Counting Triangles and other Subgraphs in Data Streams

Stefano Leonardi[1]

Joint work with:
Luciana Salete Buriol[2], Gereon Frahling[3], Alberto Marchetti-Spaccamela[1], Christian Sohler[4]
[1] Univ. of Rome "La Sapienza"
[2] Univ. of Porto Alegre
[3]Google
[4] Heinz Nixdorf Institute, Univ. of Paderborn

Counting Subgraphs

Several applications:

- Network analysis: Computation of indices, e.g. the clustering coefficient
- Network modelling: Frequent small subgraphs or motifs are considered as building blocks of universal classes of complex networks [Itzkovits et al, Science 298]
- Community detection: Occurrence of a large number of specific subgraphs, e.g. bipartite cliques, has been observed in the Webgraph [Kumar et al, 1999]
- Indexing: identify the most frequent patterns in a graphical database [Yan, Yu and Han, 2004]

Most basic problem: Counting Triangles in a Graph

- Exact computation reduces to matrix multiplication: unfeasible for networks even of medium size
- Several heuristics have been proposed and tested (Schank and Wagner, 2005, Latapy 2006)
- Resort to the ***Data Stream Model:***
- *Data arrives one item at a time. The algorithms have the task of handling the computation in small space and computational time per item.*

Main applications:

- When the streams are not stored and must be processed on the fly as they are produced (more than 20 exabytes are created every year, most of them are forgotten);
- When the memory or time for storing or processing the stream is limited;
- When an exact computation is too time consuming and just a good estimation of the underlying data is required.

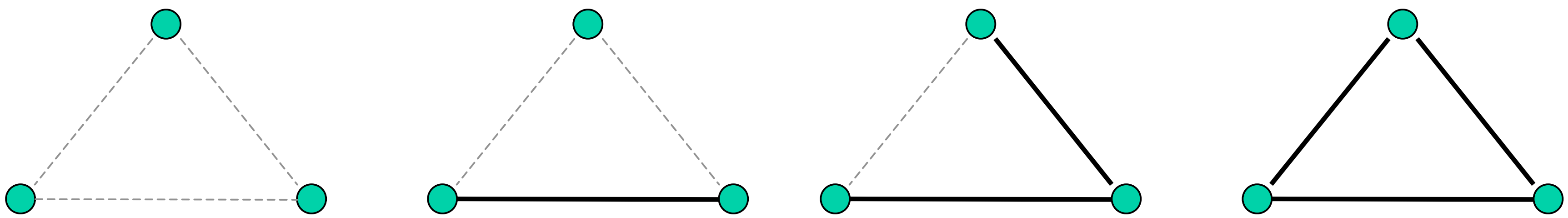
Data Stream Sampling Algorithms

- Selection of a subset of items and check some specific property on them;
- Define the kind of sample and the sample size
- Results: Algorithms that produce an $(1\pm\varepsilon)$ approximation of the number of subgraphs in the graph with probability at least $1-\delta$ by using $O(s)$ memory cells
- s is usually the number of samples needed to achieve a given precision

Counting Triangles in Data Streams

- Given a graph G=(V,E), where V is the set of vertices and E the set of edges, consider all triples of nodes of V;

- We can find four type of structures depending on the number of edges connecting them



Let's T0, T1, T2 and T3 represent the set of triples that have 0, 1, 2 and 3 edges, respectively.

Naive Sampling

- r independent samples of three distinct vertices (a,b,c) from the graph
- For the ith sample, if (a,b,c) is a triangle then output β_i=1 else output β_i=0.
- $E[\beta_i] = T_3 / (T_0 + T_1 + T_2 + T_3)$
- $T_3 = (T_0 + T_1 + T_2 + T_3) = (|V|*|V-1|*|V-2|) / 6$

Naive sampling

- Use $\Sigma_i \beta_i/r$ as an estimator of $E[\beta_i]$
- Output $T'_3 = T_3 * \Sigma_i \beta_i/r$
- By Chernoff bounds:
- If $r = O(\log(1/\delta)\, 1/\varepsilon^2\, ((T_0 + T_1 + T_2 + T_3) / T_3))$

  then $(1-\varepsilon)\, T_3 < T'_3 < T_3\, (1+\varepsilon)$ with pb $> 1-\delta$

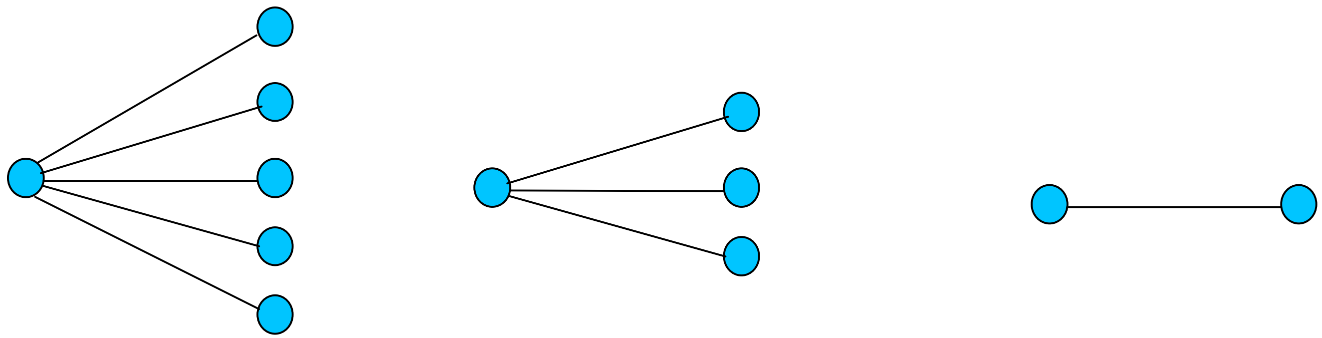
- Number of samples is prohibitive if $T_3 = o(n^2)$

The Graph as a Stream

- Adjancency Stream model: Each item of the stream is an arc of the graph

  Depending on the application, we can consider some order in the stream.

- Incidence Stream model: The entire incidence list of outgoing arcs of each node is extracted consecutively.



