A Space Efficient Streaming Algorithm for Triangle Counting Using the Birthday Paradox

By: Madhav Jha et al (KDD 2013)

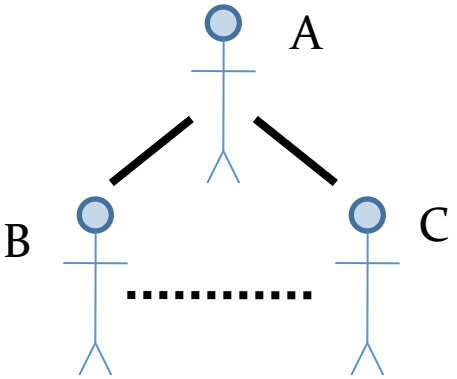
Presented by: Ahmed El-Kishky

Triangles

- A triangle is three vertices A,B,C such that there exists edges (A,B), (B,C), (A,C)
- There is a rich body of literature on analysis of triangles and counting algorithms.

Motivation

Friends of friends tend to become friends themselves!

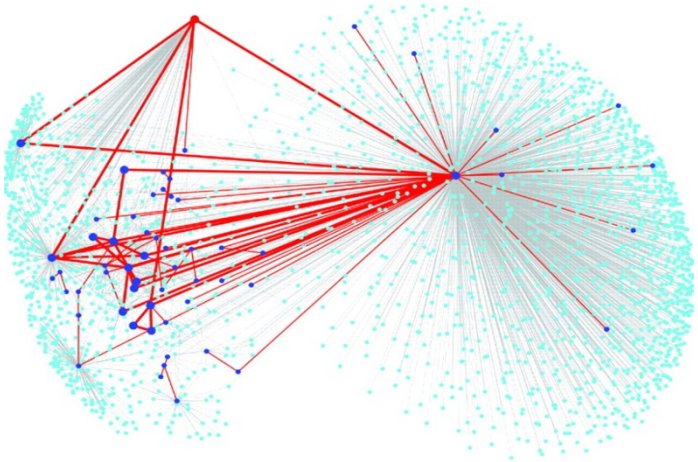


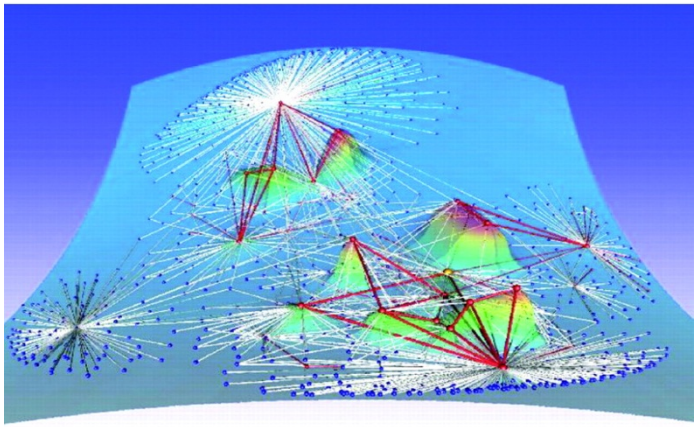
Stanley Wasserman, Kathrine Faust, 1994. ***Social Network Analysis: Methods and Applications****. Cambridge: Cambridge Universit Press*

(left to right) Paul Erdös , Ronald Graham, Fan Chung Graham

More Motivations

Key Idea: Connected regions of high curvature (i.e., dense in triangles) indicate a common topic!

