Learning from Networks

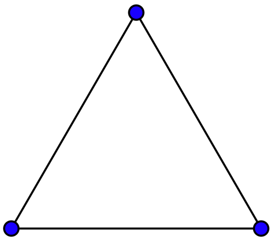
Graph Analytics: Clustering Coefficient

Fabio Vandin

October 23^{rd}, 2024

(Local) Clustering Coefficient

Another interesting feature for nodes/graphs: the number of triangles involving a node or in the graph.



Number of triangles:

- a node belongs to: *local clustering coefficient*
- in the graph: *clustering coefficient*

Idea: is a measure of the degree to which nodes in a neighborhood/graph tend to *cluster* together.

Local Clustering Coefficient

Let $\boxed{G = (V, E)}$ be a weighted/unweighted graph.

Given $v \in V$:

- let $\mathcal{N}(v)$ be the set of *neighbours* of v in G: $\boxed{\mathcal{N}(v) = \{u : (v, u) \in E\}}$.
- let $\boxed{deg(v) = |\mathcal{N}(v)|}$ be the degree of v

Definition

For a node $v \in V$, its $\boxed{\textit{local clustering coefficient } cc(v)}$ of v is:

$$\boxed{cc(v) = \frac{|\{(u_1, u_2) : u_1 \in \mathcal{N}(v), u_2 \in \mathcal{N}(v), (u_1, u_2) \in E\}|}{deg(v)\,(deg(v) - 1)}}$$

$\uparrow$
[0,1]

Note: $cc(v)$ is the fraction of triangles containing v among all the *potential* triangles containing v

Clustering Coefficient

Let $G = (V, E)$ be a weighted/unweighted graph with $|V| = n$.

Definition

The *clustering coefficient* $cc(G)$ of G is:

$$\boxed{cc(G) = \frac{|\{(u, v, z) : (u, v) \in E, (v, z) \in E, (z, u) \in E\}|}{6\binom{n}{3}}}$$

\# permutazioni di
$(u,v,z) \Rightarrow$ contate
tutte in numeratore

\# triangoli possibili

Clustering Coefficient

Sometimes the *average local clustering coefficient* is used as well:

Definition

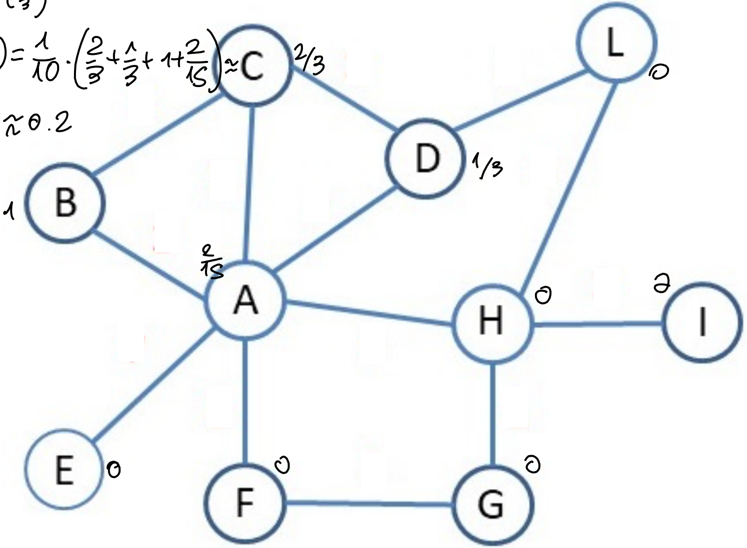
The *average local clustering coefficient* $avg_{lcc}(G)$ of G is:

$$\boxed{avg_{lcc}(G) = \frac{1}{n} \sum_{v \in V} cc(v)}$$

(Local) Clustering Coefficient: Example

$cc(G) = \frac{2 \cdot 6}{6\binom{10}{3}} \approx 0.017$

$avg_{loc}(G) = \frac{1}{10} \cdot \left(\frac{2}{3} + \frac{1}{3} + 1 + \frac{2}{15}\right) \approx 0.2$



