ estimators in PAVAN ET AL. as each estimator stores two edges. For JHA ET AL. we set the two reservoirs in the algorithm to have each size $M_i'/2$. This way, all algorithms use M_i' cells for storing (w)edges.



(a) MAPE



(b) Update Time

Figure 2: Average MAPE and average update time of the various methods on the Patent (Co-Aut.) graph with $p = 0.01$ – insertion only. TRIÈST-IMPR has the lowest error. Both PAVAN ET AL. and JHA ET AL. have very high update times compared to our method and the two MASCOT variants.

In Fig. 2(a), we compare the average MAPE of TRIÈST-BASE and TRIÈST-IMPR as well as the two MASCOT variants and the other two streaming algorithms for the Patent (Co-Aut.) graph, fixing $p = 0.01$. TRIÈST-IMPR has the smallest error of all the algorithms compared.

We now turn our attention to the efficiency of the methods. Figure 2(b) shows the average update time per operation in Patent (Co-Aut.) graph, fixing $p = 0.01$. Both JHA ET AL. [18] and PAVAN ET AL. [32] are up to ≈ 3 orders of magnitude slower than the MASCOT variants and TRIÈST. This is expected as both algorithms have an update complexity of $\Omega(M)$ (they have to go through the entire reservoir graph at each step), while both MASCOT algorithms and TRIÈST need only to access the neighborhood of the nodes involved in the edge addition.[8] This allows both algorithms to efficiently exploit larger memory sizes. We can use efficiently M up to 1 million edges in our experiments,

[7]The MAPE is not defined for t s.t. $\Delta^{(t)} = 0$ so we compute it only for t s.t. $|\Delta^{(t)}| > 0$. All algorithms we consider are guaranteed to output the correct answer for t s.t. $|\Delta^{(t)}| = 0$.

[8]We observe that PAVAN ET AL. [32] would be more efficient with batch updates. However, we want to estimate the triangles continuously at each update. In their experiments they use batch sizes of million of updates for efficiency.

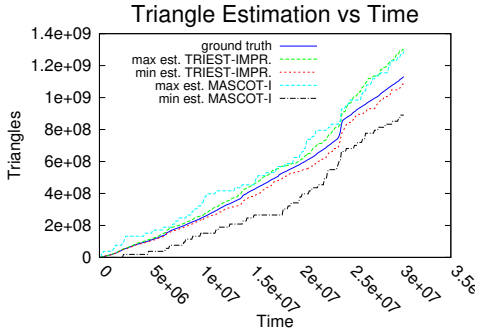


Figure 3: Variance of TRIÈST-IMPR with $M = 10000$ and of MASCOT with same expected memory, on LastFM. TRIÈST-IMPR has a smaller variance: the max/min estimation lines are closer to the ground truth. (Average estimations are qualitatively similar and not shown).

which only requires few megabytes of RAM.[9] MASCOT is one order of magnitude faster than TRIÈST (which runs in $\approx$ 28 micros/op), because it does not have to handle edge removal from the sample, as it offers no guarantees on the used memory. As we will show, TRIÈST has much higher precision and scales well on billion-edges graphs.

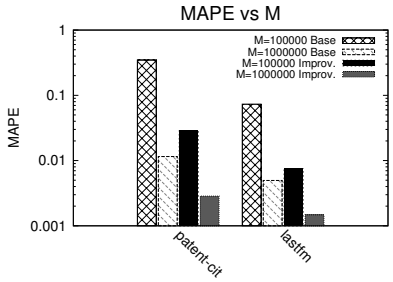
Given the slow execution of the other algorithms on the larger datasets we compare in details TRIÈST only with MASCOT.[10] Table 2 shows the average MAPE of the two approaches. The results confirm the pattern observed in Figure 2(a): TRIÈST-BASE and TRIÈST-IMPR both have an average error significantly smaller than that of the basic MASCOT-C and improved MASCOT variant respectively. We achieve up to a 91% (i.e., 9-fold) reduction in the MAPE while using the same amount of memory. This experiment confirms the theory: reservoir sampling has overall lower or equal variance in all steps for the same expected total number of sampled edges. To further validate this observation we run TRIÈST-IMPR and of the improved MASCOT variant using the same (expected memory) $M = 10000$. Figure 3 shows the max-min estimation over 10 runs. TRIÈST-IMPR shows significantly lower variance over the evolution: the maxmin estimation lines are closer to the ground truth virtually all time. This confirms our theoretical observations in the previous sections. Even with very low M (about 2/10000 of the size of the graph) TRIÈST gives a good estimation.