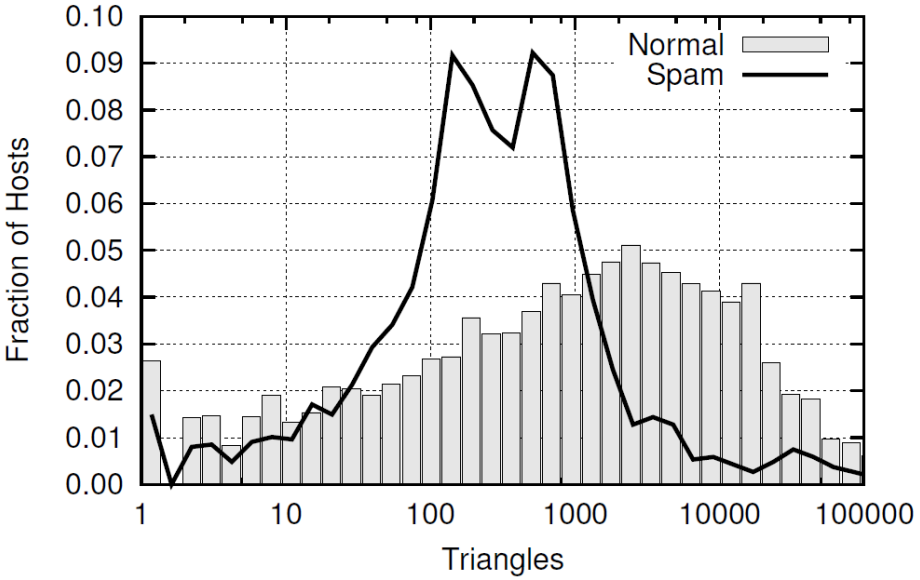




Eckmann-Moses, Uncovering the Hidden Thematic Structure of the Web (PNAS, 2001)

More Motivations

Key Idea: Triangle Distribution among spam hosts is significantly different from non-spam hosts!

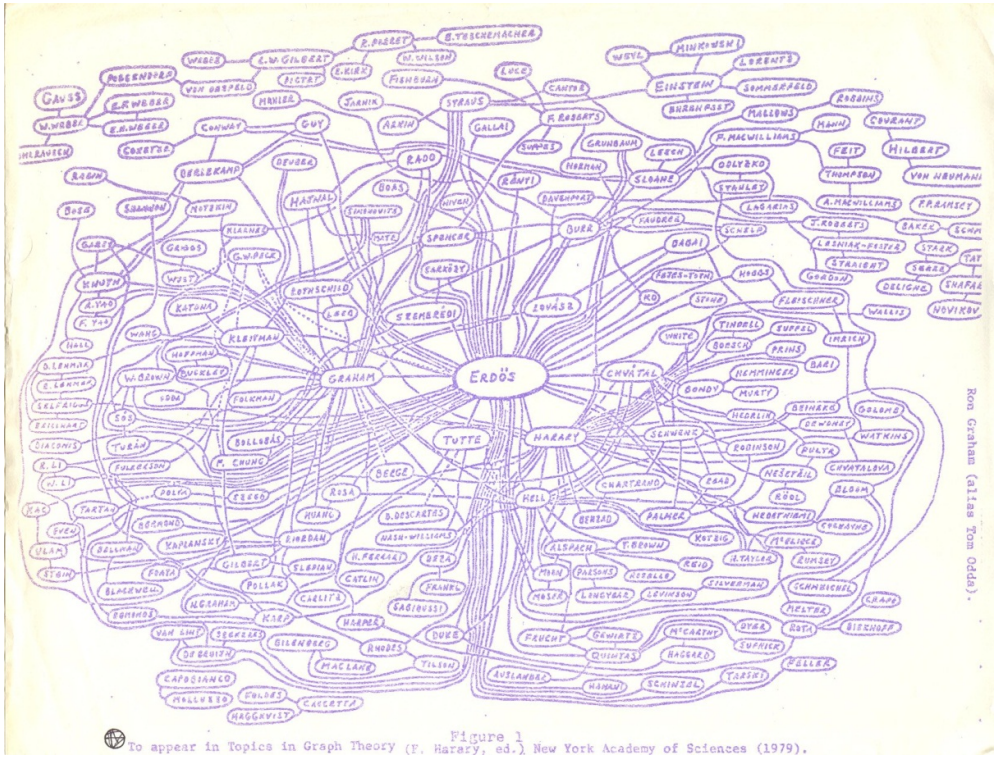


Triangles used for Web Spam Detection (Becchetti et al. KDD '08)

More Motivations

Triangles used for assessing Content Quality in Social Networksç

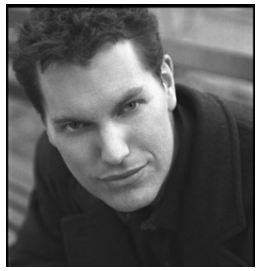
Key Claim: The amount of triangles in the self-centered social network of a user is a good indicator of the role of that user in the community!