Our result for the Adjacency Stream model

Theorem 1: There exists a 1-pass streaming algorithm which needs $s=O(\log(1/\delta)\, 1/\varepsilon^2\, ((T1 + T2 + T3) / T3))$ memory cells and $O(1+ s \log |E|/|E|)$ update time per item

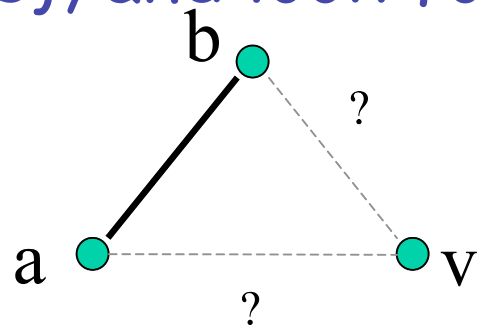
Previous best results:

$s=O(\log(1/\delta)\, 1/\varepsilon^2\, ((T_1 + T_2 + T_3)^3 / T_3) \log |V|)$

[Bar-Yossef, Kumar and Sivakumar, *Reductions in Streaming Algorithms, with an Application to Counting Triangles in Graphs*, SODA 2002]

Idea of the algorithm for the Adjacency Stream model

- We take an edge $e=(a,b) \in E$ and a node $v \in V \setminus \{a,b\}$, and look for the missing edges.



$|E|(|V|-2)$

- The following property holds for any graph:

$$T_1 + 2T_2 + 3T_3 = |E|(|V|-2)$$

- Triples belonging to T_0 are not considered.

A 3-pass streaming algorithm

1. 1st Pass: count the number of edges $|E|$ in the stream

2. 2nd Pass: sample an edge $e=(a,b)$ uniformly chosen among all edges from the stream.

   Choose a node v uniformly from $V \setminus \{a,b\}$

3. 3rd Pass: Test if edges (a,v) and (b,v) are present in the stream.

   If $(a,v) \in E$ and $(b,v) \in E$ then output $\beta=1$ else output $\beta=0$.

A 3-pass streaming algorithm

- The streaming algorithm outputs a value β having expected value:

$$E[\beta] = \frac{3T_3}{T_1 + 2T_2 + 3T_3}$$

- Furthermore:

$$T_3 = \frac{E[\beta].|E|(|V|-2)}{3}$$

A 3-pass streaming algorithm

- There is a streaming algorithm that outputs a value T'_3 satisfying $(1-\varepsilon)\, T < T' < T\,(1+\varepsilon)$ with probability $1-\delta$
- We start r parallel instances of the 3-pass algorithm, and each one outputs a value β_i

$$r = \frac{2}{\varepsilon^2} \frac{T_1 + 2T_2 + 3T_3}{T_3} \ln\left(\frac{1}{\delta}\right)$$

A 3-pass streaming algorithm

- We use $\frac{1}{r}\sum_{i=1}^{r}\beta_i$ as an estimator for

$$E[\beta] = \frac{3T_3}{T_1 + 2T_2 + 3T_3}$$

- We estimate T_3 as:

$$T'_3 = \left(\frac{1}{r}\sum_{i=1}^{r}\beta_i\right).\frac{|E|(|V|-2)}{3}$$

A 3-pass streaming algorithm

- Proof by Chernoff Bounds

$$\Pr\left[\frac{1}{r}\sum_{i=1}^{r}\beta_i \geq (1+\varepsilon)E[\beta]\right] \leq e^{-\varepsilon^2 .E[\beta].r/3}$$

$$\Pr\left[\frac{1}{r}\sum_{i=1}^{r}\beta_i \leq (1-\varepsilon)E[\beta]\right] \leq e^{-\varepsilon^2 .E[\beta].r/2}$$

- Setting

$$r = \frac{2}{\varepsilon^2}\frac{T_1 + 2T_2 + 3T_3}{T_3}\ln(\frac{1}{\delta})$$

both probabilities together are bounded by δ

A 3-pass streaming algorithm

- We suppose that the events within the brackets do not occur. In this case:

$$\frac{1}{r}\sum_{i=1}^{r}\beta_i < (1+\varepsilon)E[\beta]$$

$$\Rightarrow \frac{1}{r}\sum_{i=1}^{r}\beta_i \frac{|E|(|V|-2)}{3} < (1+\varepsilon)E[\beta]\frac{|E|(|V|-2)}{3}$$

$$\Rightarrow T'_3 < (1+\varepsilon)T_3$$

- Same argument to obtain $\Rightarrow T'_3 > (1+\varepsilon)T_3$

One pass algorithm

- A uniform choice of an edge in one pass can be done with reservoir sampling: choose the first edge as a sample edge and replacing this edge by the i-th edge of the stream with probability *1/i*.

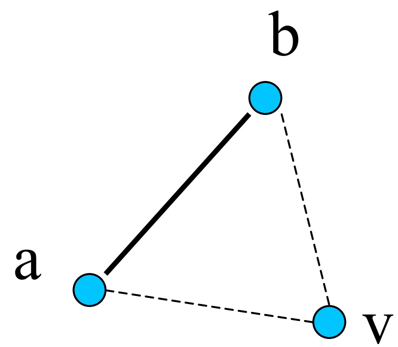
- When choosing a sample, it can happen that we already miss some arcs. We have 1/3 of probability of not doing that.

Sample one-pass

$i \leftarrow 1$;
for each edge $e_s=(a_s,b_s)$ in the stream do:

flip a coin. With probability $1/i$ do:
$a \leftarrow a_s$; $b \leftarrow b_s$;
$v \leftarrow$ node uniformly chosen from $V \setminus \{a,b\}$
$x \leftarrow$false; $y \leftarrow$false;
end do

if $e_s = (a,v)$ then $x \leftarrow$true;
If $e_s = (b,v)$ then $y \leftarrow$true;
end for
if x=true and y=true return $\beta=1$ else return $\beta=0$



