Learning from Networks

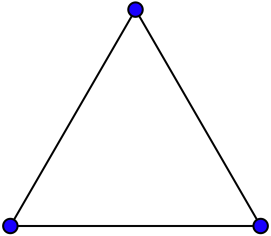
Graph Analytics: Clustering Coefficient

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 *local clustering coefficient* $cc(v)$ of v is:

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

$[0,1]$

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) ⇒ contate tutte in numeratore

triangoli possibili

Clustering Coefficient

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

Definition

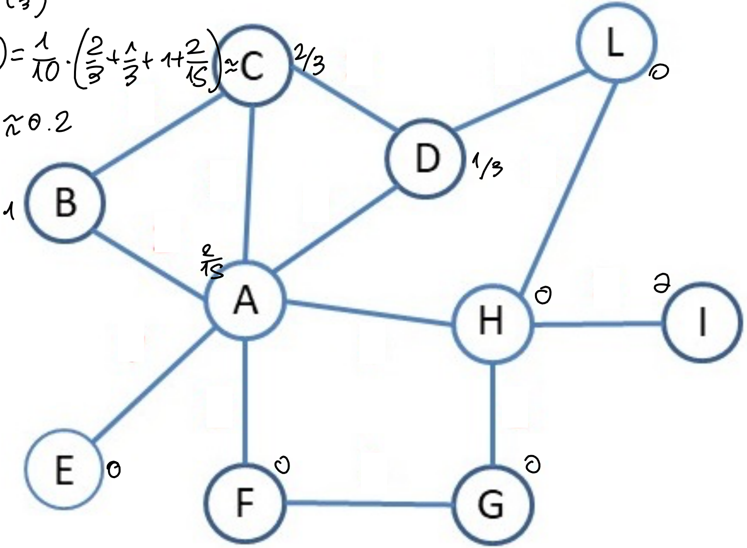
The *average local clustering coefficient* $avg_{lcc}(G)$ of G is:

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

(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? $\Theta((\text{degree}(v))^2) \in O(n^2)$

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); $\Rightarrow \Theta(\max_{v \in V} (\mathrm{degree}(v))^2)$
return values cc_v;

Complexity? $\Theta(n \cdot \max_{v \in V} (\mathrm{degree}(v))^2) \in \Theta(m\delta(G))$

$\max_{v \in V} \mathrm{degree}(v) =: \delta(G)$

$\sum_{v \in V} (\mathrm{degree}(v))^2 \leq \sum_{v \in V} (\mathrm{degree}(v) \cdot \delta(G)) = \delta(G) \sum_{v \in V} \mathrm{degree}(v) = 2m\delta(G) \in \Theta(m\delta(G))$

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

complessità di $LCC(G)$: $\Theta(\underbrace{m}_{\text{comp. di 1 pass}} \cdot \underbrace{\delta(G)}_{\#\text{pass}})$

Approximating Local Clustering Coefficients

Assume we have:

- graph $\boxed{G = (V, E) \text{ with } n = |V|,\ m = |E|}$
- 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 $\boxed{t_v^S}$ in S incident to each vertex $v \in V$

How should we fix the probability p?

If we fix $\boxed{p = \frac{M}{m}}$, 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**
    $S \leftarrow S \cup \{(u, v)\}$;
    $\mathcal{N}^S_{u,v} \leftarrow \mathcal{N}^S(u) \cap \mathcal{N}^S(v)$;
    forall $c \in \mathcal{N}^S_{u,v}$ **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 $v \in V$;

cc_v non è unbiased

vicini di entrambi fanno triangoli

qui conto triangoli 1 volta, in def. lo faccio 2 volte

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? => fattore di correzione

Analysis

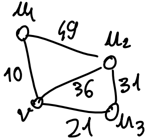
What is the expectation of cc_v?

considero $v \in V$:

$$\mathbb{E}[cc_v] = \mathbb{E}\left[\frac{2t_v^S}{\deg v(\deg v - 1)}\right] = \frac{2}{\deg v(\deg v - 1)}\mathbb{E}[t_v^S]$$

definisco $t(v) = \#$ triangoli con v in G $\Rightarrow$ ordiniamo triangoli in base a ordine con cui ultimo lato di triangolo appare in Σ $\Rightarrow$ triangoli $1, \dots, t(v)$