Welser, Gleave, Fisher, Smith *Journal of Social Structure 2007*

More Motivations

In Complex Network Analysis two frequently used measures are:



Clustering coefficient of a vertex

$$C(v) = \frac{3 \times \ \#\text{triangles(v)}}{\binom{d(v)}{2}}$$

Transitivity ratio of the graph

$$C = \frac{3 \times \ \#\text{triangles}}{\#connected\ triples}$$

Even More Motivations

- Numerous other applications including :
- Motif Detection/ Frequent Subgraph Mining (e.g., Protein-Protein Interaction Networks)
- Community Detection (Berry et al. ‘09)
- Outlier Detection (Tsourakakis ‘08)
- Link Recommendation

Previous Algorithms

- Counting triangles is important, but graphs can grow prohibitively large
- Methods such as random sampling, using the map reduce paradigm, using external memory, as well as massive parallelism have been employed
- Most methods need to store a large fraction of the data

The algorithm

- A space-efficient algorithm that approximates transitivity (global clustering coefficient) and total triangle count
- Maintains a “sketch” of the graph to perform estimates
- Single pass algorithms
- Input is a stream of edges (no edge / vertex deletion)
- O(sqrt(n)) space (n is the number of vertices)

The Streaming Setting

- Let G be a simple undirected graph with n vertices and m edges.
- Let T denote the number of triangles in the graph and W be the number of wedges (paths of length 2)
- Let k be the measure of transitivity = 3T / W
- The streaming algorithm considers a sequence of (distinct) edges e_1, e_2 … e_m
- Let G_t be the graph at time t, formed by the edge set $\{e_i\}$: i<=t.

High Level Intuition

- A wedge is closed if it participates in a triangle and open otherwise.
- $k = 3T / W$ is exactly the probability that a uniform random wedge is closed.
- This formula gives us a randomized method for estimating k and consequently estimating T by generating a set of (independent) uniform random wedges and finding the fraction that are closed.

High Level Intuition

- The formula $k = 3T / W$ is dependent on knowledge of the number of closed wedges and total number of wedges.
- How do we sample wedges from a stream of edges?

Streaming Triangles

Array $edge\_res[1 \cdots s_e]$: This is the array of *reservoir edges* and is the primary subsample of the stream maintained.

New wedges $\mathcal{N}_t$: This is a list of all wedges involving e_t formed only by edges in *edge_res*. This may often be empty, if e_t is not in *edge_res*. We do not necessarily maintain this list explicitly, and we discuss implementational details later.

Variable *tot_wedges*: This is the total number of wedges formed by edges in *edge_res*.

Array $wedge\_res[1 \cdots s_w]$: This is an array of *reservoir wedges* of size s_w.

Array $isClosed[1 \cdots s_w]$: This is a boolean array. We set $isClosed[i]$ to be true if wedge $wedge\_res[i]$ is found to be closed.

Streaming Triangles

Algorithm 1: STREAMING-TRIANGLES(s_e, s_w)

1 Initialize *edge_res* of size s_e and *wedge_res* of size s_w. For each edge e_t in stream,
2 Call UPDATE(e_t).
3 Let ρ be the fraction of entries in *isClosed* set to `true`.
4 Set $\kappa_t = 3\rho$.
5 Set $T_t = [\rho t^2 / s_e(s_e - 1)] \times tot\_wedges$.

