SAND2014-19552C

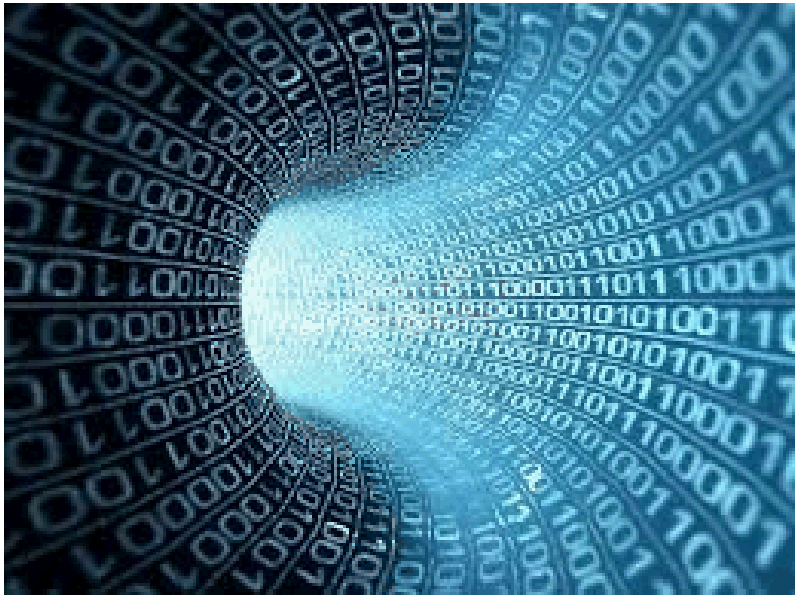
Sampling and Streaming Algorithms for Counting Small Patterns in BIG Graphs

Ali Pinar
Sandia National Laboratories

Joint work with C. Seshadhri, T. Kolda, and M. Jha

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sampling may be the key to process large data sets

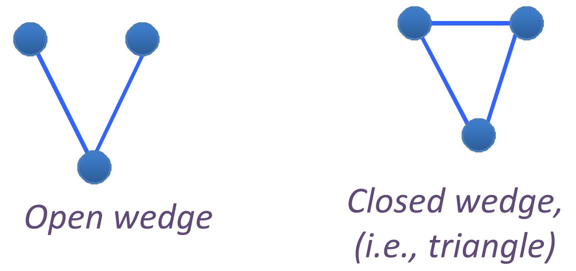


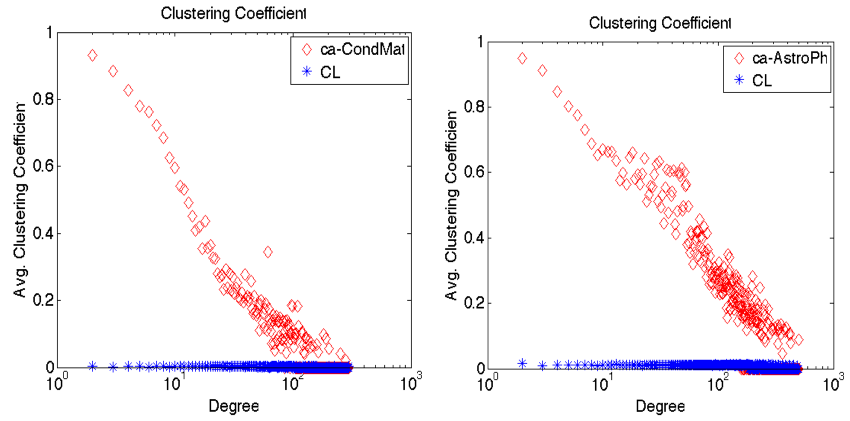
Source: http://www.greenbookblog.org/wp-content/uploads/2012/03/big-data.jpg

- Sizes of the modern data sets redefine the landscape for algorithmic research.
  - Single pass through the data may be a luxury.
- In many applications the speed of data is the challenge.
- Sampling/streaming algorithms can identify general trends in the data.
  - but not find needle in a haystack.
- The goal of sampling is to provide
  - good estimations with error/confidence bounds,
  - by looking at a small portion of the data.
- Sampling is not an alternative to parallelism.
  - They get along well together.

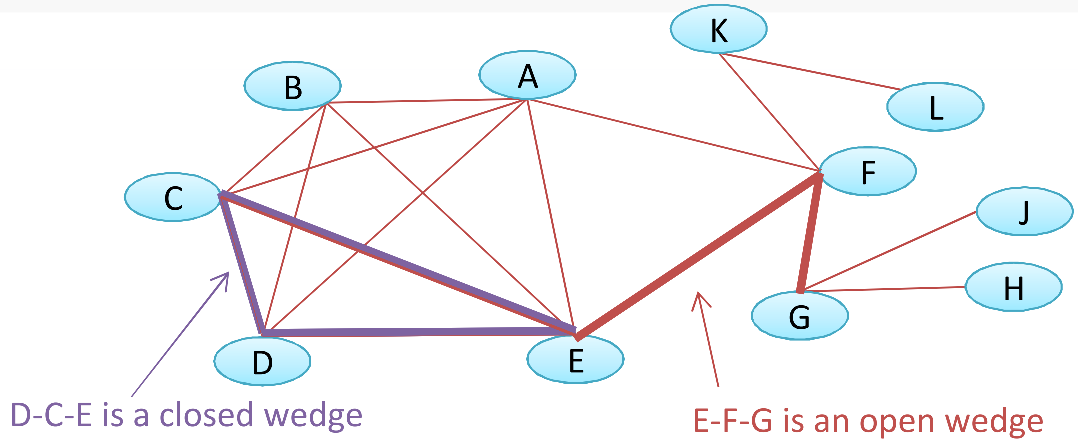
Triangles are critical for graph analysis

- Interpreted in many ways in social sciences.
  - Identifier for bridges between communities.
  - Likelihood to go against norms
- Applied to spam detection
- Used to compare graphs
- Proposed as a guide for community structure.
- Stated as a core feature for graph models [Vivar&Banks11]
  - Cornerstone for Block Two-level Erdos-Renyi (BTER) model
- Rich set of algorithmic results
  - Algorithms, runtime analysis, streaming algorithms, MapReduce, …
  - Enables decomposition into dense blocks
  - Well-defined property of the graph, not an artifact of the algorithm





Algorithms for important metrics: transitivity for large graphs



Enumeration: Find *every* wedge. Check if each is closed.

Transitivity = C = # closed wedges / # wedges

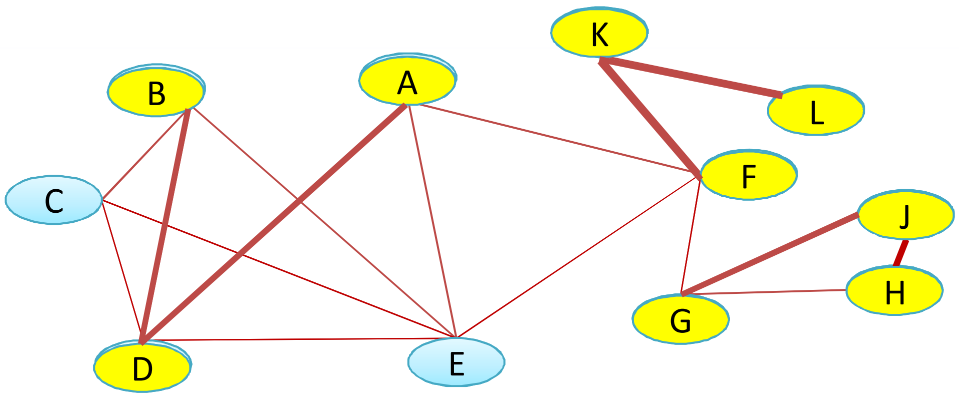
= 3*#triangles/ # wedges

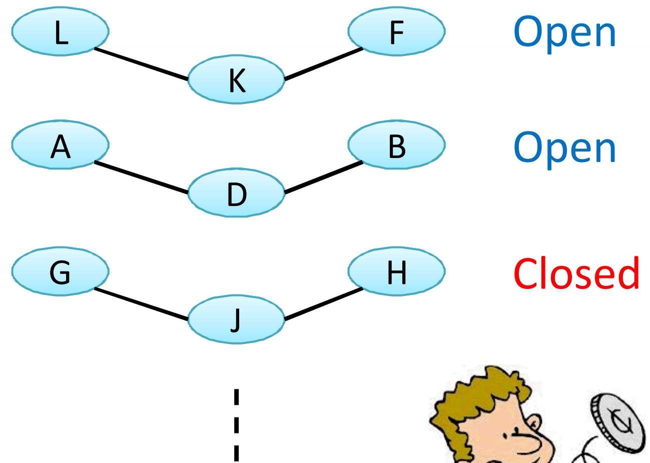
Sampling: Sample a few wedges (uniformly). Check if each is closed.

C = # closed sampled wedges / # sampled wedges

Seshadhri, P., Kolda, *SIAM Intl. Conf. Data Mining* 2013, Best Research Paper award

Wedge sampling to compute transitivity

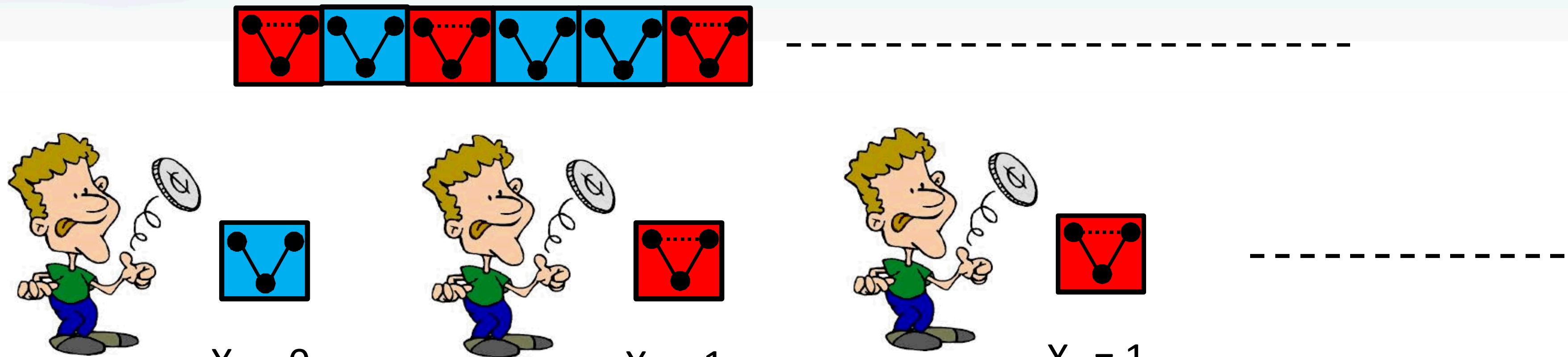




- C = 3T/W = fraction of closed wedges
- Consider list of all wedges, indexed with open/closed
- Pick a uniform random wedge. X = 1 if wedge is closed. Else X = 0
- X is Bernoulli random variable

  and E[X] = fraction of closed wedges = C = 3T/W

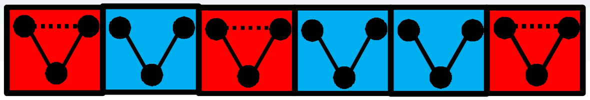
Repeat, repeat, repeat

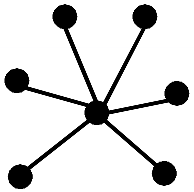


$X_1 = 0$ $X_1 = 1$ $X_2 = 1$

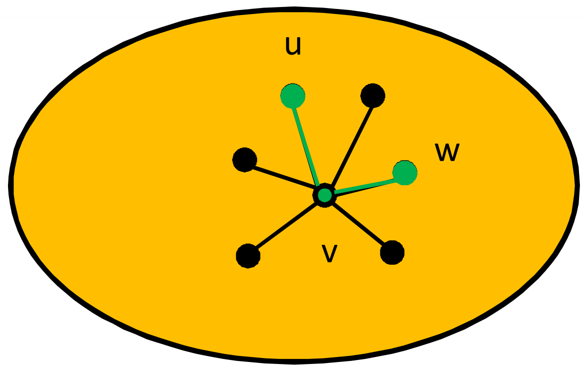
- Perform k independent experiments. Let $Y = (1/k) \sum_i X_i$
  - Y is fraction of closed wedges in sample
  - E[Y] = C. Y converges to C as k grows
- [Chernoff-Hoeffding]: $\Pr[|Y - \tau| > \varepsilon] < e^{-k\varepsilon^2}$
  - $k = \varepsilon^{-2}\log(1/\delta)$. With prob > 1- δ, estimate is accurate within ε
  - With 38K samples, error < 0.01 with prob > 0.999
  - Number of samples independent of graph size