Sample one-pass

- The streaming algorithm outputs a value b having expected value:

$$E[\beta] = \frac{\cancel{3}T_3}{T_1 + 2T_2 + 3T_3}$$

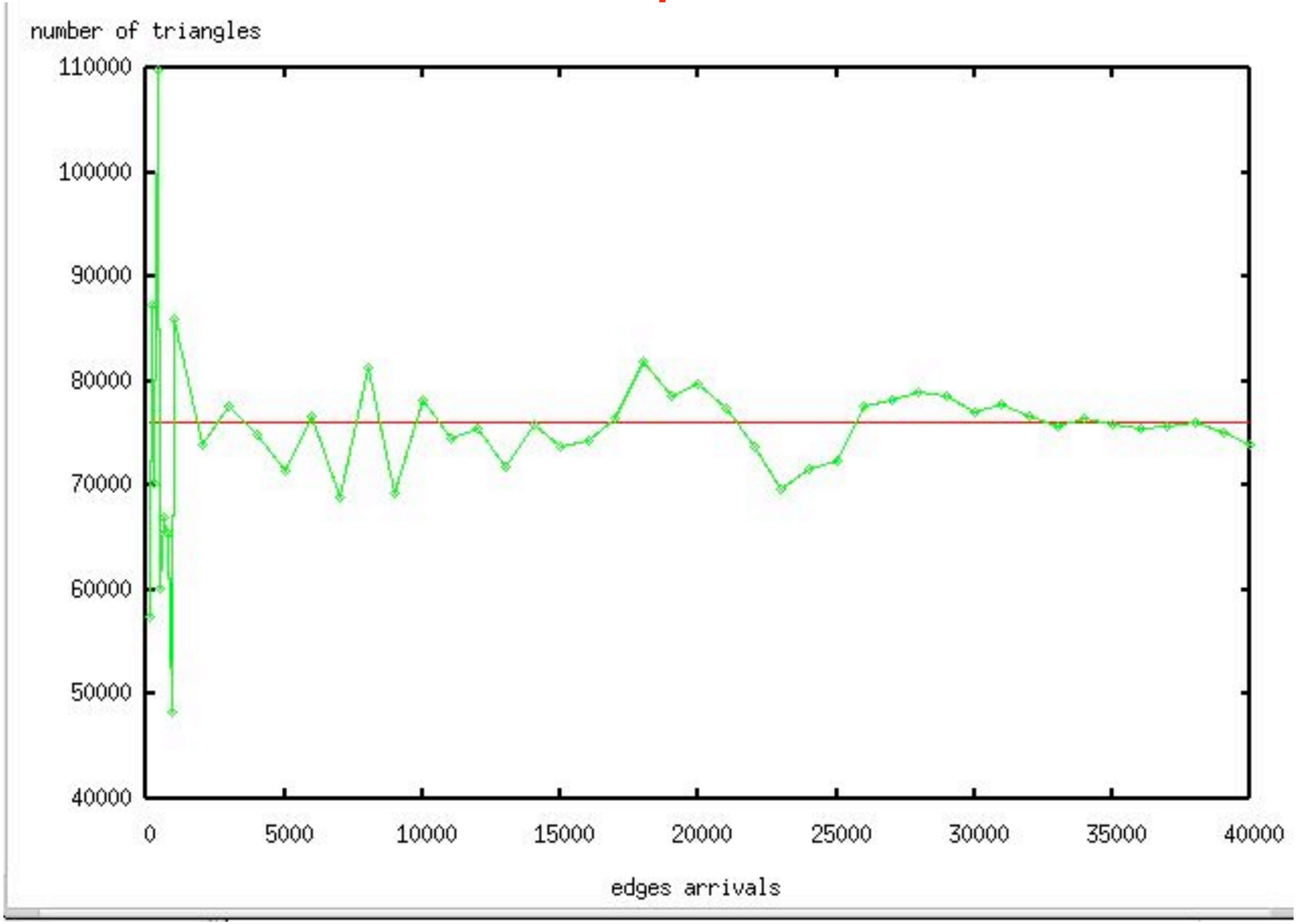
- The size of the sample

$$r = \frac{6}{\varepsilon^2} \frac{T_1 + 2T_2 + 3T_3}{T_3} \ln(\frac{1}{\delta})$$

- We estimate T_3 as:

$$T'_3 = \left(\frac{1}{r} \sum_{i=1}^{r} \beta_i \right) . |E| (|V| - 2)$$

Results for a sample set of size 100



Considering a structured stream

- Which kind of structure can benefit the algorithm and still be a natural and good representation of the graph?
- Consider the Incidence Stream model, where the adjacency lists of nodes are stored in sequence in the stream
- No order is required within each adjacency list
- Each arc is seen twice in the stream

Results on Incidence Stream

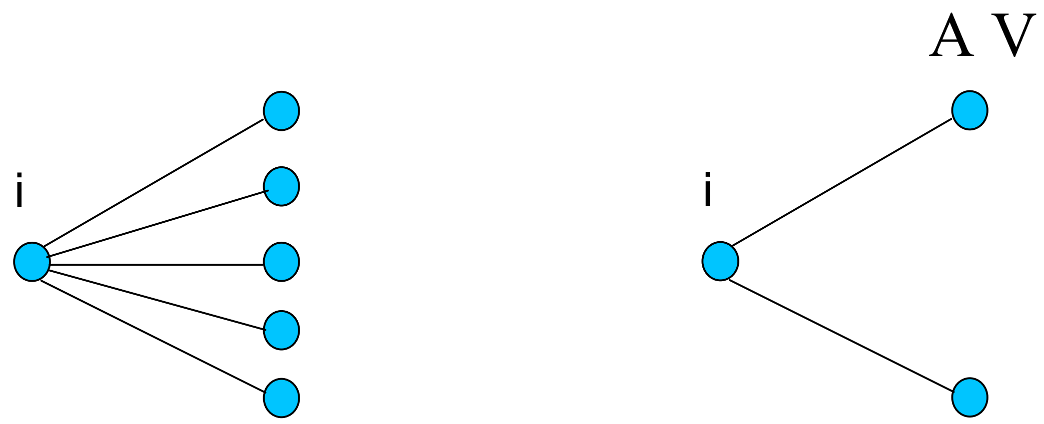
- Our result: $$O\left(\frac{1}{\varepsilon^2}.\log\left(\frac{1}{\delta}\right).\left(1+\frac{T_2}{T_3}\right)\right)$$

- Previous best results from Yossef, Kumar and Sivakumar: *Reductions in Streaming Algorithms, with an Application to Counting Triangles in Graphs*, 2002

$$O\left(\frac{1}{\varepsilon^2}.\log\left(\frac{1}{\delta}\right).\left(1+\frac{T_2}{T_3}\right)^2\log n + d\log n\right)$$