Streaming Triangles

Algorithm 2: UPDATE(e_t)

1 For every reservoir wedge *wedge_res*[i] closed by e_t, set *isClosed*[i] = `true`.
2 Flip a coin with heads probability $= 1 - (1 - 1/t)^{s_e}$. If it flips to tails, stop and proceed to the next edge in stream. (So remaining code is processed only if coin comes heads.)
3 Choose a uniform index $i \in [s_e]$. Set *edge_res*[i] $= e_t$.
4 Determine $\mathcal{N}_t$ and let *new_wedges* $= |\mathcal{N}_t|$.
5 Update *tot_wedges*, the number of wedges formed by *edge_res*.
6 Set $q =$ *new_wedges*/*tot_wedges*.
7 For each index $i \in [s_w]$,
8 Flip coin with heads probability q.
9 If tails, continue to next index in loop.
10 Pick uniform random $w \in \mathcal{N}_t$ that involves e_t.
11 Replace *wedge_res*[i] $= w$. Reset *isClosed*[i] = `false`.

The Birthday Paradox

- The event of at least two persons (edges) having the same birthday (vertex) is complementary to all persons (edges) having different birthdays (none sharing a vertex)
- As such as long as W <= m (pretty much always true for networks) O(sqrt(n)) edges suffice.

The Birthday Paradox Continued

- Because a small number of uniform random edges can give enough wedges to perform wedge sampling, these wedges can be used to estimate k.
- We can reverse the birthday paradox to estimate W.

Birthday Paradox for Wedges

LEMMA 1 (BIRTHDAY PARADOX FOR WEDGES). *Let G be a graph with m edges and $\mathcal{S}$ be some fixed subset of wedges in G. Let $\mathcal{R}$ be a set of s uniformly and independently selected edges (with replacement) from G. Let X be the random variable denoting the number of wedges in $\mathcal{S}$ formed by edges in $\mathcal{R}$.*

1. $\mathbf{E}[X] = \binom{s}{2}(2|\mathcal{S}|/m^2)$.

Proof of Birthday Paradox

PROOF. The first part is an adaptation of the birthday paradox calculation. Let the set $\mathcal{R} = \{r_1, r_2, \ldots, r_s\}$. We define random variables $X_{i,j}$ for each $i, j \in [s]$ with $i < j$. Let $X_{i,j} = 1$ if the wedge $\{r_i, r_j\}$ belongs to $\mathcal{S}$ and 0 otherwise. Then $X = \sum_{i<j} X_{i,j}$.

Since $\mathcal{R}$ consists of uniform i.i.d. edges from G, the following holds: for every $i < j$ and every (unordered) pair of edges $\{e_\alpha, e_\beta\}$ from E, $\Pr[\{r_i, r_j\} = \{e_\alpha, e_\beta\}] = 2/m^2$. This implies $\Pr[X_{i,j} = 1] = 2|\mathcal{S}|/m^2$. By linearity of expectation, we have $\mathbf{E}[X] = \binom{s}{2}\mathbf{E}[X_{1,2}] = \binom{s}{2}\Pr[X_{1,2} = 1] = \binom{s}{2}(2|\mathcal{S}|/m^2)$, as required.

Estimating T

1. $\mathbf{E}[X] = \binom{s}{2}(2|\mathcal{S}|/m^2)$. = total_wedges

This directly implies that

$$T_t = [\rho t^2 / s_e(s_e - 1)] \times tot\_wedges.$$

Another Insight

- As the reservoir of wedges is maintained, closure of wedges by the future edges in the stream is checked for. But there are closed wedges that cannot be verified, because the closing edge may have already appeared in the past.
- Insight: In each triangle, there is exactly one wedge whose closing edge appears in the future.
- When approximated, these future-closed edges = one third of closed wedges.
- Hence, the factor 3 that pops up in Streaming-Triangles