Ex:



dato ordine in Σ,

$$\Sigma = \dots, (v,u_1), \dots (v,u_3), \dots, (u_2,u_3), \dots, \underbrace{(v,u_2)}_{\text{ultimo lato di}}, \dots, \underbrace{(u_1,u_2)}_{\text{ultimo lato di}}, \dots$$

$$\boxed{(v,u_2,u_3), \quad (v,u_1,u_2)}$$

$$\Downarrow$$

ordine triangoli con v

$\forall i \in [1, t(v)]$, definisco r.v. $X_i = \begin{cases} 1 \text{ se triangolo } i \text{ contato da alg.} \\ 0 \text{ else} \end{cases} \Rightarrow$

$$\Rightarrow t_v^S = \sum_{i=1}^{t(v)} X_i \Rightarrow \mathbb{E}[t_v^S] = \sum_{i=1}^{t(v)} \mathbb{E}[X_i]$$

$\mathbb{E}[X_i] = \Pr[X_i = 1] = \Pr[\text{triangolo } i \text{ contato da alg.}] =$

$= \Pr[\text{tutti lati di } i \text{ in } S] = p^3 \Rightarrow$

$$\Rightarrow \mathbb{E}[t_v^S] = \sum_{i=1}^{t(v)} p^3 \Rightarrow \mathbb{E}[cc_v] = \frac{2}{\deg v(\deg v - 1)} p^3 t(v) = \boxed{p^3} cc(v)$$

$\Downarrow$

per unbiased estimator, moltiplicare cc_v per $1/p^3$

$\Downarrow$

così, $\mathbb{E}[cc_v] = cc(v) \; \forall v \in V$

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 $\mathtt{SampleElement}((u,v),p)$ **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?

$$\mathbb{E}[cc_v] = \mathbb{E}\left[\frac{2t_v^S}{\deg v(\deg v - 1)}\right] = \frac{2}{\deg v(\deg v-1)}$$

dato ordine e X_i definite prima: $\mathbb{E}[t_v^S] = \sum_{i=1}^{t(v)} \mathbb{E}[X_i]$,

$\mathbb{E}[X_i] = \mathbb{E}[\text{triangolo } i \text{ contato}] = \mathbb{E}[\text{primi due lati in } S \text{ secondo ordine}] = \frac{1}{p^2} \Rightarrow$

$\Rightarrow \mathbb{E}[t_v^S] = p^2 t(v) \Rightarrow \mathbb{E}[cc_v] = p^2 cc(v) \Rightarrow$ per unbiased, $cc_v \leftarrow cc_v / p^2$

Per entrambi alg.

- $\mathbb{E}[cc_v] = cc(v) \; \forall v \in V$
- 1 pass

Si può dimostrare che ImprovedLCC conta ogni triangolo con prob. maggiore ⇒ varianza minore ⇒ stime migliori

X_i non sono indipendenti

Analysis: Space

How many edges are stored by the two algorithms above?

$\approx M$

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

Vitter (1985)

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

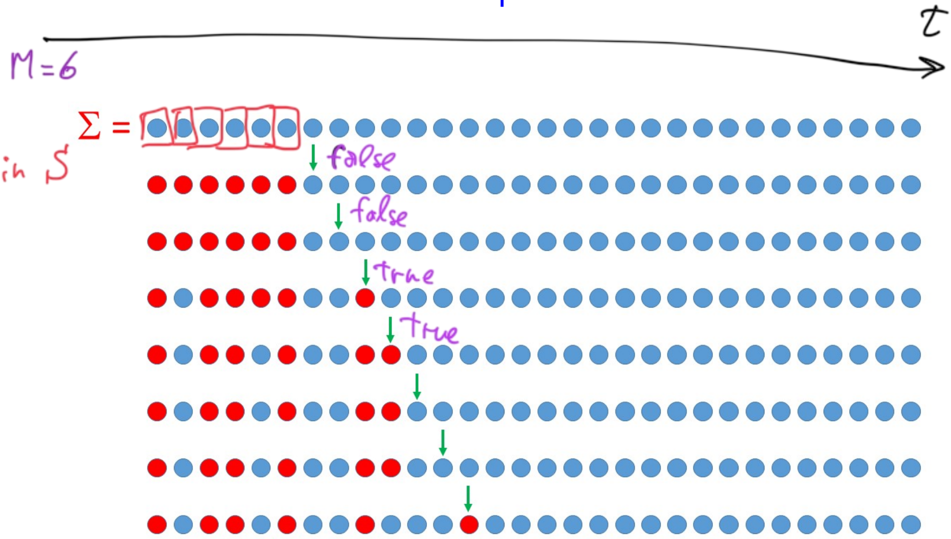
$S \leftarrow \emptyset$; $t \leftarrow 0$;
foreach x *from* Σ **do**
  $t \leftarrow t + 1$;
  if $t \leq M$ **then** $S \leftarrow S \cup \{x\}$;
  else
    if `SampleElement`$(x, M/t)$ **then**
      remove element from S chosen uniformly at random;
      $S \leftarrow S \cup \{x\}$;

returns "true" with probability M/t, and "false" = otherwise $(1 - M/t)$

among the elements of S (i.e. each element is chosen with prob. $\frac{1}{M}$)

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

Example



and so on ....