We do not need to generate a wedge list





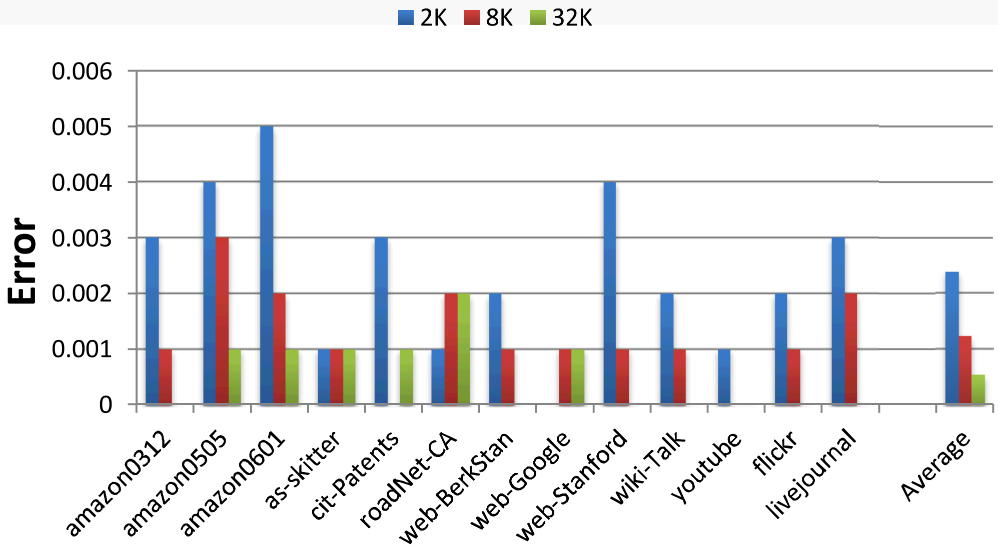
$$W_v = \binom{d_v}{2}$$

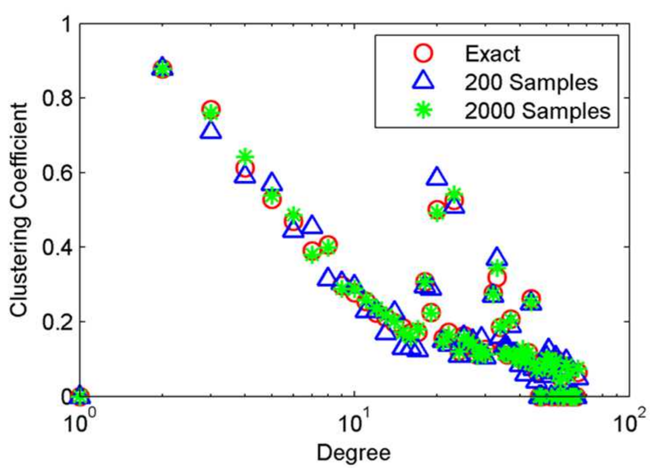


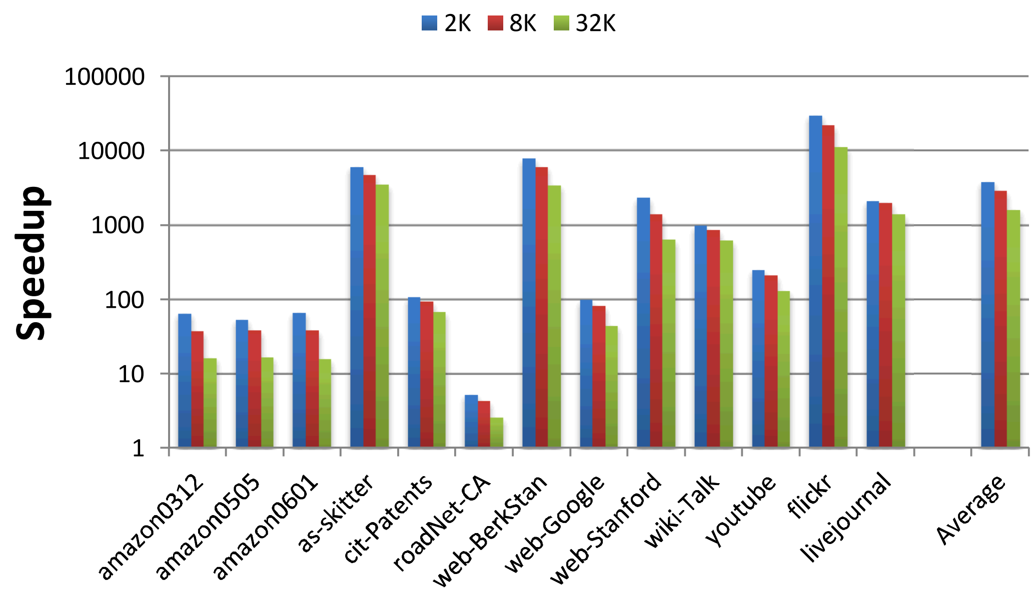
- But list of wedges not possible to generate. So how to get random wedge?
- Pick vertex v with probability W_v/W
- Pick two uniform random neighbors of v to get wedge (u,v,w)
  - This is a uniform random wedge
- So simply repeat this many times to get a set of wedges. Output fraction of closed wedges as estimate for C

7/2/2016

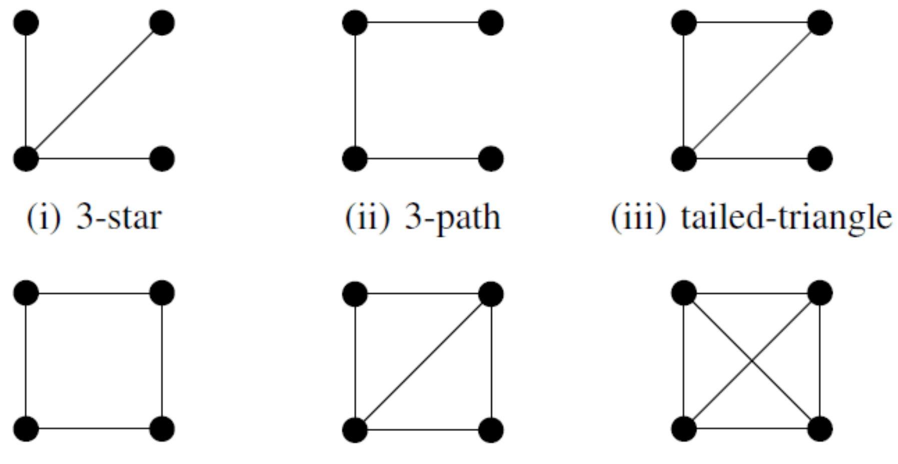
Wedge sampling is effective in practice

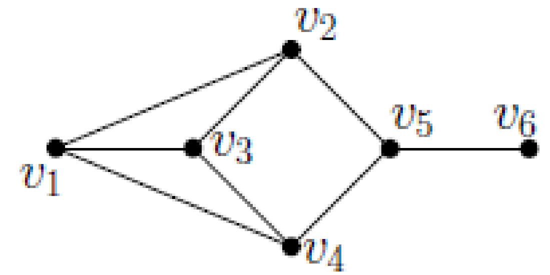






Beyond 3 vertices: how about 4?

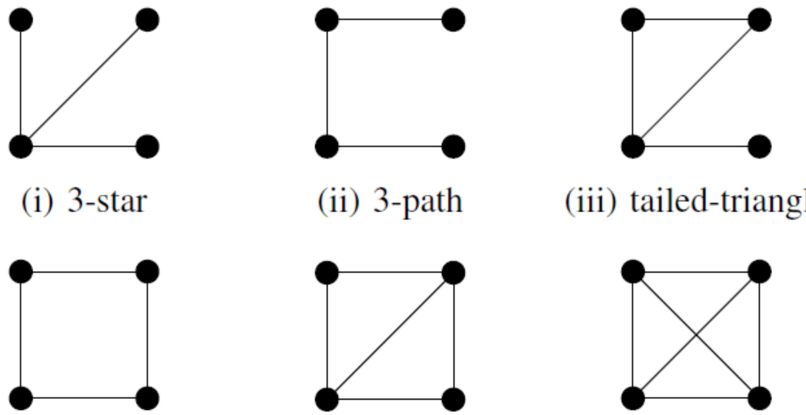

(i) 3-star (ii) 3-path (iii) tailed-triangle (iv) 4-cycle (v) chordal-4-cycle (vi) 4-clique



$$\begin{pmatrix} 1 & 0 & 1 & 0 & 2 & 4 \\ 0 & 1 & 2 & 4 & 6 & 12 \\ 0 & 0 & 1 & 0 & 4 & 12 \\ 0 & 0 & 0 & 1 & 1 & 3 \\ 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix} = \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{pmatrix}$$

- Much richer set of (connected) patterns
- Induced, C_i, vs. Non-induced, N_i
  - (Vanilla) subgraph: take subset of edges
  - Induced subgraph: take subset of vertices, take all edges in them
  - Getting vanilla counts form induced subgraph counts is not hard

Exact counting is not scalable



(i) 3-star (ii) 3-path (iii) tailed-triangle

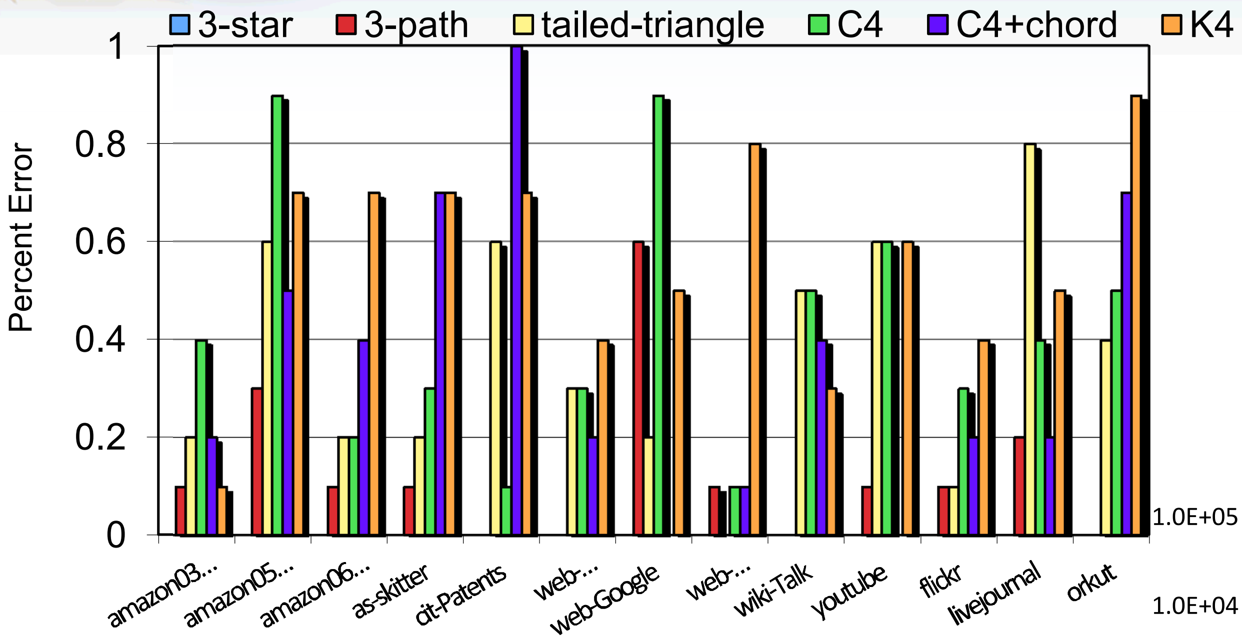
(iv) 4-cycle (v) chordal-4-cycle (vi) 4-clique