Experimental Results

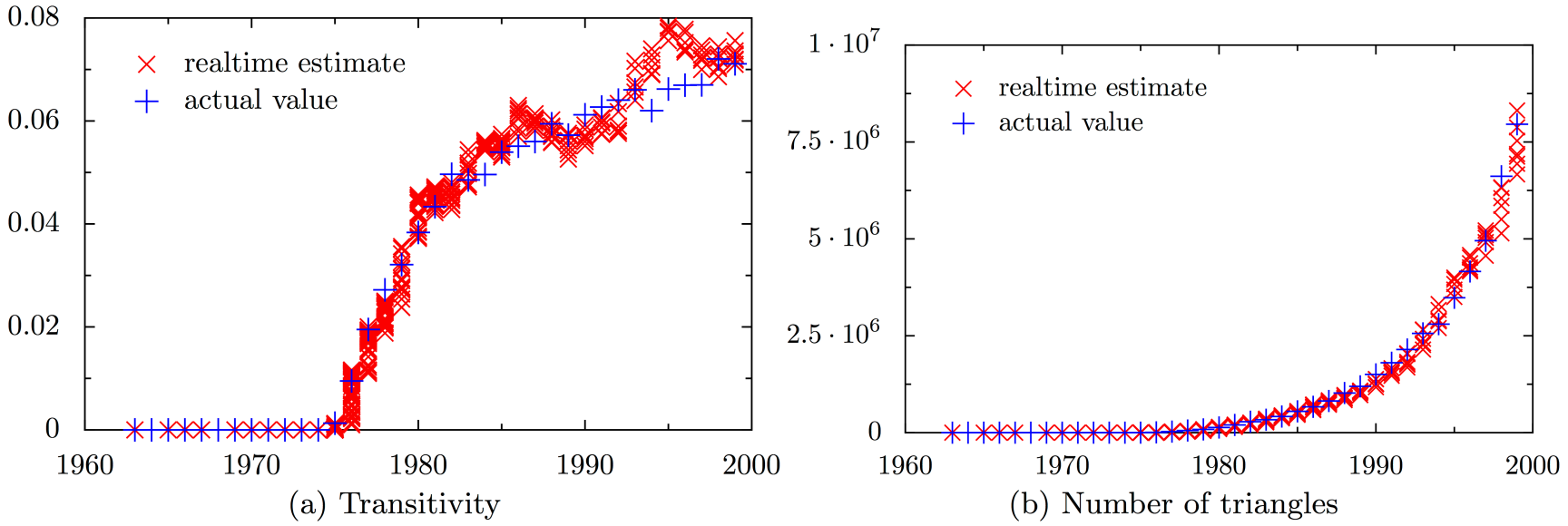
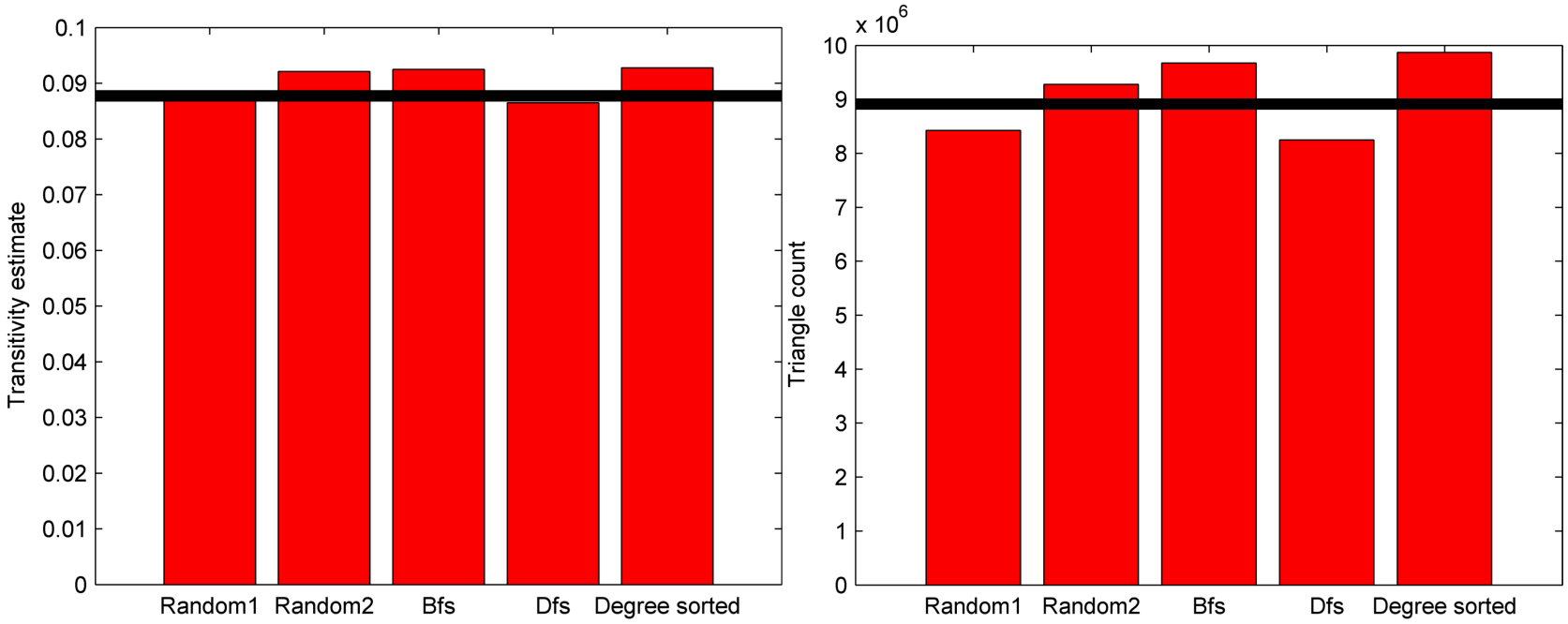