Analysis

Proposition

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

i) $|S| = M$

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

<u>Proof.</u> i) $|S| = M$: trivial

ii) By induction. Let $S_t \equiv S$ after processing $x_1, x_2, \dots, x_t$ where $t \geq M$.

Base: $t = M$, $\Pr[x \in S] = 1 = \frac{M}{M} \Rightarrow$ ok

Inductive step: consider step $t > M$

hp.: the property holds for $t-1$: $\forall\, i = 1, \dots, t-1$ we have $\Pr[x_i \in S_{t-1}] = \frac{M}{t-1}$

We need to compute $Pr[x_i \in S_t]$, $\forall i=1,\dots,t$

Two cases:

- element x_t: $Pr[x_t \in S_t] = Pr[\underbrace{\text{SampleElement}(x_t, \tfrac{M}{t})}_{=\text{ true}}]$

$$= \frac{M}{t} \Rightarrow ok$$

- elements x_i, $1 \le i < t$: define the event

  $A \equiv$ "x_t is added to S (at time t)"

Note: $Pr[A] = M/t$ and $Pr[\bar{A}] = 1 - M/t$ [$\bar{A}$ = "not A"]

Then:

law of total probability

$$Pr[x_i \in S_t] = Pr[\text{"}x_i \in S_t\text{" AND } A] + Pr[\text{"}x_i \in S_t\text{" AND } \bar{A}]$$

•) $Pr[\text{"}x_i \in S_t\text{" AND } A] = Pr[\text{"}x_i \in S_t\text{"} \mid A] \underbrace{Pr[A]}_{M/t}$

$$\Pr[\text{"}x_i \in S_t\text{"} \mid A] = \underbrace{\Pr[X_i \in S_{t-1}]}_{M/(t-1) \text{ (by induction)}} \cdot \underbrace{\Pr[X_i \text{ is not removed from } S \text{ at time } t]}_{1 - 1/M = \frac{M-1}{M}}$$

$$\Rightarrow \Pr[\text{"}x_i \in S_t\text{"} \text{ AND } A] = \frac{M}{t-1} \cdot \frac{M-1}{\cancel{M}} \cdot \frac{\cancel{M}}{t} = \frac{(M-1)M}{(t-1)t}$$

$$\cdot) \Pr[\text{"}x_i \in S_t\text{"} \text{ AND } \bar{A}] = \Pr[\text{"}x_i \in S_t\text{"} \mid \bar{A}] \cdot \underbrace{\Pr[\bar{A}]}_{1 - M/t}$$

$$\Pr[\text{"}x_i \in S_t\text{"} \mid \bar{A}] = \Pr[\text{"}x_i \in S_{t-1}\text{"}] = M/(t-1) \quad \text{(by induction)}$$

$$\Rightarrow \Pr[\text{"}x_i \in S_t\text{"} \text{ AND } \bar{A}] = \frac{M}{(t-1)} \cdot \left(1 - \frac{M}{t}\right) = \frac{(t-M) \cdot M}{(t-1)t}$$

Therefore:

$$\Pr[x_i \in S_t] = \Pr[\text{"}x_i \in S_t\text{"} \text{ AND } A] + \Pr[\text{"}x_i \in S_t\text{"} \text{ AND } \bar{A}]$$

$$= \frac{(M-1)M}{(t-1)t} + \frac{(t-M)M}{(t-1)t} = \frac{M(\cancel{M}-1+t-\cancel{M})}{(t-1)t} = \frac{M\cancel{(t-1)}}{t\cancel{(t-1)}} = \frac{M}{t}$$