Incidence streams

- Sample from all possible Vs, i.e., combinations of two arcs leaving a node



- For each node *i*, where d_i is its degree, the number of V's, having node *i* in common is:

$$\binom{d_i}{2} = d_i \cdot \left(\frac{d_i - 1}{2} \right)$$

Counting triangles in incidence streams

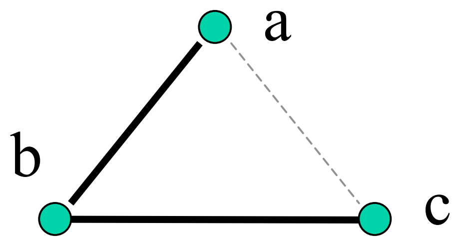
- In this case our sample is a V, and we check if the third arc is later seen in the stream

- It holds for any graph:

$$T_2 + 3T_3 = \sum_{i=1}^{|V|} d_i \cdot \binom{d_i - 1}{2}$$

Incidence 3-pass algorithm

- 1st Pass: count the number of Vs of the stream
- 2nd Pass: uniformly choose one V among all of them. Let us call it (a,b,c)



- 3rd Pass: Test if edge (a,c) is present in the stream. If $(a,c) \in E$ then output $\beta=1$ else output $\beta=0$;

Computational Experiments

- Optimized implementation of the algorithms
- Experiments on large Webgraphs, Wikigraphs, collaboration between scientists and actors
- Adjacency list model: accurate estimation for $s = 10^6$
- Incidence list model: accurate estimation for $s = 10^4$

Results for the Incidence List model

Table 2. Results for the one pass algorithm for counting triangles in an undirected graph structured as an incidence list. Samples of sizes of 10,000, 100,000 and 1,000,000 were considered.

Graph	r=10,000			r=100,000			r=1,000,000			$\frac{2T_3}{T_2+3T_3}$
	$\hat{T}_3$	Qlt (%)	Time	$\hat{T}_3$	Qlt (%)	Time	$\hat{T}_3$	Qlt (%)	Time	
webgraph	7,991,057,264	-	153.78	7,541,370,749	-	393.78	7,993,479,298	-	490.56	
	6,461,924,928	-	153.55	7,384,193,673	-	392.20	8,097,287,808	-	490.00	
	9,977,868,646	-	153.69	8,337,706,066	-	393.92	7,591,170,489	-	491.28	
actor2004	1,127,610,593	-4.16	12.29	1,155,564,261	-1.79	33.28	1,181,693,982	0.43	35.84	0.174932
	1,111,095,851	-5.57	12.52	1,192,599,566	1.36	20.28	1,177,782,402	0.10	35.11	
	1,177,449,181	0.07	12.12	1,175,270,762	-0.11	20.30	1,178,307,250	0.14	85.48	
google-2002	43,353	-1.22	0.28	45,489	3.65	1.20	44,765	2.00	4.97	0.004922
	45,293	3.20	0.28	45,435	3.52	1.00	43,704	-0.42	4.85	
	37,346	-14.91	0.27	42,420	-3.34	0.99	44,208	0.73	7.55	
actor2002	344,973,896	-0.53	6.70	345,817,151	-0.29	11.93	347,151,238	0.10	24.36	0.110693
	351,507,109	1.35	6.59	347,683,085	0.25	12.03	345,810,766	-0.29	24.38	
	330,775,554	-4.62	6.62	344,359,433	-0.71	12.00	347,532,178	0.21	55.16	
authors	1,636,611	-1.73	0.43	1,665,394	-0.01	1.21	1,670,148	0.28	4.47	0.227631
	1,586,971	-4.71	0.44	1,648,484	-1.02	1.19	1,665,792	0.02	4.45	
	1,633,188	-1.94	0.44	1,650,487	-0.90	1.20	1,664,291	-0.07	6.86	
itdk0304	458,517	0.76	0.33	449,558	-1.21	1.24	457,604	0.56	4.58	0.040506
	399,317	-12.25	0.34	458,260	0.70	1.11	451,481	-0.79	4.44	
	438,002	-3.75	0.34	453,440	-0.36	1.11	451,358	-0.81	6.40	
wikiEN	21,099,883	7.35	2.19	20,693,869	5.29	5.34	19,938,256	1.44	16.73	0.003876
	17,713,801	-9.87	2.21	20,206,714	2.81	4.78	19,894,603	1.22	16.78	
	20,695,192	5.30	2.19	17,977,501	-8.53	4.78	19,414,246	-1.22	26.72	
wikiDE	7,524,028	-6.87	0.91	8,265,424	2.31	3.24	8,120,882	0.52	10.54	0.027802
	8,327,148	3.07	0.89	8,213,376	1.66	2.44	8,080,158	0.01	10.54	
	8,114,584	0.44	0.94	8,162,754	1.04	2.45	8,024,967	-0.67	16.43	
wikiFR	3,060,821	-3.23	0.34	3,255,383	2.92	1.45	3,125,790	-1.18	7.67	0.038523
	3,476,882	9.92	0.34	3,199,530	1.15	1.29	3,125,613	-1.18	7.61	
	3,447,016	8.98	0.34	3,206,780	1.38	1.28	3,138,100	-0.79	10.63	
wikiES	863,765	8.45	0.18	782,798	-1.72	0.94	793,282	-0.40	5.09	0.042708
	791,437	-0.63	0.18	774,447	-2.76	0.90	800,619	0.52	5.09	
	768,999	-3.45	0.18	827,132	3.85	0.87	803,774	0.92	6.85	
wikiIT	339,404	3.39	0.12	313,241	-4.58	0.75	337,843	2.92	4.16	0.038986
	318,664	-2.92	0.12	308,480	-6.03	0.74	330,290	0.62	4.11	
	305,763	-6.85	0.12	339,498	3.42	0.73	322,894	-1.64	5.53	
wikiPT	70,699	0.94	0.07	70,443	0.57	0.53	70,942	1.28	2.63	0.026090
	62,620	-10.60	0.07	71,136	1.56	0.53	72,329	3.26	2.58	
	80,752	15.29	0.07	69,568	-0.68	0.53	69,203	-1.20	3.32	

Dimension of some graphs extracted from different sorces

Graph	$\|V\|$	$\|E\|_D$	$\|E\|_{ND}$	min	avg	max
actor2002	382,219	15,038,083	30,076,166	1	78.69	3,96
actor2004	667,609	27,581,275	55,162,550	1	82.63	4,605
authors	307,971	831,557	1,663,114	1	5.40	248
google-2002	394,510	480,259	960,518	1	2.43	1,160
itdk0304	192,244	609,066	1,218,132	1	6.34	1,071
wikiEN	339,834	4,811,393	9,622,786	0	28.32	47,123
wikiDE	116,251	1,907,891	3,815,782	0	32.82	5,962
wikiFR	42,987	577,781	1,155,562	0	26.88	7,651
wikiES	27,262	246,316	492,632	0	18.07	2,973
wikiIT	13,132	134,342	268,684	0	20.46	1,793
wikiPT	8,645	42,083	84,166	0	9.74	2,317
WebGraph						