Figure 1: Realtime tracking of number of triangles and transitivities on cit-Patents (16M edges), storing only 60K edges from the past. (Absolute values are shown with time (in years) on the x-axis.)

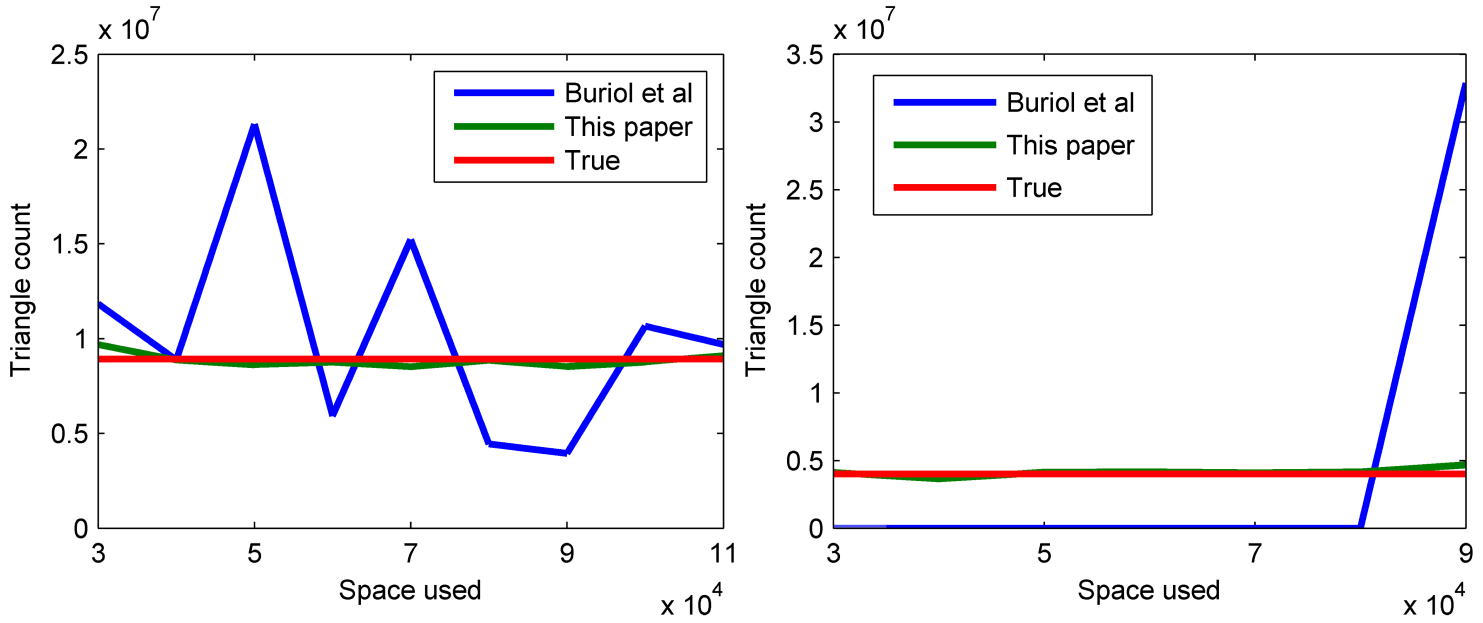
Experimental Results



(a) Transitivity (b) Number of triangles

Figure 2: Output of a single run of STREAMING-TRIANGLES on various orderings of the data stream for web-NotreDame. The true value is given by the thick black line.

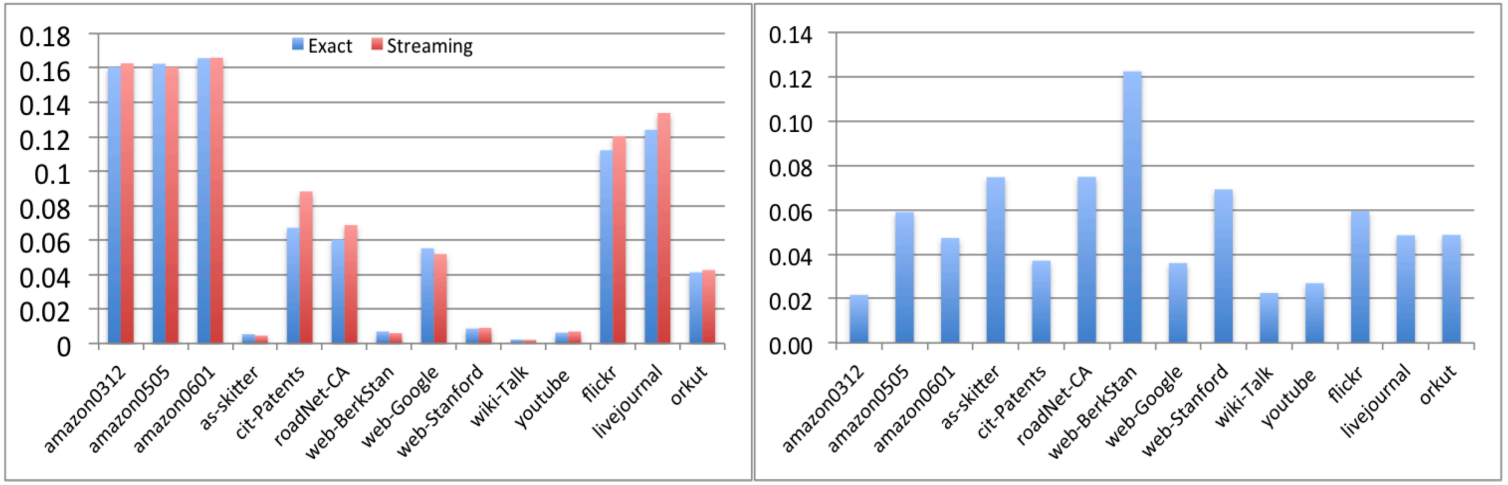
Experimental Results



(a) web-NotreDame (b) amazon0505

Figure 5: Comparison of our algorithm with Buriol et al [9] for amazon0505 and web-NotreDame. We use a 20K edge reservoir and variable wedge reservoir for our storage.

Experimental Results



(a) Transitivity (b) Triangles

Figure 6: Output of a single run of STREAMING-TRIANGLES on a variety of real datasets with 20K edge reservoir and 20K wedge reservoir. The plot on the left gives the estimated transitivity values (labelled streaming) alongside their exact values. The plot on the right gives the relative error of STREAMING-TRIANGLES's estimate on triangles T. Observe that the relative error for T is mostly below 8%, and often below 4%.