$\square$

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

Fast and Accurate Triangle Counting in Graph Streams Using Predictions

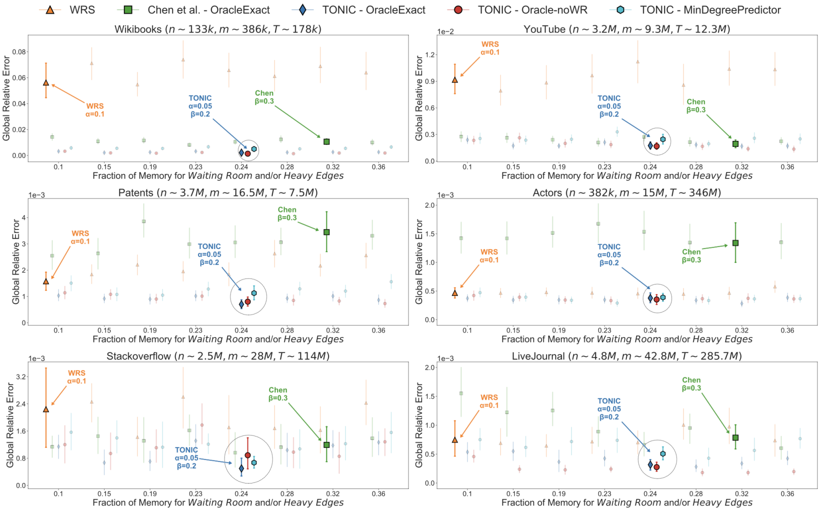
IEEE International Conference on Data Mining
9-12 December 2024, Abu Dhabi, UAE

Cristian Boldrin
Dept. of Information Engineering
University of Padova
Padova, Italy
boldrincri@dei.unipd.it

Fabio Vandin
Dept. of Information Engineering
University of Padova
Padova, Italy
fabio.vandin@unipd.it

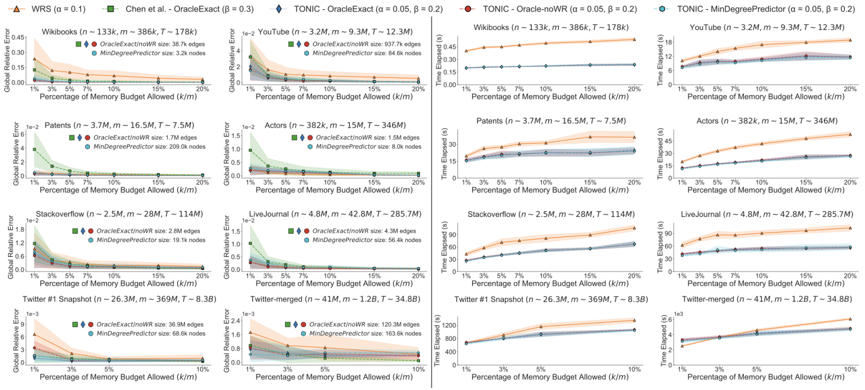
Dataset	n	m	T
Single Graphs			
Edit EN Wikibooks	$133k$	$386k$	$178k$
SOC YouTube Growth	$3.2M$	$9.3M$	$12.3M$
Cit US Patents	$3.7M$	$16.5M$	$7.5M$
Actors Collaborations	$382k$	$15M$	$346.8M$
Stackoverflow	$2.5M$	$28.1M$	$114.2M$
SOC LiveJournal	$4.8M$	$42.8M$	$285.7M$
Twitter-merged	$41M$	$1.2B$	$34.8B$
Snapshot Sequences			
Oregon (9 graphs)	$11k$	$23k$	$19.8k$
AS-CAIDA (122 graphs)	$26k$	$53k$	$36.3k$
AS-733 (733 graphs)	$6k$	$13k$	$6.5k$
Twitter (4 graphs)	$29.9M$	$373M$	$4.4B$

Accuracy



WRS
Chen et al. - OracleExact
TONIC - OracleExact
TONIC - Oracle-noWR
TONIC - MinDegreePredictor
Wikibooks ($n \sim 133k, m \sim 386k, T \sim 178k$)
YouTube ($n \sim 3.2M, m \sim 9.3M, T \sim 12.3M$)
Patents ($n \sim 3.7M, m \sim 16.5M, T \sim 7.5M$)
Actors ($n \sim 382k, m \sim 15M, T \sim 346M$)
Stackoverflow ($n \sim 2.5M, m \sim 28M, T \sim 114M$)
LiveJournal ($n \sim 4.8M, m \sim 42.8M, T \sim 285.7M$)
Global Relative Error
Fraction of Memory for *Waiting Room* and/or *Heavy Edges*
WRS α=0.1
TONIC α=0.05 β=0.2
Chen β=0.3

TIme and Memory



Dynamic Scenario