Local triangle counting. We compare the precision in local triangle count estimation of TRIÈST with that of MASCOT [27] using the same approach of the previous experiment. We can not compare with JHA ET AL. and PAVAN ET AL. algorithms as they provide only global estimation. As in [27], we measure the Pearson coefficient and the average ε error (see [27] for definitions). In Table 3 we report the Pearson coefficient and average ε error over all timestamps for the smaller graphs.[11] TRIÈST (significantly) improves

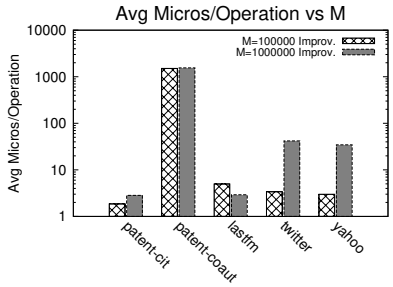
[9]The experiments in [18] use M in the order of 10^3, and in [32], large M values require large batches for efficiency.

[10]We attempted to run the other two algorithms but they did not complete after 12 hours for the larger datasets in Table 2 with the prescribed p parameter setting.

[11]For efficiency, in this test we evaluate the local number of triangles of all nodes every 1000 edge updates.



(a) M vs MAPE



(b) M vs Update Time

Figure 4: Trade-offs between M and MAPE or avg. update time (μs) – edge insertion only. Higher M implies lower errors but higher update times.

(i.e., has higher correlation and lower error) over the state-of-the-art MASCOT, using the same amount of memory.

Memory vs accuracy trade-offs. We study the tradeoff between the sample size M vs the running time and accuracy of the estimators. Figure 4(a) shows the tradeoffs between the accuracy of the estimation and the size M for the smaller graphs for which the ground truth number of triangles can be computed exactly using the naive algorithm. Even with small M TRIÈST-IMPR achieves very low MAPE value. As expected, larger M corresponds to higher accuracy and for the same M TRIÈST-IMPR outperforms TRIÈST-BASE. Figure 4(b) shows the average time per update in microseconds (μs) for TRIÈST-IMPR as function of M. Larger M requires longer update times (a larger sample implies larger graph on which to count triangles). On average a few hundreds of microseconds are sufficient for handling any update even in very large graphs with billions of edges. Our algorithms can handle hundreds of thousands of edge updates per second with very small error (Fig. 4(a)), and therefore can be used efficiently and effectively in high-velocity contexts.

Alternative edge orders. In all previous experiments the edges are added in their natural order (i.e., in order of their appearance).[12] While the natural order is the most important use case, we have assessed the impact of other ordering on the accuracy of the algorithms. We experiment with both the uniform-at-random (u.a.r.) order of the edges and the random BFS order: until all the graph is explored a BFS is started from a u.a.r. unvisited node and edges are added in order of their visit (neighbors are explored in u.a.r. order). The results for the random BFS order (Fig. 5) and for the

[12]Excluding twitter for which we used the random order, given the lack of timestamps.

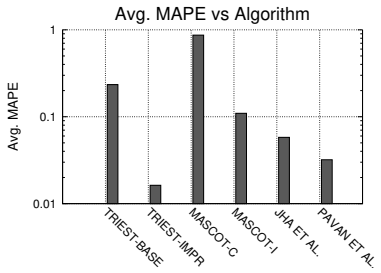
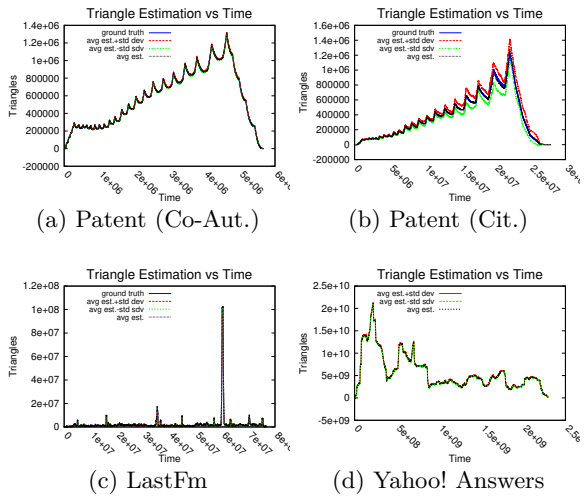


Figure 5: Average MAPE on Patent (Co-Aut.), with $p = 0.01$ – insertion only in Random BFS order. TRIÈST-IMPR has the lowest error.



(a) Patent (Co-Aut.) (b) Patent (Cit.)

(c) LastFm (d) Yahoo! Answers

Figure 6: Evolution of the global number of triangles – fully dynamic case.

u.a.r. (omitted for lack of space) confirm that TRIÈST has the lowest error and is very scalable in every tested ordering.