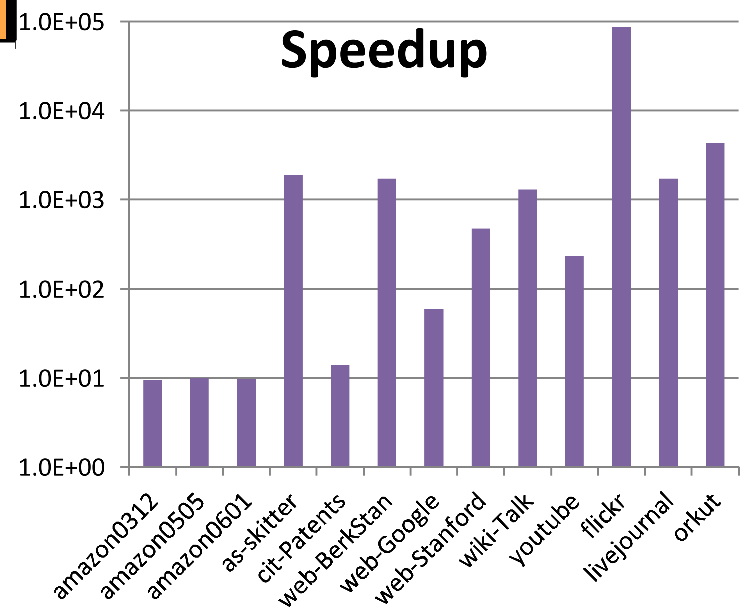
Past approximate counting work does not scale, either

- MCMC methods, color coding, graph sparsification
- No provable methods, accuracies at best ~10%, often need computer clusters
- No results for (say) 100M edges

Graph	n	m	3-path	Tail-tri	4-cycle	4-clique	Time
Web-Berk	600K	6M	10B	400B	20B	1B	2 hrs
Flickr	1M	15M	7T	100M	100B	25B	60 hrs
Orkut	3M	200M	10T	1T	70B	3B	19 hrs

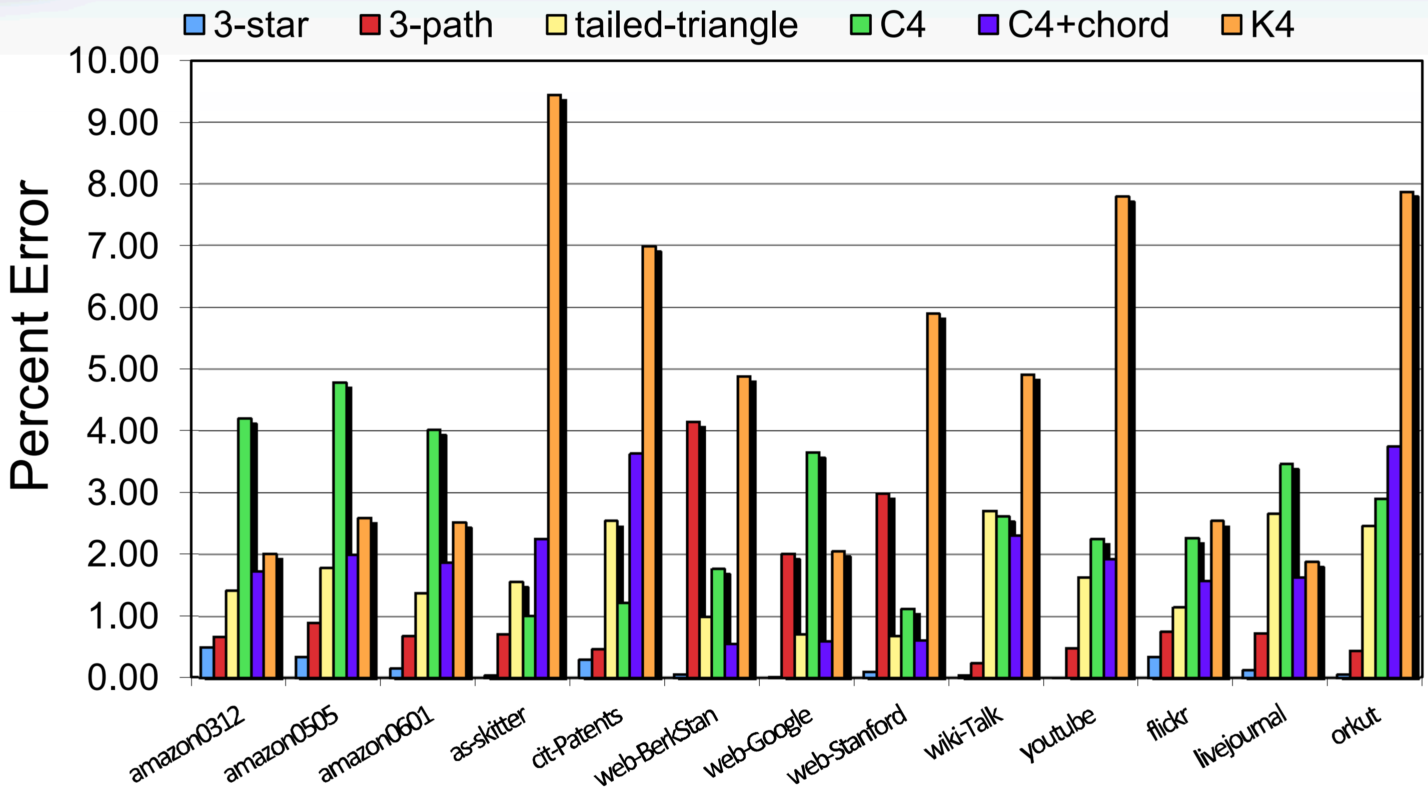
3-path sampling algorithm is fast and accurate





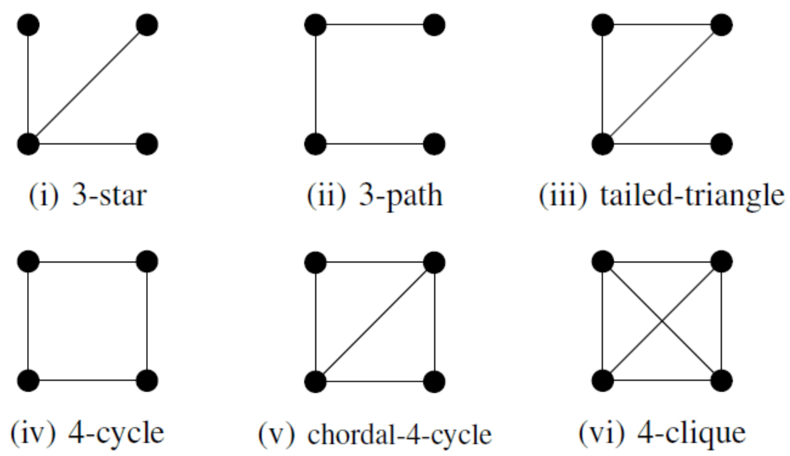
Graph	Time (exact)	Path-sampling
Web-Berk	2 hrs	3 sec
Flickr	60 hrs	2 sec
Orkut	19 hrs	16 sec

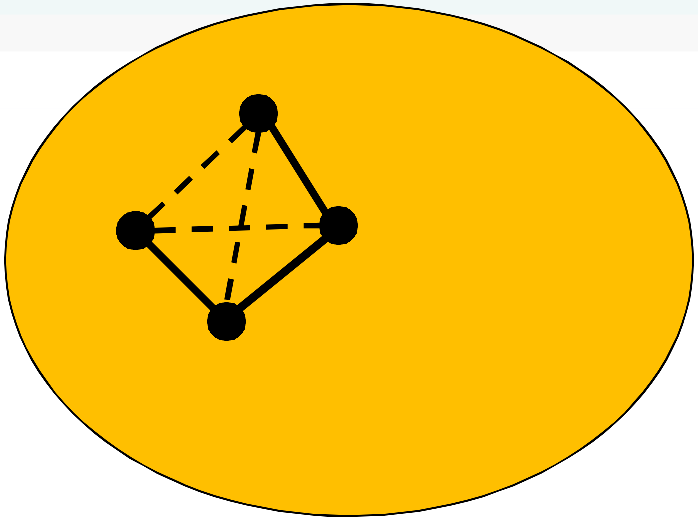
Sampling gives provable accurate results



- Algorithm outputs hard error bounds for any desired confidence
  - "With confidence > 99.9%, the output is within 3% of true answer."
- No assumption on the graph; probability is over the randomness of the algorithm.

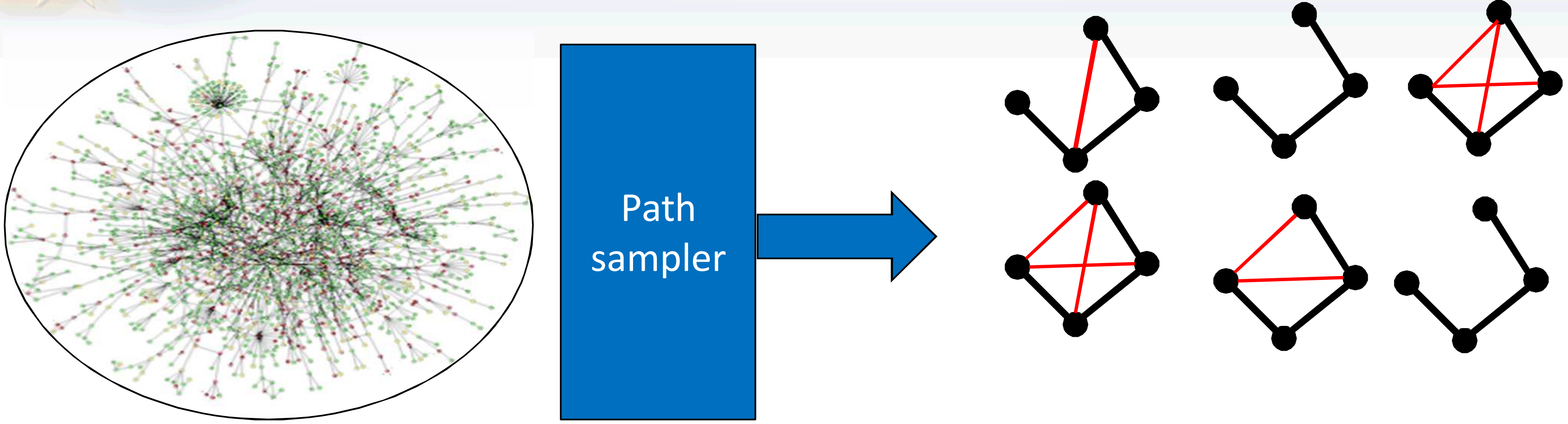
The method





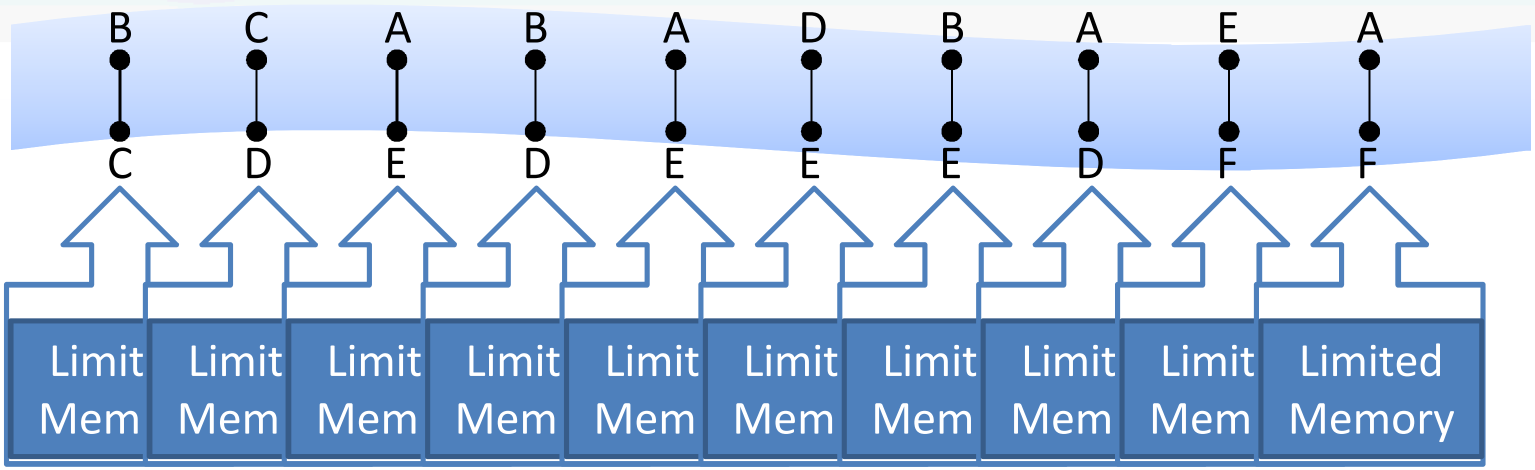
- Except for 3-stars, each pattern contains a 3-path
- Sample set of uniform random 3-path, check the vertices to see what pattern is induced
  - We do not need to generate a full list of 3-paths.
- Extrapolate these counts to get estimates