Number of triangles of the graphs

Graph	#△	# Triples	transitivity	cc
actor2002	346,813,199	6,266,209,411	0.1660	0.78
actor2004	1,176,613,576	13,452,269,555	0.2624	0.80
authors	1,665,486	14,633,230	0.3414	0.76
google-2002	43,888	17,834,734	0.0074	0.23
itdk0304	455,062	22,468,727	0.0608	0.20
wikiEN	19,654,359	10,142,714,082	0.01	0.30
wikiDE	8,079,044	581,182,129	0.04	0.25
wikiFR	3,163,074	164,215,854	0.06	0.32
wikiES	796,465	37,298,489	0.06	0.31
wikiIT	328,265	16,840,168	0.06	0.33
wikiPT	70,043	5,369,380	0.04	0.46
WebGraph				

Comparing with the optimal computation [*Schank and Wagner, 2004*]

Graph	r=1000		r=10,000		r=100,000	
	Qlt.	Time	Qlt.	Time	Qlt.	Time
actor2002	9.3	5.89	3.60	7.59	0.93	17.88
actor2004	3.75	10.81	1.35	13.37	-0.56	28.94
authors	12.63	0.35	9.62	0.54	10.65	1.81
google-2002	60.79	0.20	28.52	0.37	26.5	1.71
itdk0304	16.00	0.26	9.18	0.43	9.31	1.81
wikiEN	71.55	1.88	1.30	2.60	0.08	7.15
wikiDE	22.13	0.77	-0.04	1.13	3.22	4.00
wikiFR	-3.59	0.24	1.76	0.45	1.52	2.19
wikiES	6.75	0.11	-0.36	0.26	1.00	1.50
wikiIT	25.14	0.07	3.91	0.20	3.80	1.29
wikiPT	-9.13	0.03	11.64	0.13	9.46	0.88
WebGraph	163,927,225	121.60	173,143,470	122.14	204,250,410	126.80

Clustering Coefficient

- Graph G = (V, E) V: set of n vertices E: set of m edges
- N(v) = set of vertices adjacent to v
- Local Clustering Coefficient of vertex

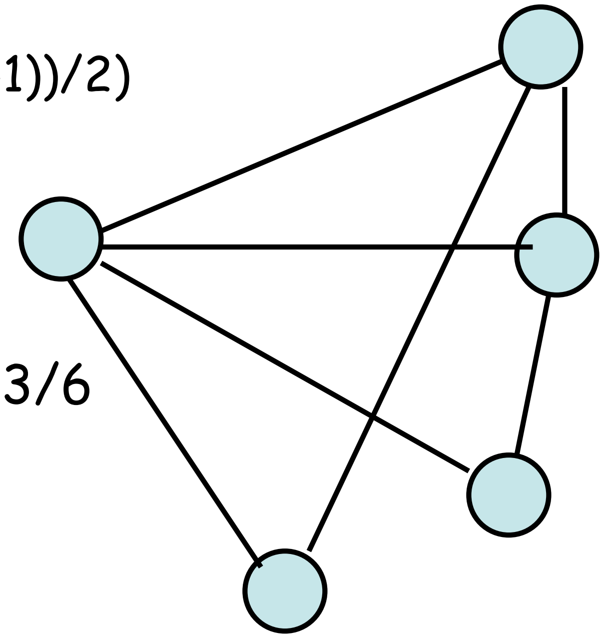
probability that any two vertices in N(v) are connected

$C(v) = |\{(u,v) \in E : u,v \in N(v)\}| / (N(v) * (N(v) - 1))/2)$

- Clustering Coefficient of a graph:

$C(G) = 1/n \sum C(v)$

V



C(v)=3/6

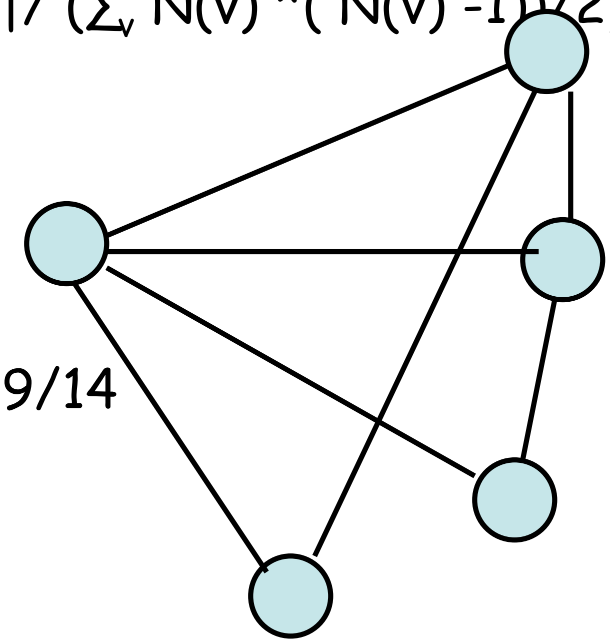
Transitivity Coefficient

- Transitivity Coefficient:

probability that any two vertices adjacent to a third vertex in the graph are connected

$$T(G) = \Sigma_v |\{(u,w) \in E : u,w \in N(v)\}| / (\Sigma_v N(v) *(N(v) -1))/2)$$

- Reduce to counting number of triangles in the graph



T(G)=9/14

Computing the Clustering Coefficient

- Our results:

  There is a 1-pass streaming algorithm which with pb $(1-\delta)$ returns an ε-approximation of $C(G)$ when the graph is given as an incidence stream that uses

  $$O(\log(1/\delta)\log n/\varepsilon^2 C(G)) \text{ memory cells.}$$

- $C(G)$ is usually in $[10^{-1},10^{-5}]$: feasible for networks of any size.