5.2 Fully-dynamic case

We evaluate TRIÈST-FD on fully-dynamic streams. We cannot compare TRIÈST-FD with the algorithms previously used [18, 27, 32] as they only handle insertion-only streams.

In the first set of experiments we model deletions using the widely used *sliding window model*, where a sliding window of the most recent edges defines the current graph. The sliding window model is of practical interest as it allows to observe recent trends in the stream. For Patent (Co-Aut.) & (Cit.) we keep in the sliding window the edges generated in the last 5 years, while for LastFm we keep the edges generated in the last 30 days. For Yahoo! Answers we keep the last 100 millions edges in the window[13].

Figure 6 shows the evolution of the global number of triangles in the sliding window model using TRIÈST-FD using $M = 200{,}000$ ($M = 1{,}000{,}000$ for Yahoo! Answers). The sliding window scenario is significantly more challenging than the addition-only case (very often the entire sample of edges is flushed away) but TRIÈST-FD maintains good variance and scalability even when, as for LastFm and Yahoo! Answers, the global number of triangles varies quickly.

[13]The sliding window model is not interesting for the Twitter dataset as edges have random timestamps. We omit the results for Twitter but TRIÈST-FD is fast and has low variance.

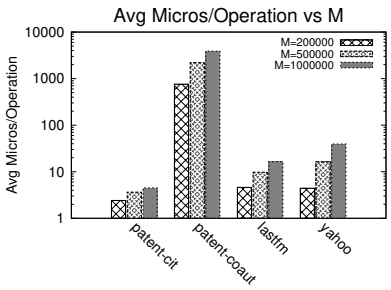


Figure 7: Trade-offs between the avg. update time (μs) and M for TRIÈST-FD.

Graph	M	Avg. Global MAPE	Avg. Local Pearson	Avg. Local ε Err.
LastFM	200000	0.005	0.98	0.02
	1000000	0.002	0.999	0.001
Pat. (Co-Aut.)	200000	0.01	0.66	0.30
	1000000	0.001	0.99	0.006
Pat. (Cit.)	200000	0.17	0.09	0.16
	1000000	0.04	0.60	0.13

Table 4: Estimation errors for TRIÈST-FD.

Continuous monitoring of triangle counts with TRIÈST-FD allows to detect patterns that would otherwise be difficult to notice. For LastFm (Fig. 6(c)) we observe a sudden spike of several order of magnitudes. The dataset is anonymized so we cannot establish which songs are responsible for this spike. In Yahoo! Answers (Fig. 6(d)) a popular topic can create a sudden (and shortly lived) increase in the number of triangles, while the evolution of the Patent co-authorship and co-citation networks is slower, as the creation of an edge requires filing a patent (Fig. 6(a) and (b)). The almost constant increase over time[14] of the number of triangles in Patent graphs is consistent with previous observations of *densification* in collaboration networks as in the case of nodes' degrees [26] and the observations on the density of the densest subgraph [13].

Table 4 shows the results for both the local and global triangle counting estimation provided by TRIÈST-FD. In this case we can not compare with previous works, as they only handle insertions. It is evident that precision improves with M values, and even relatively small M values result in a low MAPE (global estimation), high Pearson correlation and low ε error (local estimation). Figure 7 shows the tradeoffs between memory (i.e., accuracy) and time. In all cases our algorithm is very fast and it presents update times in the order of hundreds of microseconds for datasets with billions of updates (Yahoo! Answers).

Alternative models for deletion. We evaluate TRIÈST-FD using other models for deletions than the sliding window model. To assess the resilience of the algorithm to massive deletions we run the following experiment. We added edges in their natural order but each edge addition is followed with probability q by a mass deletion event where each edge cur-

[14]The decline at the end is due to the removal of the last edges from the sliding window after there are no more edge additions.

rently in the graph is deleted with probability d independently. We run experiments with $q = 3{,}000{,}000^{-1}$ (i.e., a mass deletion expected every 3 millions edges) and $d = 0.80$ (in expectation 80% of edges are deleted). We observe that TRIÈST-FD maintains a good accuracy and scalability even in face of a massive (and unlikely) deletions of the vast majority of the edges: e.g., for LastFM with $M = 200000$ (resp. $M = 1{,}000{,}000$) we observe 0.04 (resp. 0.006) Avg. MAPE. More results are available in our full version online [9].

6. CONCLUSIONS

We presented TRIÈST, the first suite of algorithms that use reservoir sampling and its variants to continuously maintain unbiased, low-variance estimates of the local and global number of triangles in fully-dynamic graphs streams of arbitrary edge/vertex insertions and deletions using a fixed, user-specified amount of space. Our experimental evaluation shows that TRIÈST outperforms state-of-the-art approaches and achieves high accuracy on real-world datasets with more than one billion of edges, with update times of hundreds of microseconds. Interesting directions for future work include the use of color-coding techniques [30], and the extension to 3-profiles and complex graph motifs [11].