The big picture



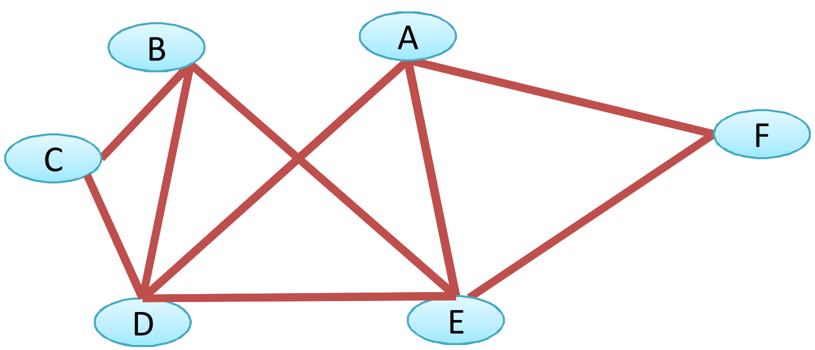
- Use pattern counts from samples to estimate true count
- Not hard to argue that our output is unbiased estimator of true count
- No assumption on graph, probability over randomness of algorithm
- How many samples needed to get accurate estimates?
  - For better results, we sample "centered 3-paths"

Streaming Triangle Counting



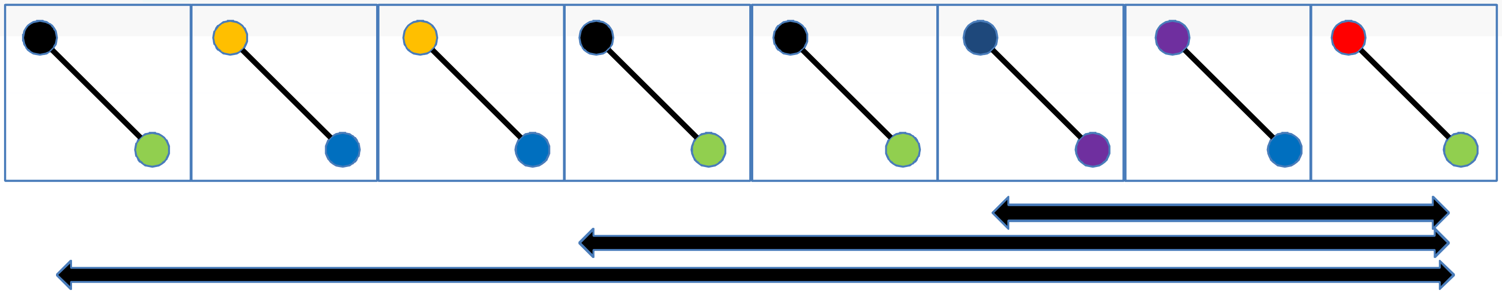
Triangles so far: 4

Graph seen so far:



- Data streams important for situational awareness
  - Streaming algorithms also useful for large data sets
- Algorithmically
  - See each edge only once
  - Either take action or lose that piece of information forever

Real-world messiness



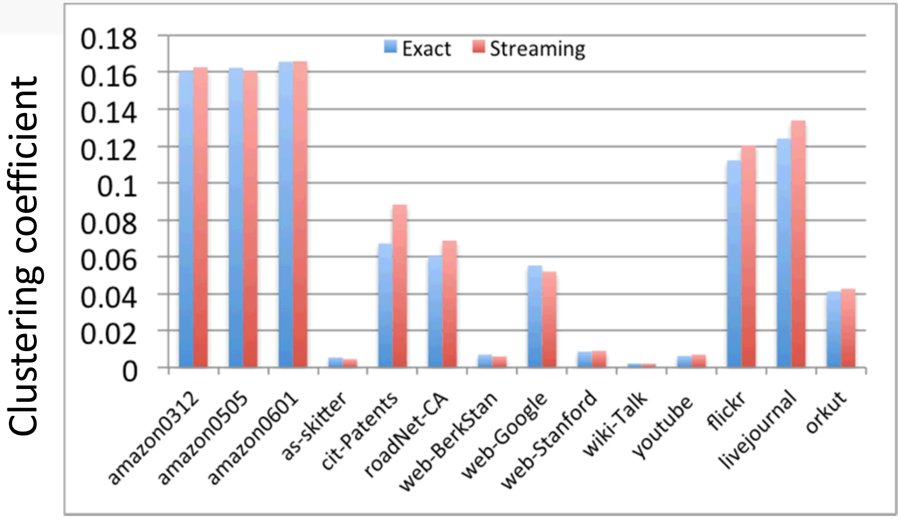
- Real-world streams are multigraphs: edges can be repeated
- There is no “true” graph. It depends on how you aggregate

Standard approaches and their drawbacks

- There are no repeats. Assume graph is simple
  - Removing repeated edges requires extra pass over edges
  - Assumption of no repeats is expensive to enforce
- Aggregate every edge seen. The “window” is all of history
  - Not clear how to store information of various time-windows simultaneously

We can analyze streams of edges

- Approximating triangle counts and transitivity in graph stream with repeated edges
  - No preprocessing.
    Works with raw stream
- Information on multiple time windows with same data structures
  - Potential solution to the problem of how much data to store

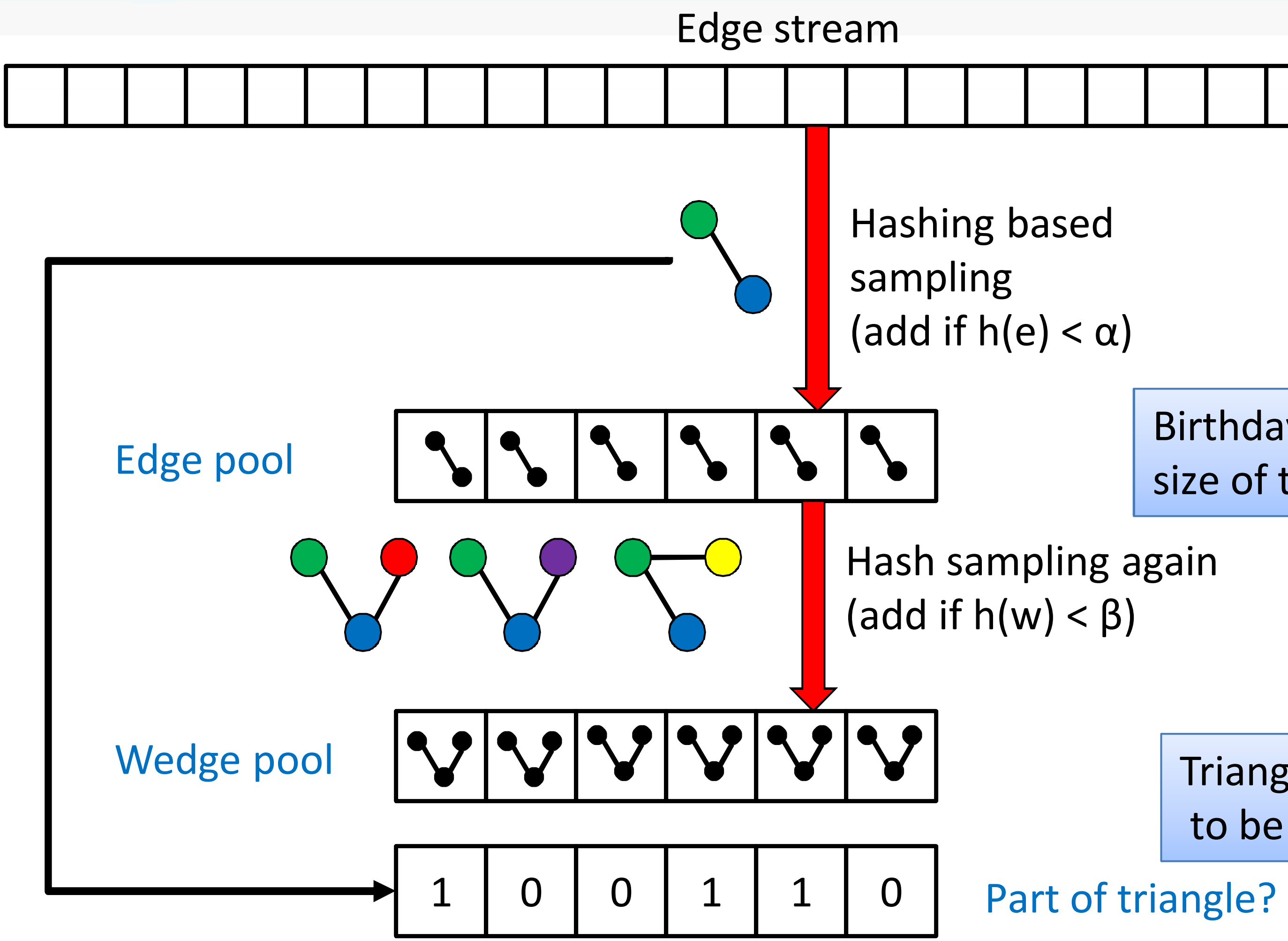


- Edge pool size: 20K; Wedges pool size is: 20K
- Jha, Seshadhri, P. KDD13, Best Student Paper award

- Provable bounds on accuracy, excellent empirical behavior
- Based on methods in [Jha-Seshadhri-Pinar13], but needs new ideas to overcome issues

Core Idea:

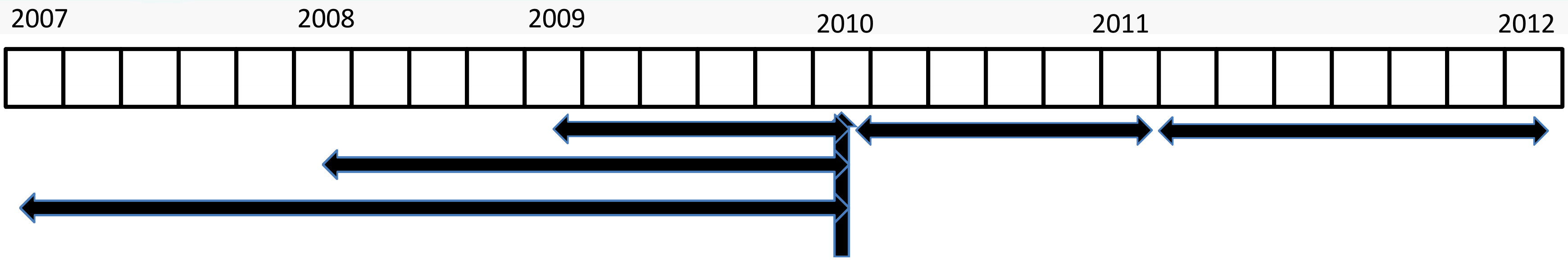
Wedge sampling on a stream



Birthday Paradox tells the size of the hedge pool

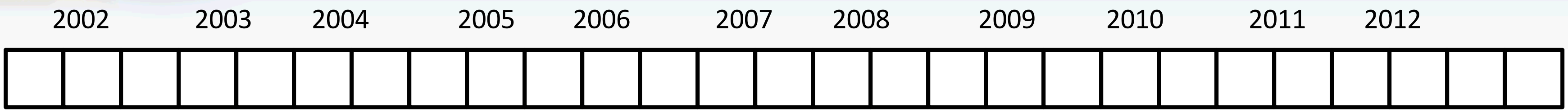
Triangle counts need to be debiased

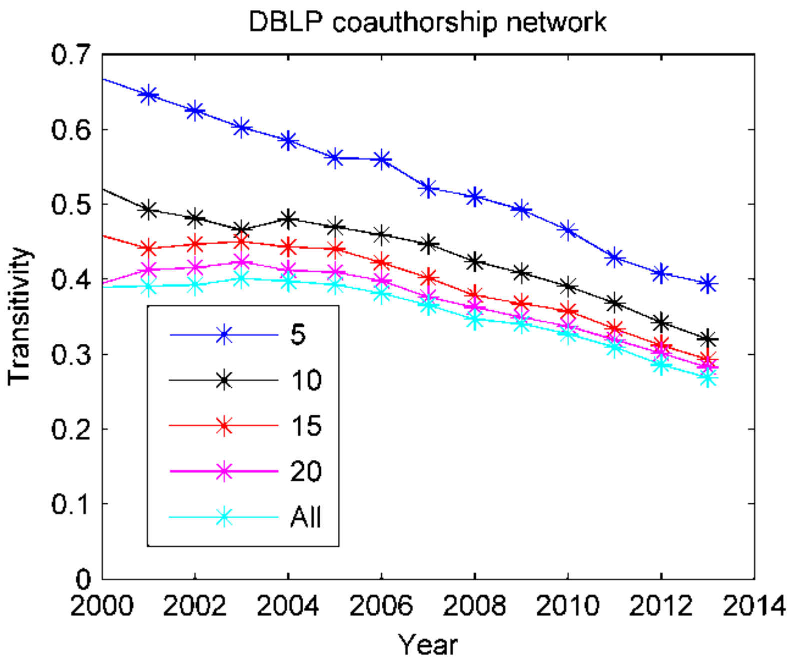
Case study: DBLP graph

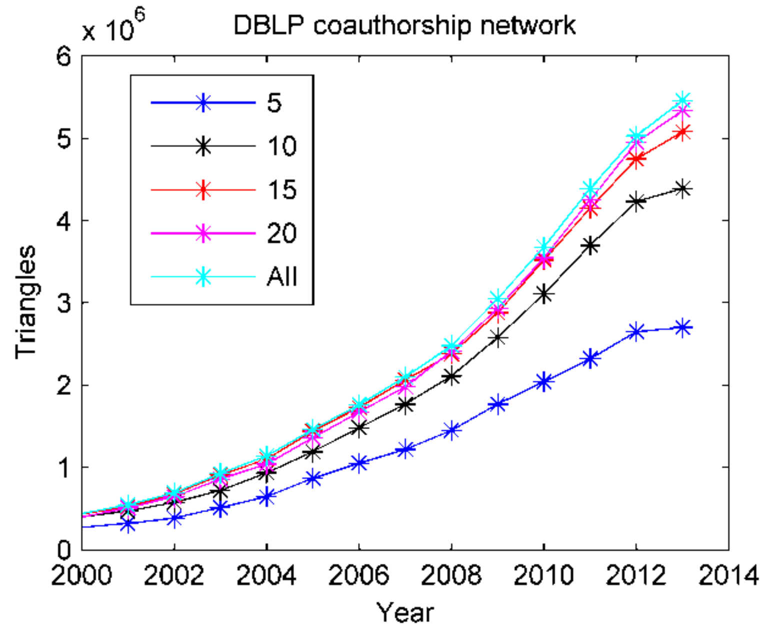


- DBLP co-authorship graph: all paper records over 50 years gives graph stream
  - Naturally repeated edges. Colleagues work together for many papers
  - Size = 3600K, non-repeated edges = 254K
- For graph G[t:t+Δt], there is associated transitivity and triangle count
  - How does this vary with t and Δt?

Triangle trends in DBLP graph

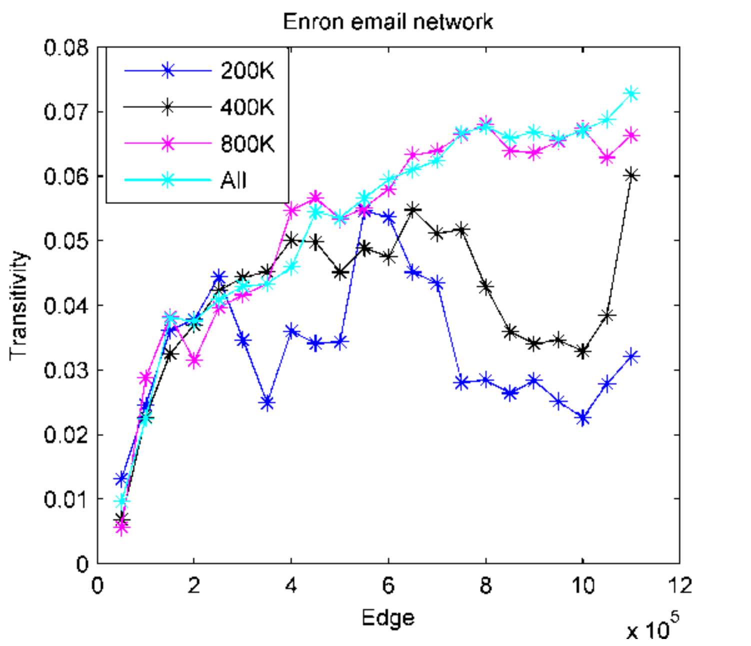


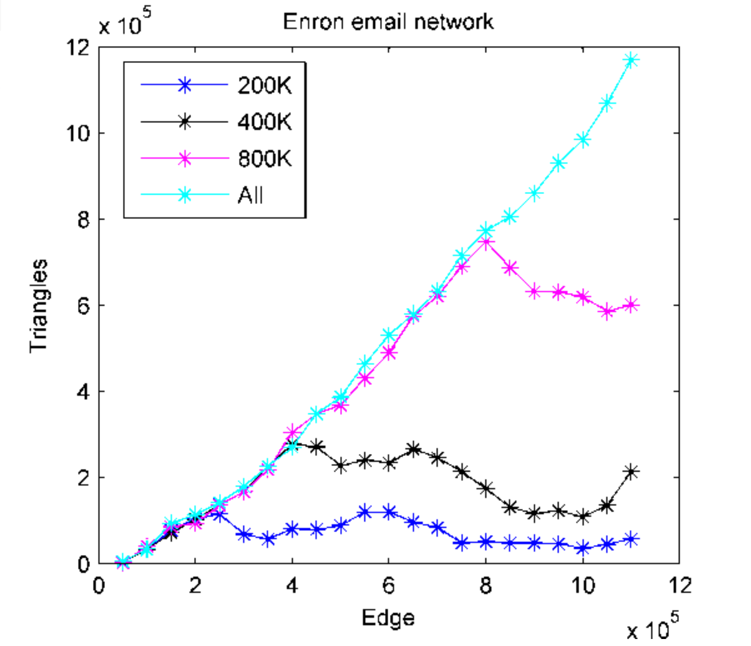




- Size = 3600K, non-repeated edges = 254K
- Results obtained with storing 30K edges

Triangle trends in Enron graph

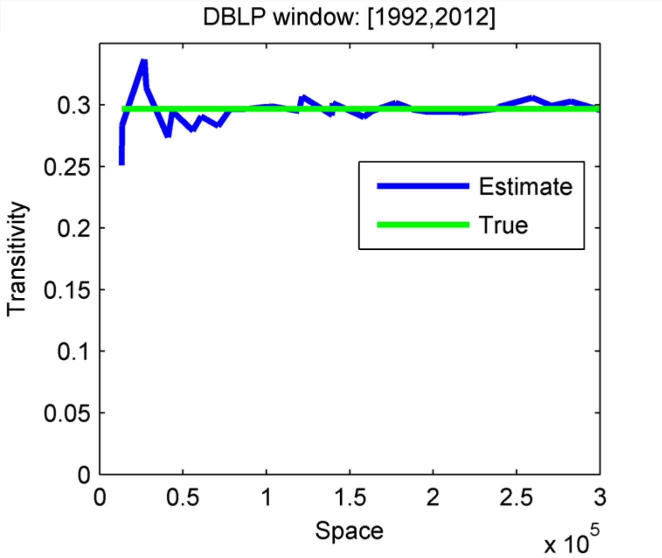




- Enron email network: stream size 1100K, non-repeated 300K
- Storage used = 8K
- Trends “opposite” to DBLP graph

Streaming Algorithm Features

- Only two parameters α, β
  - No knowledge of graph required
- Provable guarantee on expectation
  - Provable variance bound (though not useful in practice)
- Space around 1% of total stream
- Accuracy always within 5%



Conclusions

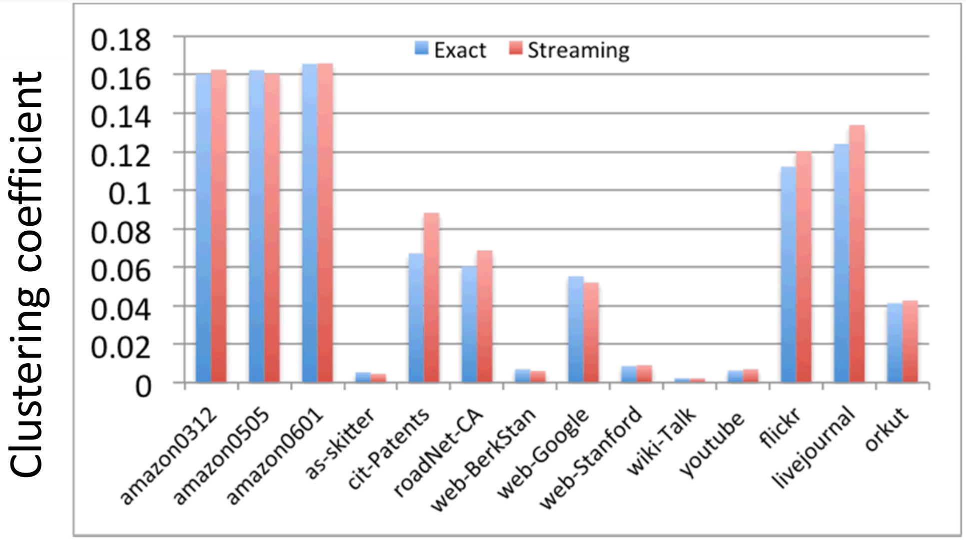
- If you need the *counts* of small patterns on a large graph, use sampling (streaming).
- If you need a list of small patterns,
  - If the output size is small, enumerate!
  - If not, the list should be an input to another process, and let's talk about the full process.
- Wedge sampling enables efficient computation of many triadic measures
  - Has provable error/confidence bounds
  - Amenable to handling distributed data
  - Extended to streaming analysis
    - Can handle repeated edges and different time windows
- Similar techniques can be used for 4-vertex patterns
  - Used 3-path sampling instead of wedge sampling

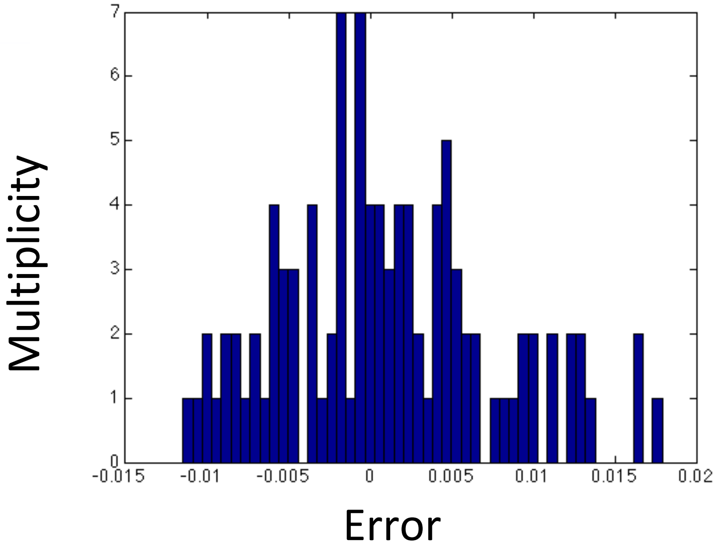
References

- Wedge Sampling: C. Seshadhri, A. Pinar and T. G. Kolda, ***Triadic Measures on Graphs: The Power of Wedge Sampling***, Proc. SIAM Intl. Conf. on Data Mining (SDM'13), Apr 2013 (preprint: arXiv:1202.5230).
- Wedge Sampling MapReduce: T. G. Kolda, A. Pinar, T. Plantenga, C. Seshadhri, and C. Task, ***Counting Triangles in Massive Graphs with MapReduce***, arXiv:1301.5887, to appear SIAM Scientific Computing.
- Wedge Sampling for directed patterns: C. Seshadhri, A. Pinar and T. G. Kolda, ***Wedge Sampling for Computing Clustering Coefficients and Triangle counts for Large Graphs,*** arXiv:1309.3321, Stat. Analysis & Data Mining, 7(4), pages: 294–307.
- Streaming Algorithm for triangles: M. Jha, C. Seshadhri, and A. Pinar, ***A Space Efficient Streaming Algorithm for Triangle Counting using the Birthday Paradox***," Proc. ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD'13) August 2013 (preprint; arXiv:1212.2264).
- Streaming Algorithm for multigraphs: M. Jha, A.Pinar, and C. Seshadhri, ***Counting Triangles in Real-World Graph Streams: Dealing with Repeated Edges and Time Windows***, arXiv:1310.7665.
- Counting 4-vertex patterns: M. Jha, A. Pinar, and C. Seshadhri, ***Path Sampling: A Fast and Provable Method for Estimating 4-Vertex Subgraph Counts***, submitted for conference publication.