A 2-pass streaming algorithm

1. Sample s vertices $w_1, \ldots.., w_s$.

2. for i = 1 to s do

   sample at random pair (u,v), $u \neq v$, of points of $N(w_i)$

   If $(u,v) \in E$ then $X_i = 1$
   else $X_i = 0$

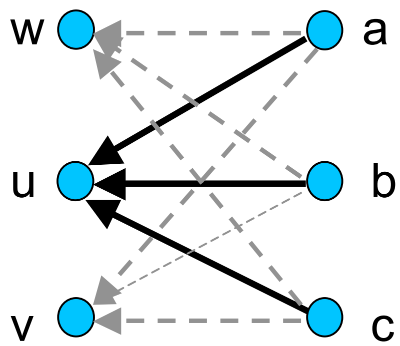
3. Output $X = 1/s \sum_i X_i$

Counting k3,3 in Data Streams

- Let k3,3 denote the number of k3,3 minors and k3,1 denote the number of k3,1 minors
- We assume the outdegree of the graph bounded by *d*
- The edges are sorted by destination nodes
- We do not assume any order by source nodes

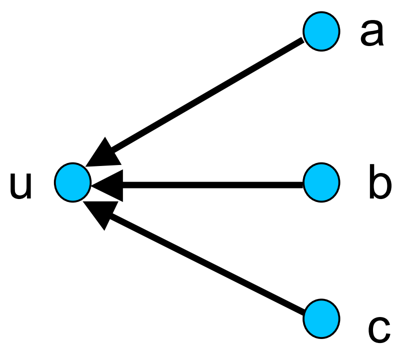
Sample

- Sample a k3,1 and 2 nodes not belonging to the k3,1



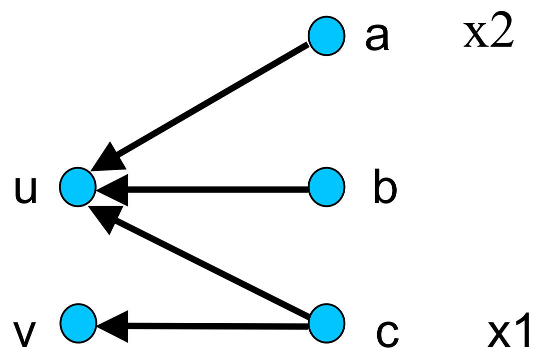
Counting k3,3 in Data Streams

- From all k3,1 occuring in the stream, chose one uniformly
- Let the three edges be (a,u), (b,u) and (c,u)



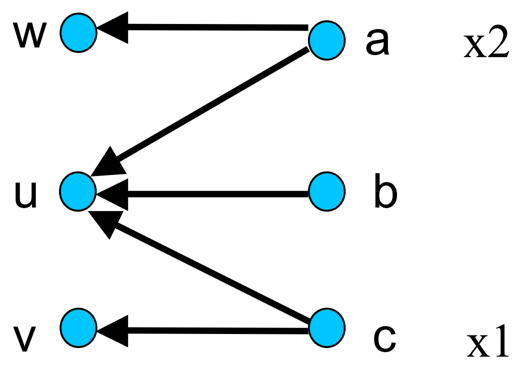
Counting k3,3 in Data Streams

- From all k3,1 occuring in the stream, chose one uniformly
- Let the three edges be (a,u), (b,u) and (c,u)
- Select uniformly x1, x2 $\in$ {a,b,c}
- Choose uniformly random variables k1, k2 $\in$ {1,2,…,d}
- If k1=k2 and x1=x2 then output $\beta = 0$
- Go on passing over the stream
- Select the (x1,v) as the k1-th edge (x1,) after selecting the k3,1



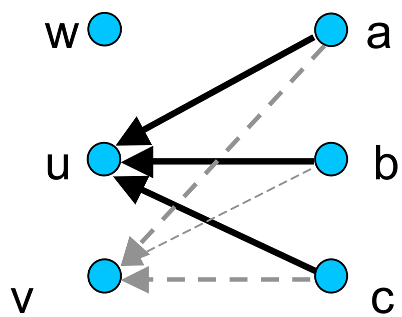
Counting k3,3 in Data Streams

- From all k3,1 occuring in the stream, chose one uniformly
- Let the three edges be (a,u), (b,u) and (c,u)
- Select uniformly x1, x2 $\in$ {a,b,c}
- Choose uniformly random variables k1, k2 $\in$ {1,2,…,d}
- If k1=k2 and x1=x2 then output β = 0
- Go on passing over the stream
- Select the (x1,v) as the k1-th edge (x1,) after selecting the k3,1
- Select the (x2,w) as the k2-th edge (x2,) after selecting the k3,1



Counting k3,3 in Data Streams

- From all k3,1 occuring in the stream, chose one uniformly
- Let the three edges be (a,u), (b,u) and (c,u)
- Select uniformly x1, x2 ∈ {a,b,c}
- Choose uniformly random variables k1, k2 ∈ {1,2,…,d}
- If k1=k2 and x1=x2 then output β = 0
- Go on passing over the stream
- Select the (x1,v) as the k1-th edge (x1,) after selecting the k3,1
- Select the (x2,w) as the k2-th edge (x2,) after selecting the k3,1
- From the time of selecting (x1,v): check if (a,v), (b,v), (c,v) are present in the stream



One-pass algorithm

- From all k3,1 occuring in the stream, chose one uniformly
- Let the three edges be (a,u), (b,u) and (c,u)
- Select uniformly x1, x2 $\in$ {a,b,c}
- Choose uniformly random variables k1, k2 $\in$ {1,2,…,d}
- If k1=k2 and x1=x2 then output $\beta = 0$
- Go on passing over the stream
- Select the (x1,v) as the k1-th edge (x1,) after selecting the k3,1
- Select the (x2,w) as the k2-th edge (x2,) after selecting the k3,1
- From the time of selecting (x1,v): check if (a,v), (b,v), (c,v) are present in the stream
- From the time of selecting (x2,w): check if (a,w), (b,w), (c,w) are present in the stream
- In this case output $\beta = 1$ else output $\beta = 0$

Probability of finding a k3,3

- The k3,3 will be chosen in case the following events occur:
  - Nodes a,b,c,u are chosen as the k3,1 with u being the destination node Pr = 1/k3,1
  - v and w must be chosen Pr = 1/d*1/d
  - x1 must be the first within the incidence list of v

    Pr = 1/3
  - x2 must be the first within the incidence list of w

    Pr = 1/3

Counting k3,3 in Data Streams

- The algorithm outputs a value β such that:

$$E[\beta] = \frac{k_{3,3}}{9d^2 k_{3,1}}$$

The following property holds for any graph:

$$k3,1 = \binom{d_i}{3} = \sum_{i=1}^{|V|} \frac{d_i(d_i-1)(d_i-2)}{6}$$

Counting k3,3 in Data Streams

- Number of samples:

$$r = \frac{1}{\varepsilon^2} . \frac{k_{3,1} . d^2}{k_{3,3}} . \ln \frac{1}{\delta}$$