Example: Real-World Networks have High Clustering Coefficient

Published: 04 June 1998

Collective dynamics of 'small-world' networks

Duncan J. Watts & Steven H. Strogatz

Nature **393**, 440–442 (1998) | Cite this article

Table 1 Empirical examples of small-world networks

	L_{actual}	L_{random}	C_{actual}	C_{random}
Film actors	3.65	2.99	0.79	0.00027
Power grid	18.7	12.4	0.080	0.005
C. elegans	2.65	2.25	0.28	0.05

Example: Clustering Coefficient of Networks from Different Domains

The structure and function of complex networks

M. E. J. Newman

Department of Physics, University of Michigan, Ann Arbor, MI 48109, U.S.A. and Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, U.S.A.

	network	type	n	m	$C^{(1)}$	$C^{(2)}$
social	film actors	undirected	449 913	25 516 482	0.20	0.78
	company directors	undirected	7 673	55 392	0.59	0.88
	math coauthorship	undirected	253 339	496 489	0.15	0.34
	physics coauthorship	undirected	52 909	245 300	0.45	0.56
	biology coauthorship	undirected	1 520 251	11 803 064	0.088	0.60
	telephone call graph	undirected	47 000 000	80 000 000		
	email messages	directed	59 912	86 300		0.16
	email address books	directed	16 881	57 029	0.17	0.13
	student relationships	undirected	573	477	0.005	0.001
	sexual contacts	undirected	2 810			
information	WWW `nd.edu`	directed	269 504	1 497 135	0.11	0.29
	WWW Altavista	directed	203 549 046	2 130 000 000		
	citation network	directed	783 339	6 716 198		
	Roget's Thesaurus	directed	1 022	5 103	0.13	0.15
	word co-occurrence	undirected	460 902	17 000 000		0.44
technological	Internet	undirected	10 697	31 992	0.035	0.39
	power grid	undirected	4 941	6 594	0.10	0.080
	train routes	undirected	587	19 603		0.69
	software packages	directed	1 439	1 723	0.070	0.082
	software classes	directed	1 377	2 213	0.033	0.012
	electronic circuits	undirected	24 097	53 248	0.010	0.030
	peer-to-peer network	undirected	880	1 296	0.012	0.011
biological	metabolic network	undirected	765	3 686	0.090	0.67
	protein interactions	undirected	2 115	2 240	0.072	0.071
	marine food web	directed	135	598	0.16	0.23
	freshwater food web	directed	92	997	0.20	0.087
	neural network	directed	307	2 359	0.18	0.28

Computing the Local Clustering Coefficient: One Node

Given $v \in V$, how can we compute its local clustering coefficient $cc(v)$?

Computing the Local Clustering Coefficient: One Node (continue)

Algorithm $\texttt{LCC}(G, v)$
Input: graph $G = (V, E)$ with $|V| = n$ and $|E| = m$; $v \in V$
Output: clustering coefficient cc_v of v

$num_t \leftarrow 0$;
forall $u_1 \in \mathcal{N}(v)$ **do**
    forall $u_2 \in \mathcal{N}(v)$ **do**
        if $u_1 \neq u_2$ *and* $(u_1, u_2) \in E$ **then**
            $num_t \leftarrow num_t + 1$;
$deg_v \leftarrow |\mathcal{N}(v)|$;
return $\frac{num_t}{deg_v(deg_v - 1)}$;

Complexity?

Computing the Local Clustering Coefficients

How to compute the clustering coefficient for all nodes in a network?

Algorithm `LCCs`(G)
Input: graph $G = (V, E)$ with $|V| = n$ and $|E| = m$
Output: clustering coefficient cc_v of v for each $v \in V$
forall $v \in V$ **do**
  $cc_v \leftarrow$ `LCC`(G, v);
return values cc_v;

Complexity?

Approximating Local Clustering Coefficients

We want to approximate the local clustering coefficient $cc(v)$ of all vertices $v \in V$.