Questions

Streaming algorithm is effective in practice

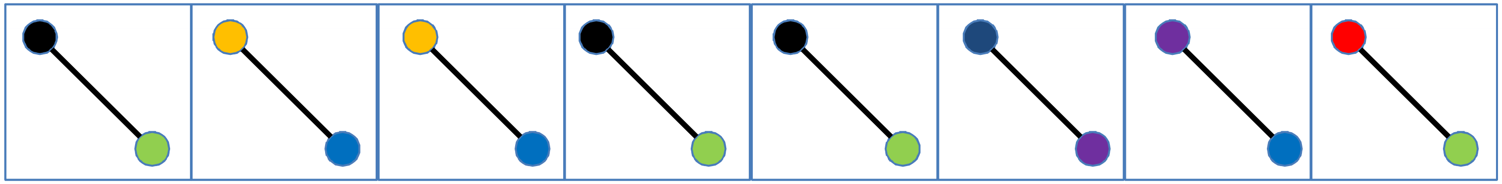




- Experiments on public data sets
- Edge pool size: 20K; Wedges pool size is: 20K
  - Pool sizes are independent of the graph size.
- The estimates are accurate.
- The variance is small.

Jha, Seshadhri, P. KDD 2013, Best Student Paper award

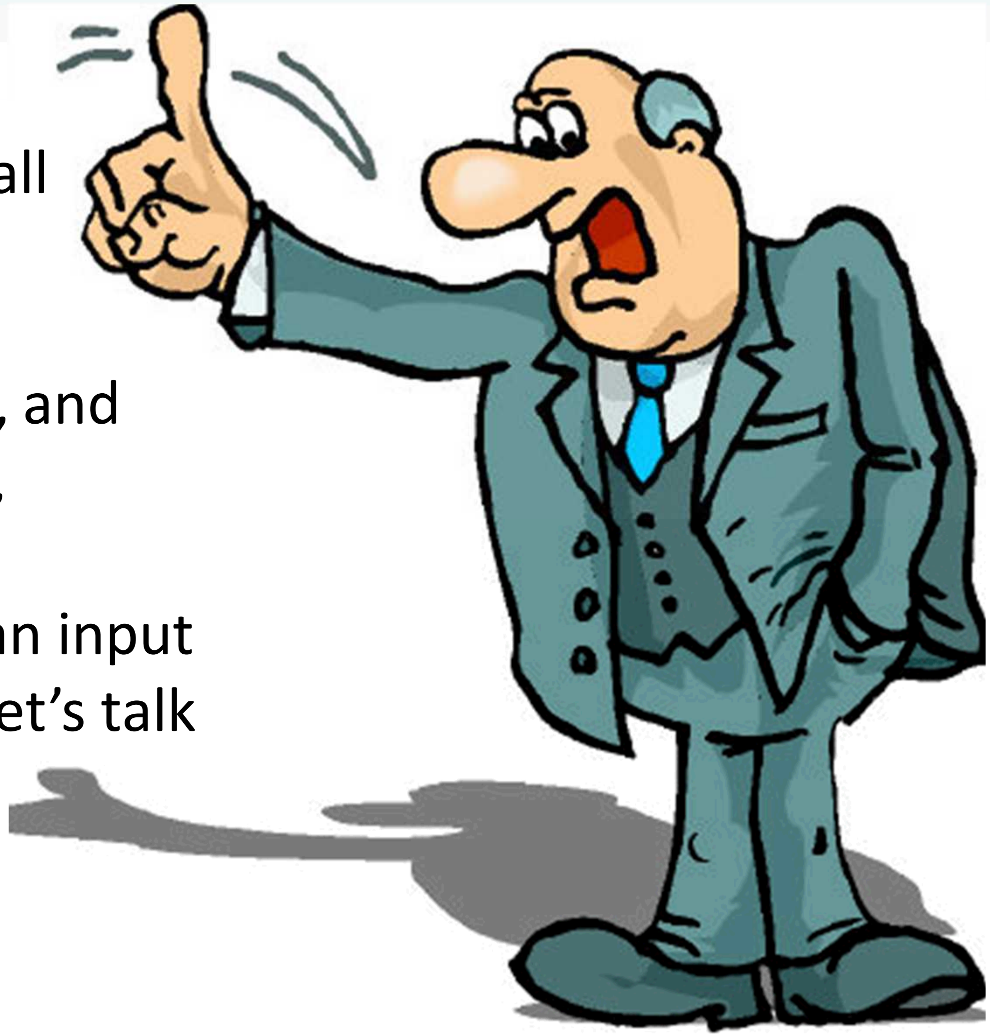
Drawbacks of ignoring repeats



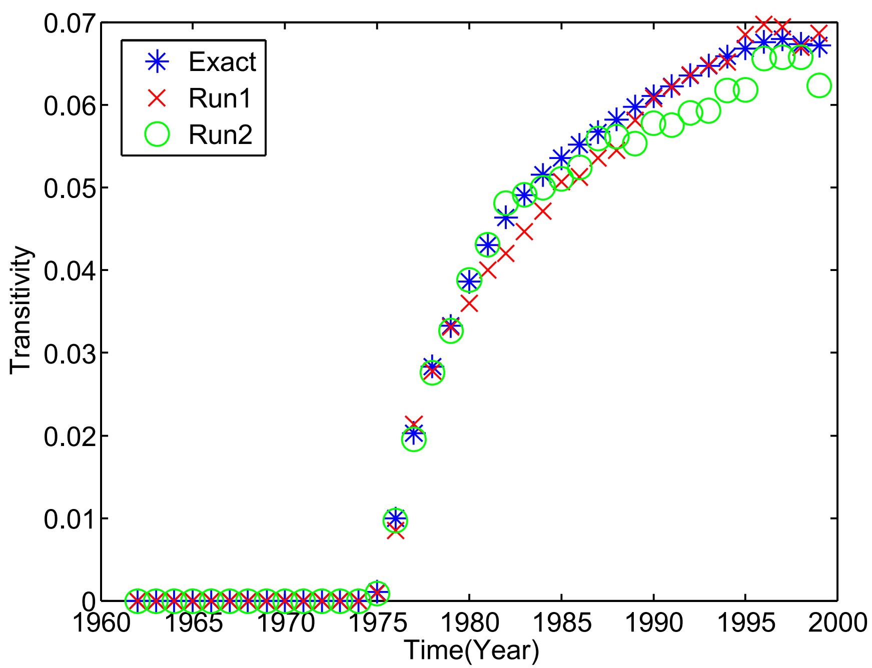
- Assumptions useful for algorithmic progress, but avoids real-world complexities
  - Algorithms cannot be deployed in "wild"
- Removing repeated edges requires extra pass over edges
  - Assumption of no repeats is expensive to enforce
- Not clear how to store information of various time-windows simultaneously

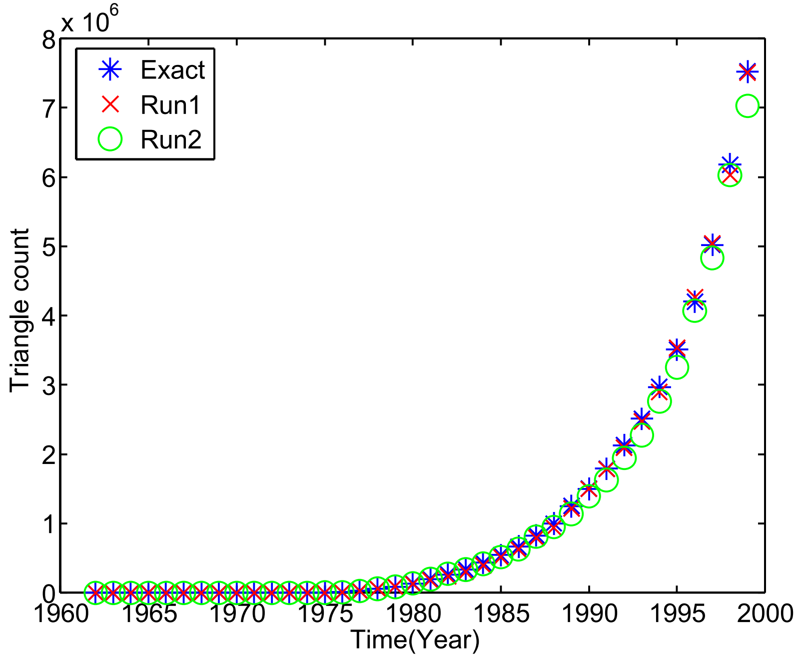
Take home lesson

- If you need the *counts* of small patterns on a large graph, use sampling.
- If you need a list of patterns, and
  - if the output size is small, enumerate.
  - If not, the list should be an input to another process, and let's talk about the full process.



Streaming algorithm provides a running estimate

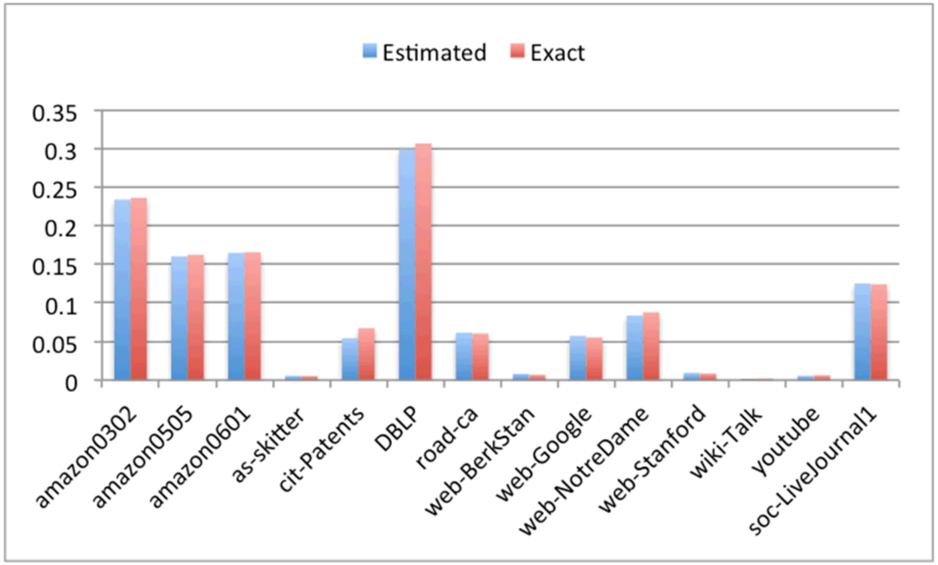




- Results on the patent citation network
  - 3.8M vertices, 16.5M edges.
- The algorithm provides accurate running estimates.

Next step: Streaming multigraphs

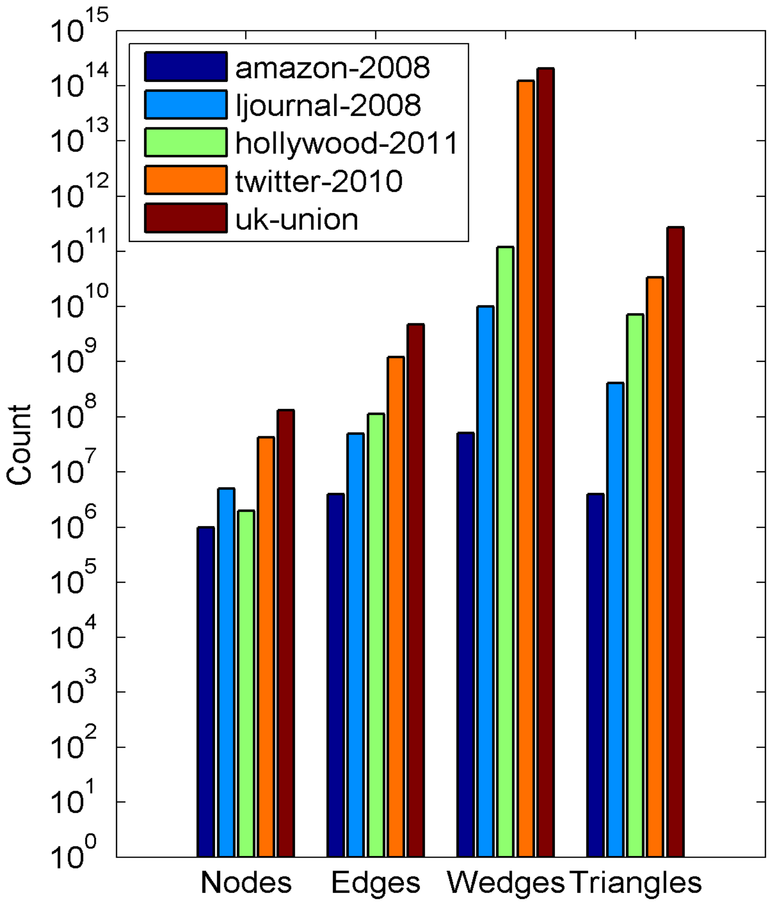
- Many graphs are a stream of *repeated* edges. (Emails, data transfers, co-authorship etc.)
- Generalized our algorithm for multigraphs.
  - Used random hashing to detect multiple instances.
  - Devised an unbiasing technique to avoid stream order sensitivity.
    - aaabbbccc vs. abcabcabc
- Processed the DBLP raw data
  - |V|=1.2M, |E|=5.1M,
    9.0 repeated edges,
    11.4M triangles transitivity= 0.174
  - Estimate with 30K edges
    and 30K wedges
    11.3M triangles transitivity= 0.173