- Approximation:

$$\widetilde{K}_{3,3} = \left(\frac{1}{r} \sum_{i=1}^{r} \beta_i \right) . \left(\sum_{i=1}^{|V|} d_i . (d_i - 1) . (d_i - 2) \right) . \left(\frac{9d^2}{6} \right)$$

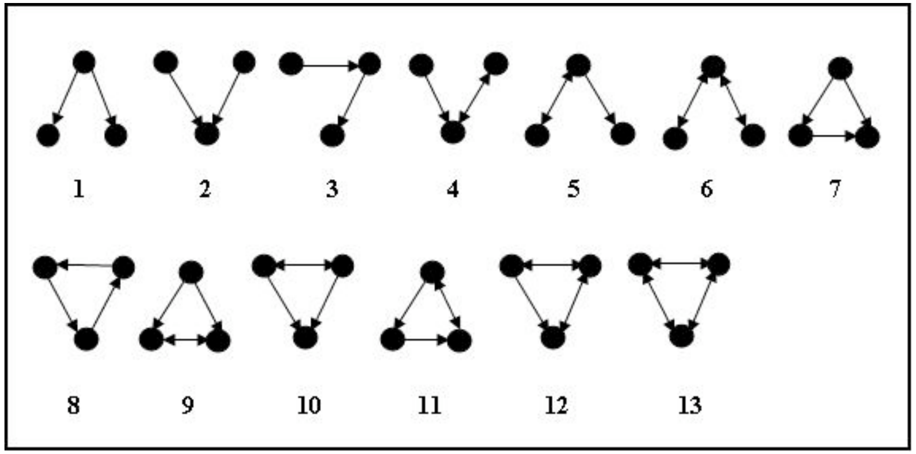
1-Pass algorithm for counting $K_{3,3}$

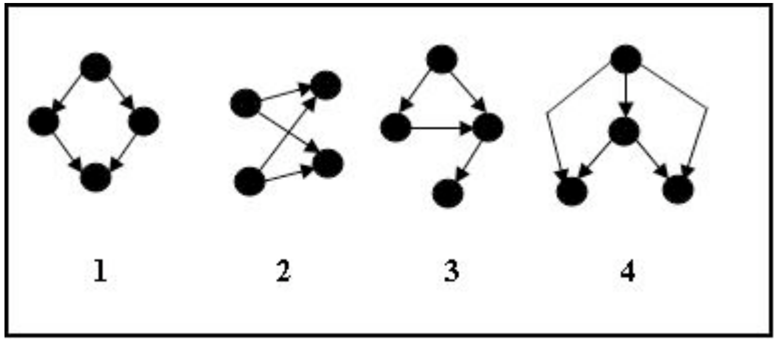
- There is a one pass algorithm that counts the number of k3,3 of a graph in incidence streams ordered by destination nodes with outdegree bounded by *d* up to a multiplicative error of ε with probability at least 1-δ, which space is

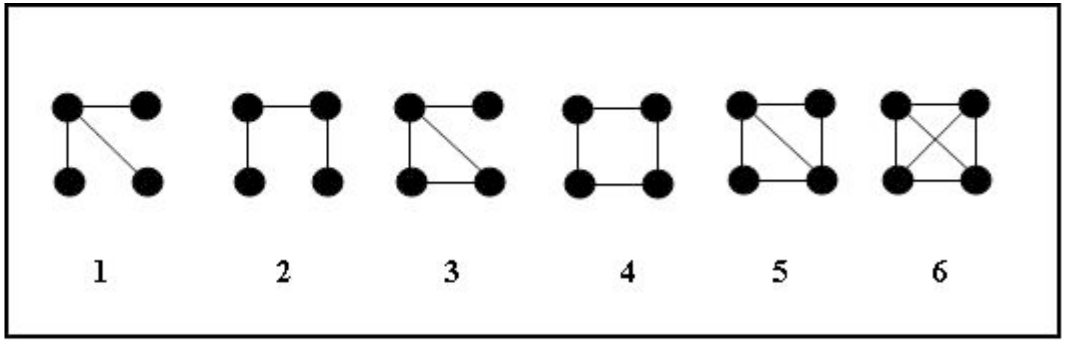
$$O\left(\log(|V|).\frac{1}{\varepsilon^2}.\frac{k_{3,1}.d^2}{k_{3,3}}.\ln\frac{1}{\delta}\right)$$