Acknowledgments. This work was supported in part by NSF grant IIS-1247581 and NIH grant R01-CA180776.

7. REFERENCES

[1] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella. Graph Sample and Hold: A framework for big-graph analytics. KDD'14.

[2] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. SODA'02.

[3] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient algorithms for large-scale local triangle counting. *TKDD* 4 (3):13:1–13:28, 2010.

[4] J. W. Berry, B. Hendrickson, R. A. LaViolette, and C. A. Phillips. Tolerating the community detection resolution limit with edge weighting. *Phys. Rev. E*, 83(5):056119, 2011.

[5] P. Boldi, M. Rosa, M. Santini, and S. Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. WWW'11.

[6] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. Counting triangles in data streams. PODS'06.

[7] Ò. Celma Herrada. Music recommendation and discovery in the long tail. UPF Technical report, 2009.

[8] E. Cohen, G. Cormode, and N. Duffield. Don't let the negatives bring you down: sampling from streams of signed updates. SIGMETRICS'12.

[9] L. De Stefani, A. Epasto, M. Riondato, and E. Upfal. TRIÈST: Counting local and global triangles in fully-dynamic streams with fixed memory size. *CoRR*, abs/1602.07424, 2016. http://arxiv.org/pdf/1602.07424.pdf.

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

[11] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Dimakis. Beyond triangles: A distributed framework for estimating 3-profiles of large graphs. KDD'15.

[12] A. Epasto, S. Lattanzi, V. Mirrokni, I. O. Sebe, A. Taei, and S. Verma. Ego-net community mining applied to friend suggestion. *Proceedings of the VLDB Endowment*, 2015.

[13] A. Epasto, S. Lattanzi, and M. Sozio. Efficient densest subgraph computation in evolving graph. WWW'15.

[14] R. Gemulla, W. Lehner, and P. J. Haas. Maintaining bounded-size sample synopses of evolving datasets. *VLDBJ*, 17(2):173–201, 2008.

[15] A. Hajnal and E. Szemerédi. Proof of a conjecture of P. Erdős. *Combinat. theo. and its appl., II*, 601–623, 1970.

[16] B. H. Hall, A. B. Jaffe, and M. Trajtenberg. The NBER patent citation data file: Lessons, insights and methodological tools. NBER Technical report, 2001.

[17] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *Int. J. Forecasting*, 22(4): 679–688, 2006.

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

[19] H. Jowhari and M. Ghodsi. New streaming algorithms for counting triangles in graphs. *Computing and Combinatorics*, LNCS 3595, 710–716, 2005.

[20] D. M. Kane, K. Mehlhorn, T. Sauerwald, and H. Sun. Counting arbitrary subgraphs in data streams. ICALP'12.

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

[22] K. Kutzkov and R. Pagh. On the streaming complexity of computing local clustering coefficients. WSDM'13.

[23] K. Kutzkov and R. Pagh. Triangle counting in dynamic graph streams. SWAT'14.

[24] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? WWW'10.

[25] M. Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *TCS*, 407(1):458–473, 2008.

[26] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *TKDD*, 1 (1):2, 2007.

[27] Y. Lim and U. Kang. MASCOT: Memory-efficient and accurate sampling for counting local triangles in graph streams. KDD'15.

[28] M. Manjunath, K. Mehlhorn, K. Panagiotou, and H. Sun. Approximate counting of cycles in streams. ESA'11.

[29] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594): 824–827, 2002.

[30] R. Pagh and C. E. Tsourakakis. Colorful triangle counting and a MapReduce implementation. *IPL*, 112(7):277–281, 2012.

[31] H.-M. Park and C.-W. Chung. An efficient MapReduce algorithm for counting triangles in a very large graph. CIKM'13.

[32] A. Pavan, K. Tangwongsan, S. Tirthapura, and K.-L. Wu. Counting and sampling triangles from a graph stream. VLDB'13.

[33] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. WWW'11.

[34] The Koblenz Network Collection (KONECT). Last.fm song network dataset. http://konect.uni-koblenz.de/networks/lastfm_song.

[35] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos. Doulion: counting triangles in massive graphs with a coin. KDD'09.

[36] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller. Triangle sparsifiers. *JGAA*, 15(6):703–726, 2011.

[37] J. S. Vitter. Random sampling with a reservoir. *TOMS*, 11 (1):37–57, 1985.

[38] Yahoo! Research Webscope Datasets. Yahoo! Answers browsing behavior version 1.0. http://webscope.sandbox.yahoo.com.