We want efficient algorithms for large graphs: consider the *semi-streaming* model:

- G is stored on disk as a list of edges; in general: arbitrary order
- no random access is possible
- only sequential access is allowed
- the memory size is much smaller than the size of G

Goal: limited number of sequential scans of the data stored in secondary memory

Approximating Local Clustering Coefficients

Assume we have:

- graph $G = (V, E)$ with $n = |V|$, $m = |E|$
- memory:
  - M edges, with $M << m$;
  - n space for the vertices (e.g., a constant number of counters for each vertex)

Goal: approximate the local clustering coefficient $cc(v)$ of all vertices $v \in V$ with few (one?) scans of the data

A First Sampling Approach

Idea:

- build a sample S by sampling each edge with probability p
- count the number of triangles t_v^S in S incident to each vertex $v \in V$

How should we fix the probability p?
If we fix $p = \frac{M}{m}$, then at the end we will sample $\approx M$ edges

How many passes are needed?
Only 1 pass (to compute S)

What about the degree?
Can be computed in the same pass!

A First Sampling Algorithm

Algorithm `EstimateLCC`(G, M)

Input: stream Σ of edges for graph $G = (V, E)$ with $|V| = n$ and $|E| = m$; edge memory size M

Output: estimate cc_v of clustering coefficient of $v, \forall v \in V$

$S \leftarrow \emptyset$; $p \leftarrow M/m$;
$t_v^S \leftarrow 0$ for all $v \in V$; $deg_v \leftarrow 0$ for all $v \in V$;
foreach *edge* (u, v) *from* Σ **do**
    $deg_v \leftarrow deg_v + 1$; $deg_u \leftarrow deg_u + 1$;
    if *SampleElement*$((u,v),p)$ **then**
        $S \leftarrow S \cup \{(u, v)\}$;
        $\mathcal{N}_{u,v}^S \leftarrow \mathcal{N}^S(u) \cap \mathcal{N}^S(v)$;
        forall $c \in \mathcal{N}_{u,v}^S$ **do**
            $t_v^S \leftarrow t_v^S + 1$;
            $t_u^S \leftarrow t_u^S + 1$;
            $t_c^S \leftarrow t_c^S + 1$;
$cc_v \leftarrow 2t_v^S/(deg_v(deg_v - 1))$ for all $v \in V$;
return cc_v for all $v \in V$;

A First Sampling Algorithm

Notes:

- `SampleElement(`$(u, v), p$`)` returns `True` with probability p, and `False` with probability $1 - p$
- $\mathcal{N}^S(u)$ = neighbours of u considering only edges in the sample S

Anything missing?

Analysis

What is the expectation of cc_v?

A Simple Improvement

Algorithm `ImprovedEstimateLCC`(G, M)

Input: stream Σ of edges for graph $G = (V, E)$ with $|V| = n$ and $|E| = m$; edge memory size M

Output: estimate cc_v of clustering coefficient of $v, \forall v \in V$

$S \leftarrow \emptyset$; $p \leftarrow M/m$;

$t_v^S \leftarrow 0$ for all $v \in V$; $deg_v \leftarrow 0$ for all $v \in V$;

foreach *edge* (u, v) *from* Σ **do**

- $deg_v \leftarrow deg_v + 1$; $deg_u \leftarrow deg_u + 1$;
- $\mathcal{N}_{u,v}^S \leftarrow \mathcal{N}^S(u) \cap \mathcal{N}^S(v)$;
- **forall** $c \in \mathcal{N}_{u,v}^S$ **do**
  - $t_v^S \leftarrow t_v^S + 1$;
  - $t_u^S \leftarrow t_u^S + 1$;
  - $t_c^S \leftarrow t_c^S + 1$;
- **if** `SampleElement`$((u,v),p)$ **then**
  - $S \leftarrow S \cup \{(u, v)\}$;

$cc_v \leftarrow 2t_v^S/(deg_v(deg_v - 1))$ for all $v \in V$;

return cc_v for all $v \in V$;

A Simple Improvement

Notes:

- `SampleElement((`u`,`v`),`p`)` returns `True` with probability p, and `False` with probability $1 - p$
- $\mathcal{N}^S(u)$ = neighbours of u considering only edges in the sample S

Anything missing?

Analysis

What is the expectation of cc_v?