Jha, Seshadhri, P., *arxiv 1310.7665*

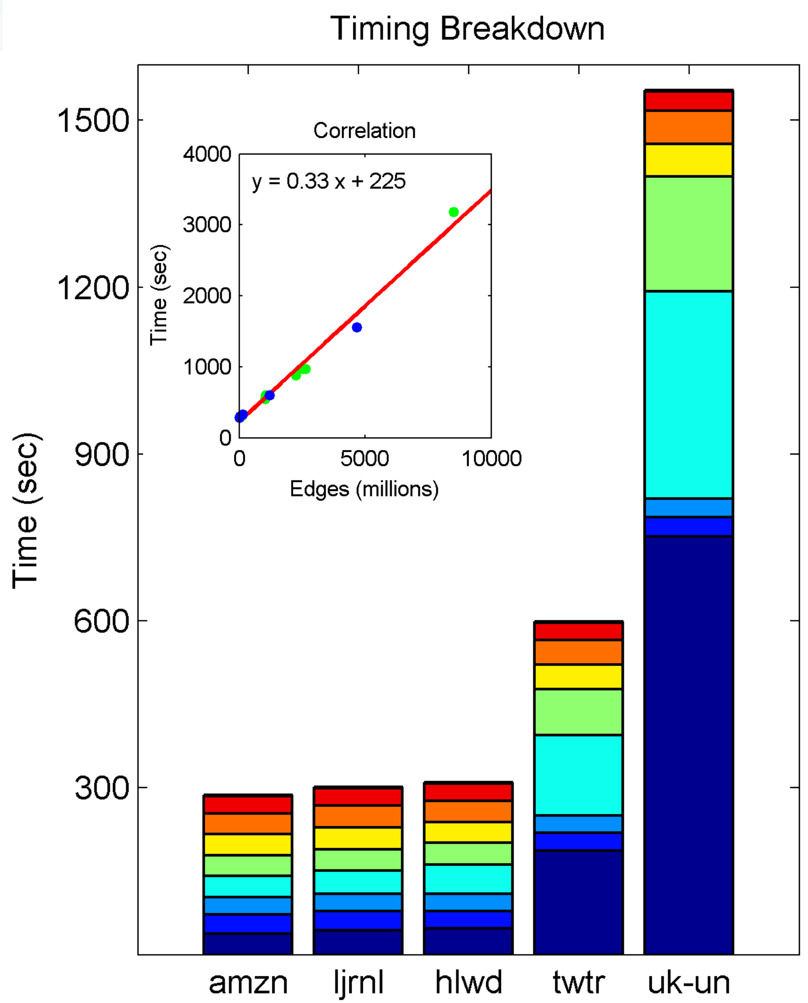
How about even bigger graphs?

- Wedge sampling can be executed when the data is distributed.
- We proposed a Hadoop implementation.
  - Key to success: data movement is minimal.
- 5 real-world networks
  - Source: Laboratory for Web Algorithms
  - Largest: 132M nodes, 4.6B edges
- Distributed Server: 32-Node Hadoop Cluster
  - 32 Intel 4-Core i7 930 2.8GHz CPU
  - 32 x 12GB = 384GB memory



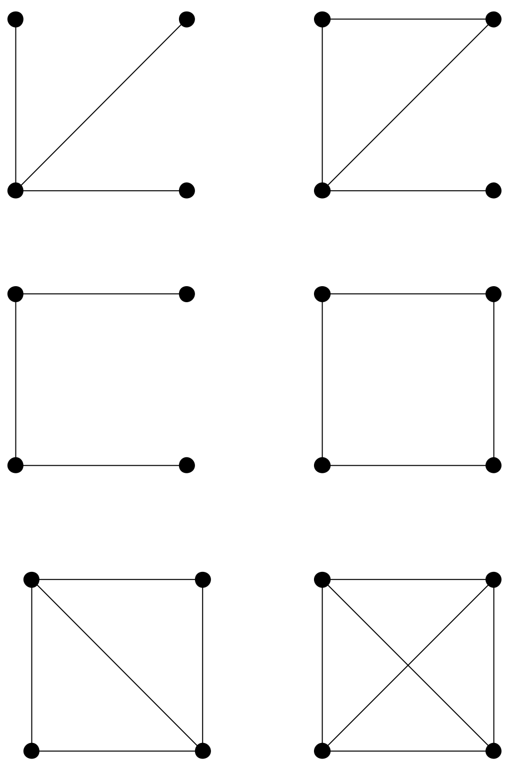
Wedge Sampling for BIG Graphs

- 32-node Hadoop cluster results using wedge sampling to compute degree wise clustering coefficients
  - Logarithmic bins; 2000 samples per bin
- Compare twitter times
  - Sampling: 10 mins on 32-node Hadoop cluster
  - Enumeration: 483 mins on 1636-node Hadoop cluster
    - Suri & Vassilvitskii, 2011
  - Enumeration: 180 mins on 32-core SGI, using 128GB RAM
    - by Jon Berry, 2013
- No comparisons for uk-union due to its size

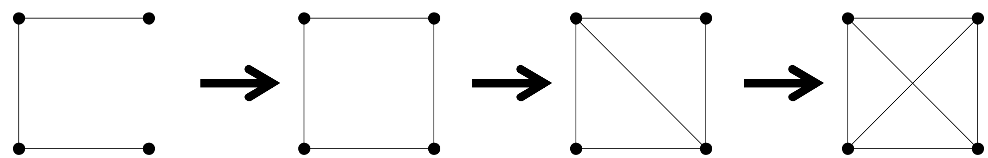


Kolda, P., Plantenga, Seshadhri, Task, arXiv:1301.5886, 2013 to appear in SISC

Counting 4-vertex patterns



- Our sampling approach can be generalized to count 4- vertex patterns.
- Algorithm
  - Count the number of 3-paths
  - Sample 3-paths and count how many of them other patterns

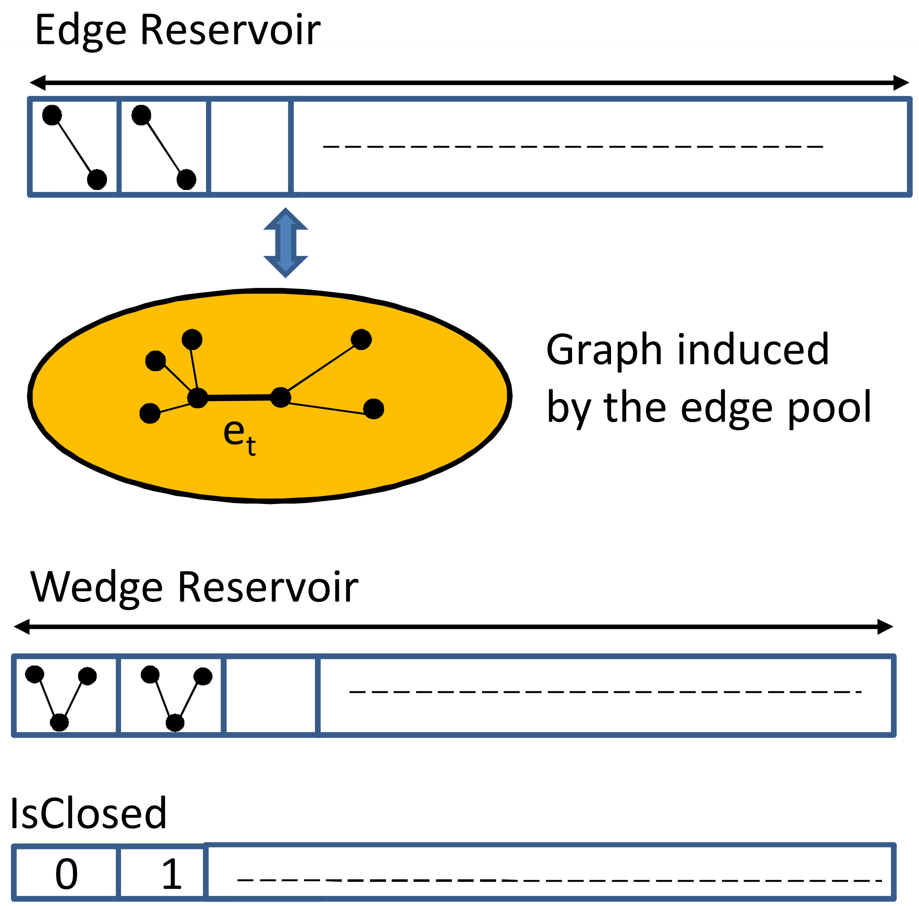


- Experiments show >1K speedups, with <%1 error using 160K samples.

Jha, Seshadhri, P., *coming soon*

Wedge sampling in a streaming world

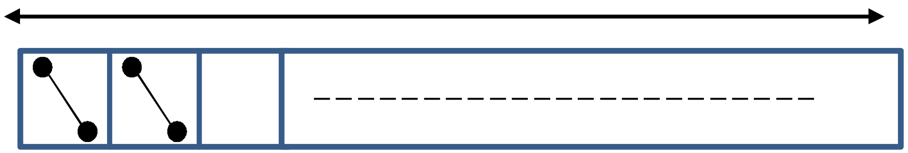
- Keep a random sample of the edges using the reservoir sampling.
- Keep a random sample of the wedges generated by the edges in the edge reservoir.
- Track whether the wedges are closed or not.
- The clustering coefficient is 3*ratio of closed wedges.



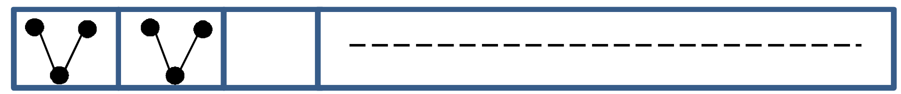
Jha, Seshadhri, P., *KDD* 2013, Best Student Paper award

Birthday paradox to the rescue

Edge reservoir



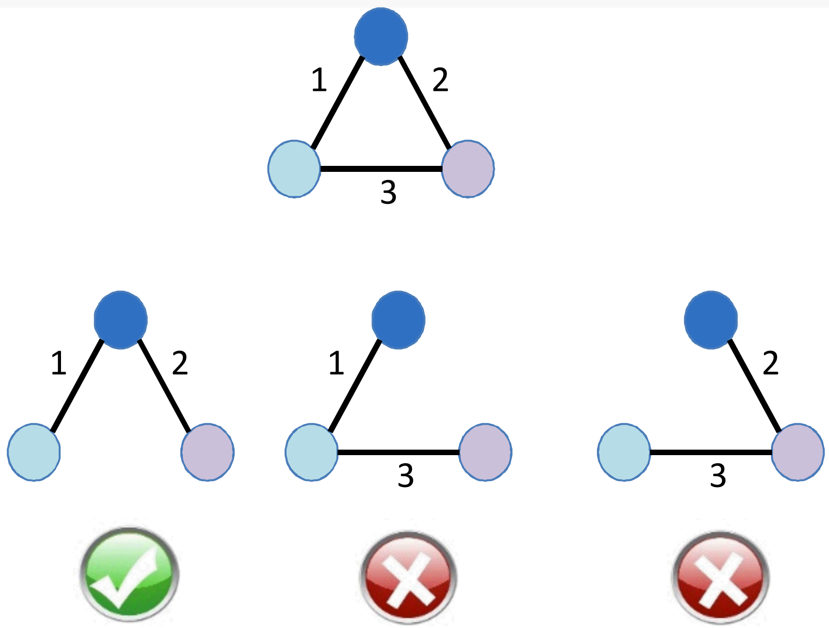
Wedge Reservoir



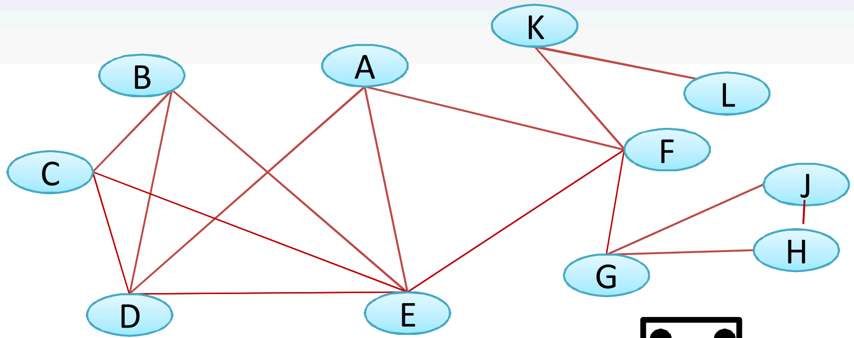
- A wedge is formed by two edge with the same birthday.
- Birthday paradox: O(√n) edges are sufficient to generate a wedge.
  - O(k√n) edges will produce $O(k^2)$wedges.
- Idealized algorithm: Maintain a separate edge reservoir for each wedge
  - Needs O(|S|√n)storage for |S| samples.
  - Has provable bounds; but not as effective in practice.
- Practical algorithm: Maintain a single and slightly bigger edge pool
  - Needs O(√(|S|n) storage
  - Wedge samples are biased, but in practice so enough wedges are generated to unbias the sample.
  - Effective in practice