Counting other Subgraphs

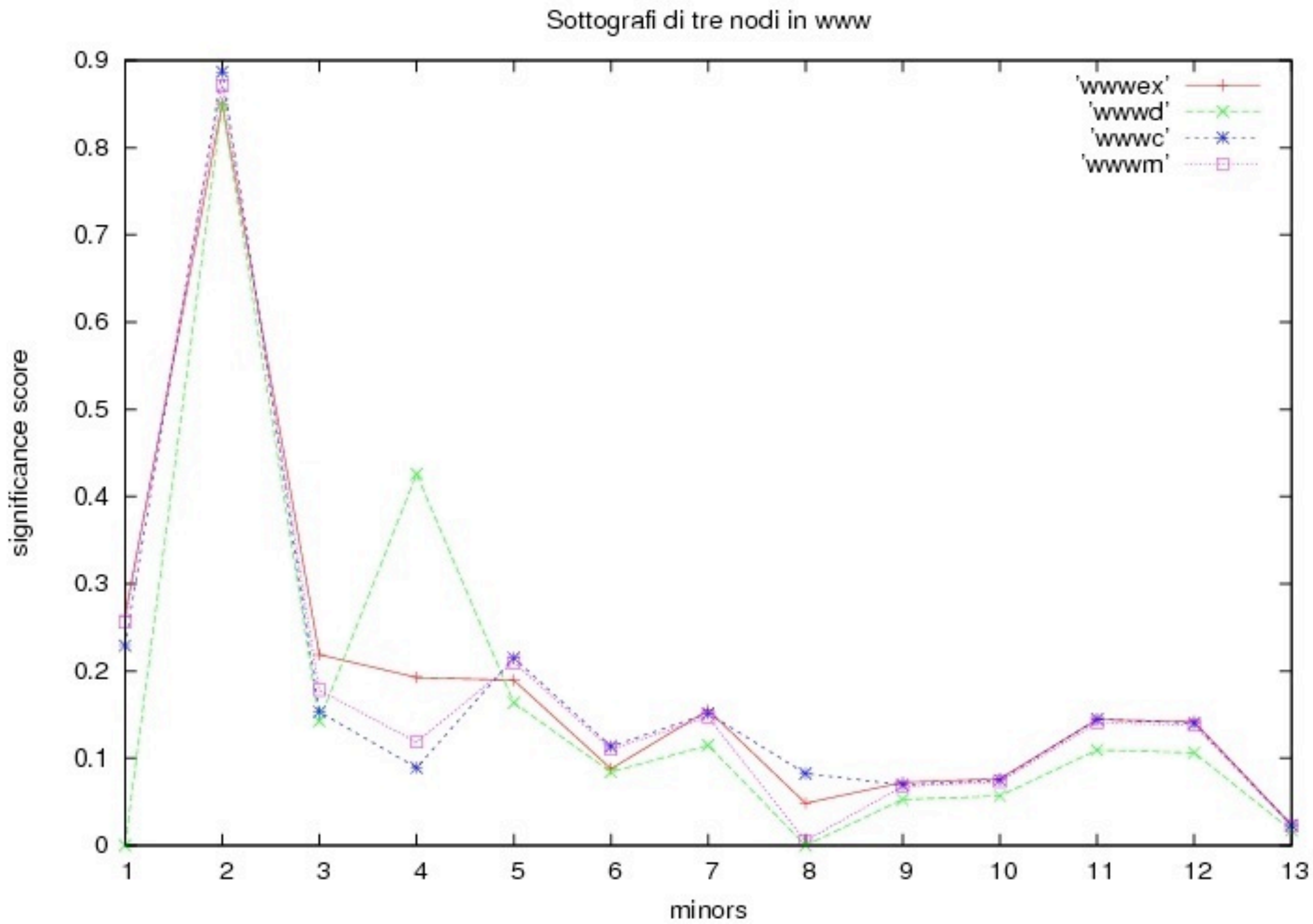
(with Ilaria Bordino and Debora Donato)



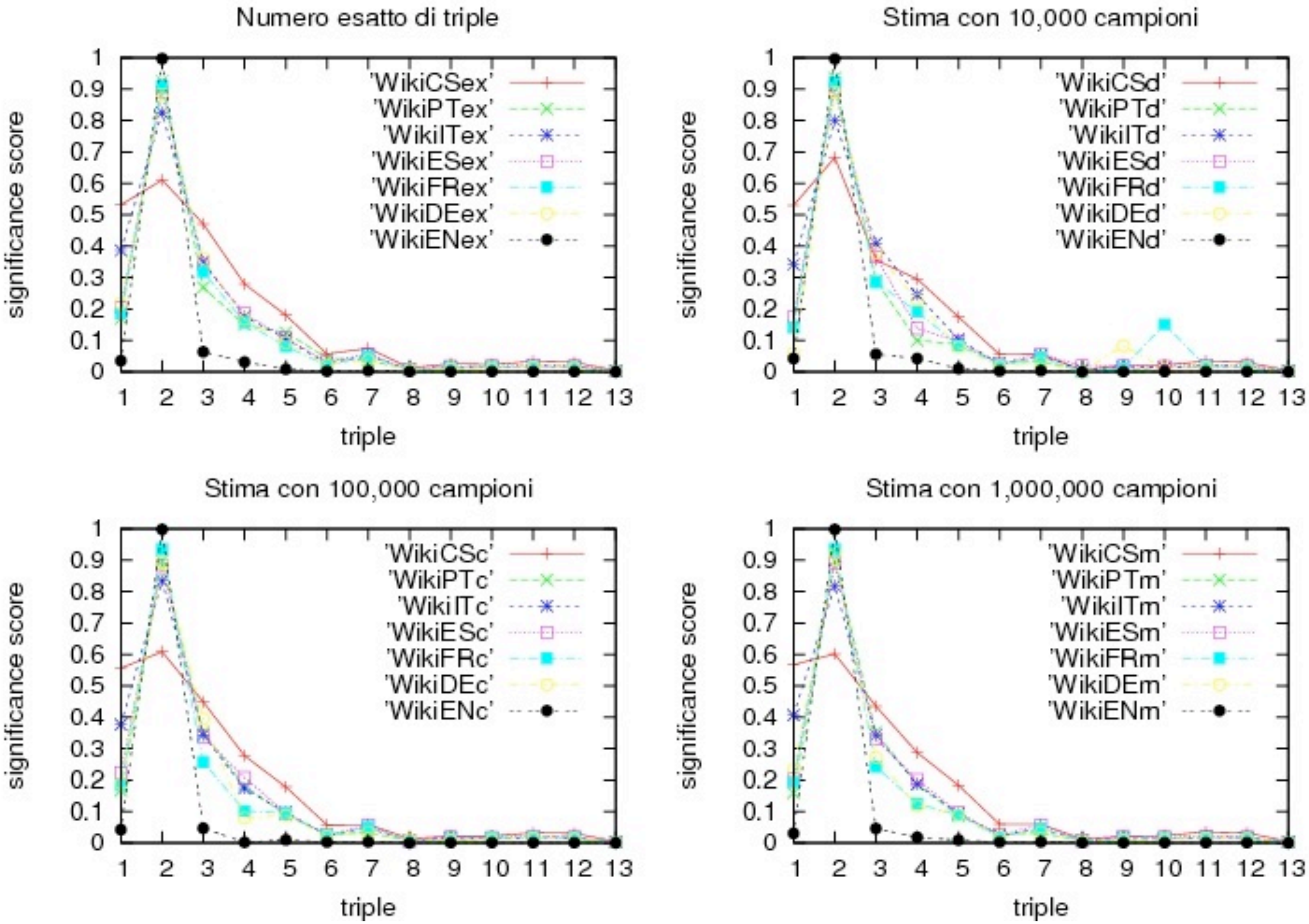




Experimental results



Experimental results



Conclusions and Open Problems

- Random Sampling Data Stream Algorithms for counting the number of some minors in a graph.
- Algorithms scale up to networks of any size for graph minors of size 3 and 4.
- Automatically select the best strategy for each given graph minor
- Counting on streams of insertions and deletions