Analysis: Space

How many edges are stored by the two algorithms above?

Note that $|S|$ is r.v.... Distribution?

$|S| \sim Bin(m, p) = Bin(m, M/m)$

$\mathbb{E}[|S|] = M$

But this is only in expectation!

What if we need to make sure that we do not store more than M edges?

Moreover, the memory is filled only at the end of the algorithm: in some sense we are wasting space during the execution.

How can we fix the problems above?

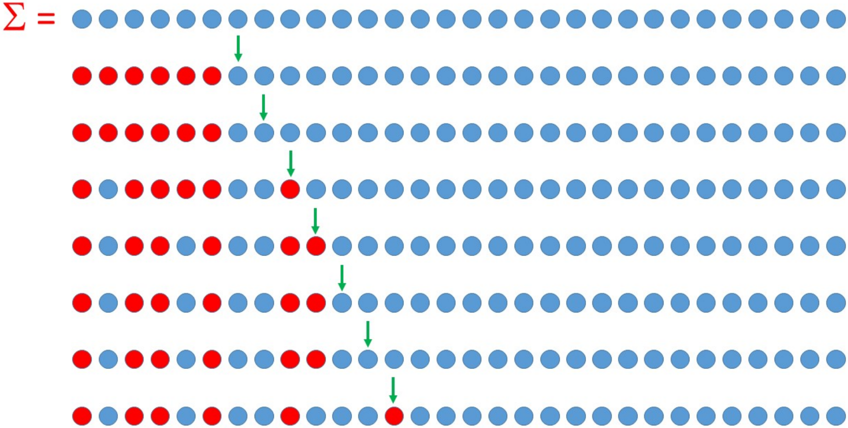
Reservoir Sampling

Algorithm `ReservoirSampling`(Σ, M)
Input: stream Σ of elements; sample size M
Output: at each step, S is a random sample of all elements seen up to that step in the stream

$S \leftarrow \emptyset$; $t \leftarrow 0$;
foreach x *from* Σ **do**
    $t \leftarrow t + 1$;
    if $t \leq M$ **then** $S \leftarrow S \cup \{x\}$;
    else
        if *SampleElement*$(x, M/t)$ **then**
            remove element from S chosen uniformly at random;
            $S \leftarrow S \cup \{x\}$;

Note: the knowledge of the stream size is not required!

Example



and so on ....

Analysis

Proposition

Let $\Sigma = x_1, x_2, \ldots$. For any step $t \geq M$:

i) $|S| = M$
ii) for each $x \in x_1, x_2, \ldots, x_t$, we have $\Pr[x \in S] = M/t$

Final Algorithm

Combines the simple improved algorithm with the reservoir sampling

Exercise

Write the pseudocode of the final algorithm.

Analysis

- $\mathbb{E}[cc_v] = cc(v)\ \forall v \in V$
- what about concentration results?

  Difficult! The appearance of triangles in the sample S are not independent!

  See De Stefani et al. (2017) and Lee et al. (2020) for some results.

Experimental Evaluation

The VLDB Journal (2020) 29:1501–1525
https://doi.org/10.1007/s00778-020-00624-7

REGULAR PAPER

Check for updates

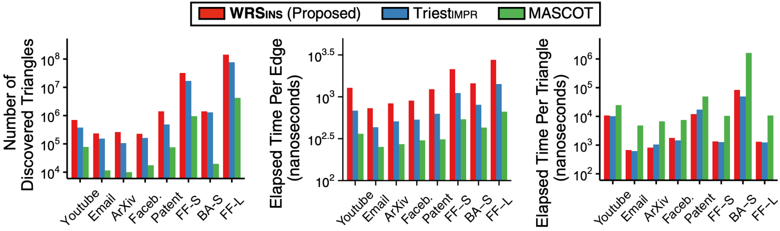
Temporal locality-aware sampling for accurate triangle counting in real graph streams

Dongjin Lee[1] · Kijung Shin[2] · Christos Faloutsos[3]

Received: 11 November 2019 / Revised: 25 April 2020 / Accepted: 28 July 2020 / Published online: 12 August 2020