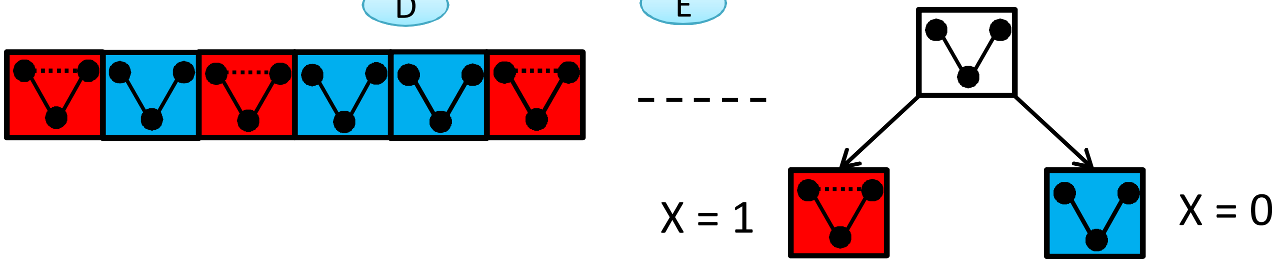
Making up for wedges closed by earlier edges

- Each triangle comprises of 3 wedges.
- In the original wedge sampling, we were able to detect any wedge as closed.
- In the streaming algorithm, we can only detect 1 of the 3 as closed.
- Since wedges are selected randomly, the expected closure rate is 3* the closure rate of the wedge pool.



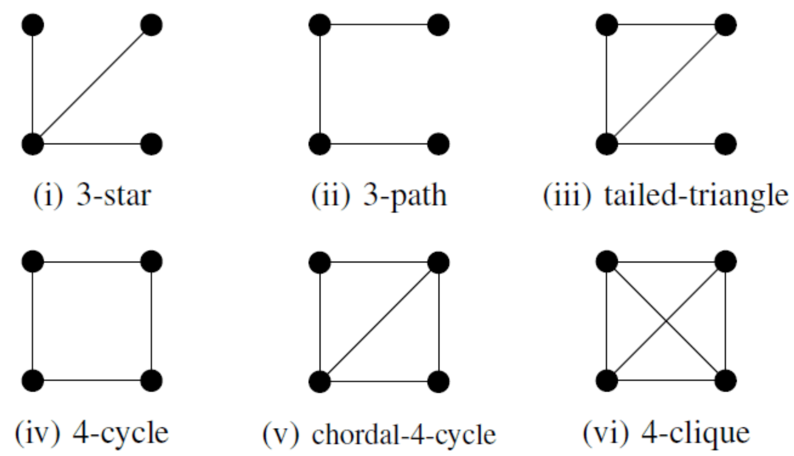
Wedge sampling for τ

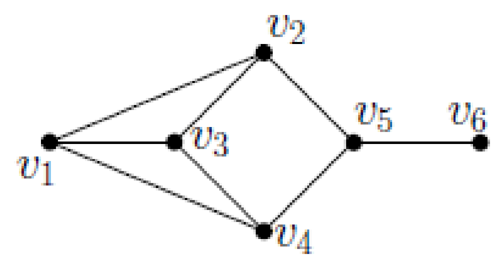




- C = 3T/W = fraction of closed wedges
- Consider list of all wedges, indexed with open/closed

Induced vs non-induced

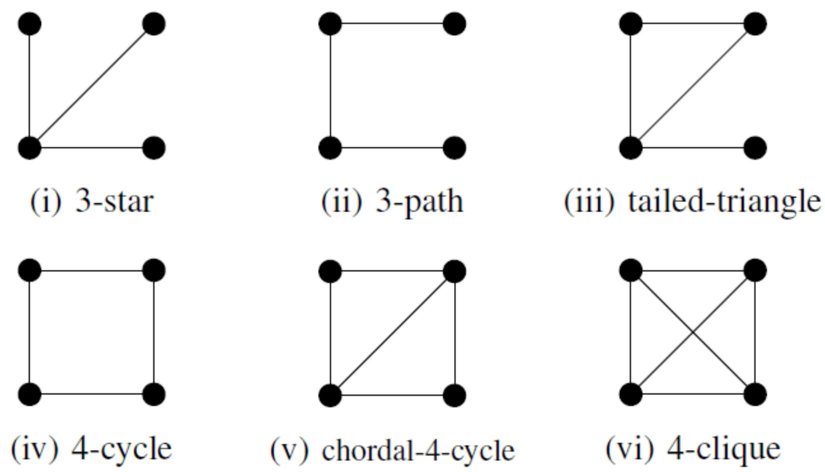




$$\begin{pmatrix} 1 & 0 & 1 & 0 & 2 & 4 \\ 0 & 1 & 2 & 4 & 6 & 12 \\ 0 & 0 & 1 & 0 & 4 & 12 \\ 0 & 0 & 0 & 1 & 1 & 3 \\ 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix} = \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{pmatrix}$$

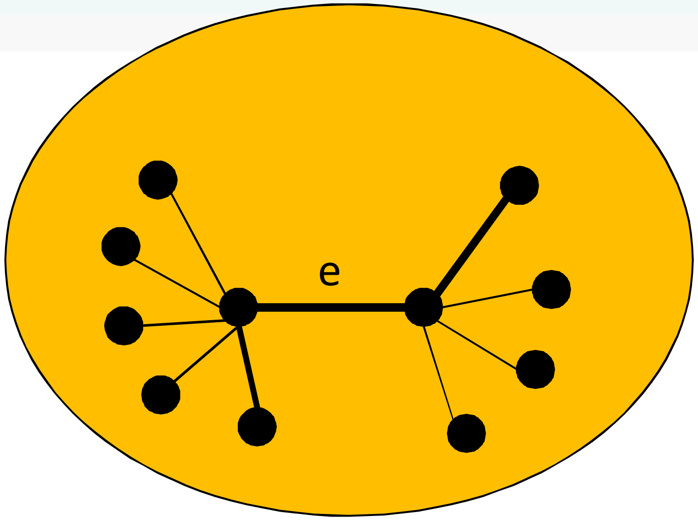
- (Vanilla) subgraph: take subset of edges
- Induced subgraph: take subset of vertices, take all edges in them
- Let C_i is induced count of pattern i
  - Getting vanilla counts not hard

Past art does not scale either



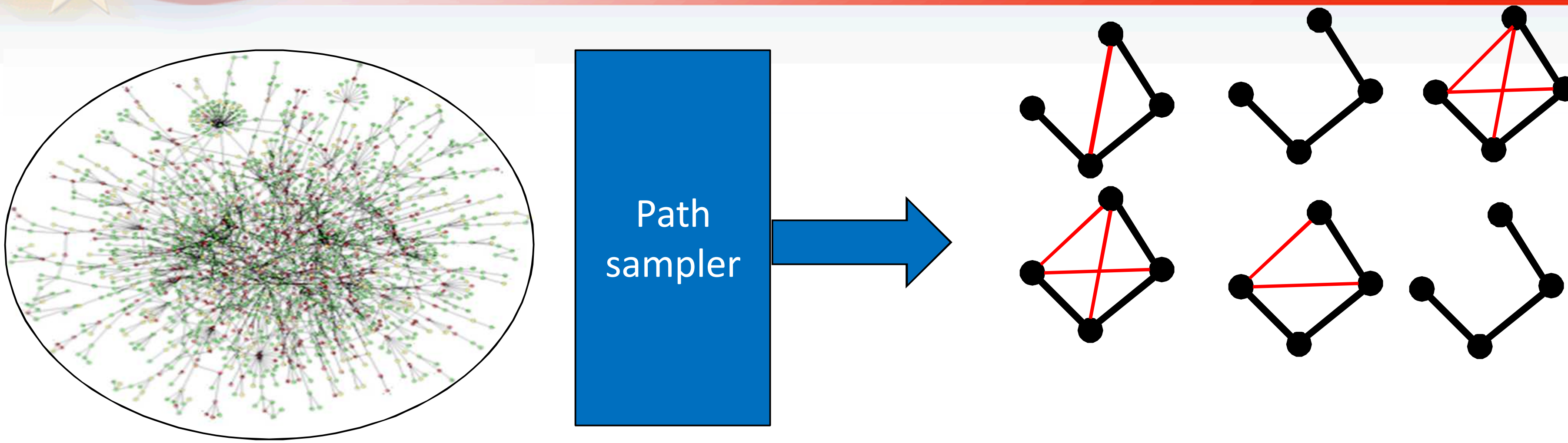
- MCMC methods, color coding, graph sparsification
- No provable methods, accuracies at best ~10%, often need computer clusters
- Nothing tailored for 4 vertices
- No results for (say) 100M edges

Sampling random 3-paths



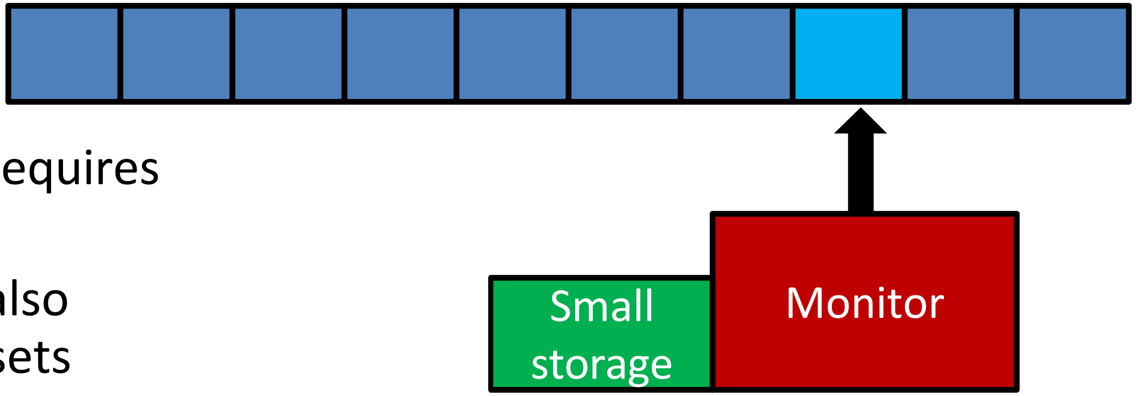
- First set for all edges, $W_{u,v} = (d_u - 1)(d_v - 1)$.
- Pick edge e = (u,v) with probability prop. to $W_{u,v}$
- Pick uniform random neighbor of u and of v
- If output is 3-path, guaranteed to be uniform random

The devilish details



- Works, but (provable) accuracy is not great
- Design methods to reduce samples
- Can give provable bounds: “for s samples, with 99.9% confidence, the true count is within 1% of answer”

What if we observe the data as a stream of edges?



- Many data analysis problems deal with data streams.
  - Situational awareness requires real time analysis.
- Streaming algorithms are also used to analyze large data sets with limited memory.
  - Multiple passes may be feasible.
- Algorithmically
  - We see each data point only once.
  - We either take action, or forever hold our peace.
- Not all problems are amenable to streaming analysis.
  - We cannot find needle in a haystack
  - But we can count frequent items, such as triangles