Name	# Nodes	# Edges	# Triangles	Description
ArXiv	30, 565	346, 849	988, 009	Citation
Facebook	61, 096	614, 797	1, 756, 259	Friendship
Email	86, 978	297, 456	1, 180, 387	Email
Youtube	3, 181, 831	7, 505, 218	7, 766, 821	Friendship
Patent	3, 774, 768	16, 518, 947	7, 515, 023	Citation
FF-Small	3, 000, 000	23, 345, 764	94, 648, 815	Synthetic
FF-Large	10, 000, 000	87, 085, 880	426, 338, 480	Synthetic
BA-Small	3, 000, 000	10^8	1, 958, 656	Synthetic
BA-Large	10, 000, 000	10^9	60, 117, 894	Synthetic
ER	1, 000, 000	10^{11}	1.333×10^{15}	Synthetic

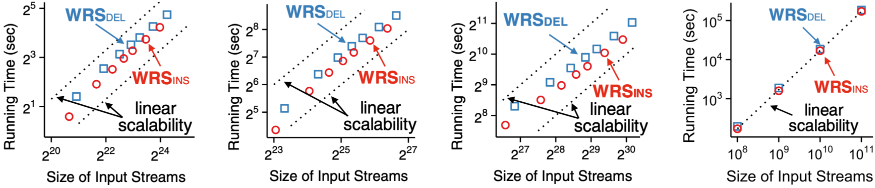
Running Time



(a) Number of Discovered Triangles (Insertion Only)

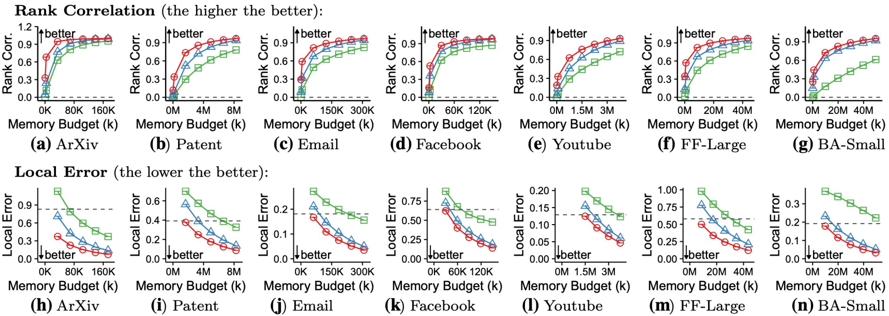
(b) Running Time (Insertion Only)

(c) Elapsed Time Per Triangle(Insertion Only)



(a) Patent **(b)** FF-Large **(c)** BA-Large **(d)** Erdős-Rényi

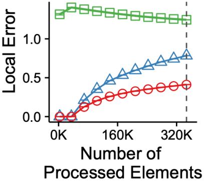
Accuracy



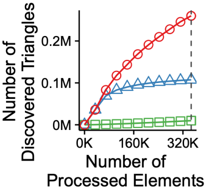
Rank Correlation (the higher the better):
Rank Corr.
better
Memory Budget (k)
0K 80K 160K
0M 4M 8M
0K 150K 300K
0K 60K 120K
0M 1.5M 3M
0M 20M 40M
(a) ArXiv
(b) Patent
(c) Email
(d) Facebook
(e) Youtube
(f) FF-Large
(g) BA-Small
Local Error (the lower the better):
Local Error
better
Memory Budget (k)
(h) ArXiv
(i) Patent
(j) Email
(k) Facebook
(l) Youtube
(m) FF-Large
(n) BA-Small

Examples

ArXiv Dataset:

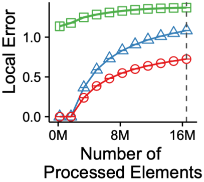


(c) Local errors over time (Insertion Only)

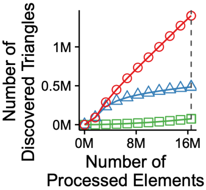


(d) Number of discovered triangles over time (Insertion Only)

Patent Dataset:



(g) Local errors over time (Insertion Only)



(h) Number of discovered triangles over time (Insertion Only)