Counting and Sampling Triangles from a Graph Stream

A. Pavan, Kanat Tangwongsan, Srikanta Tirthapura, Kun-Lung Wu

2013, VLDB Conference

Marten Heidemeyer

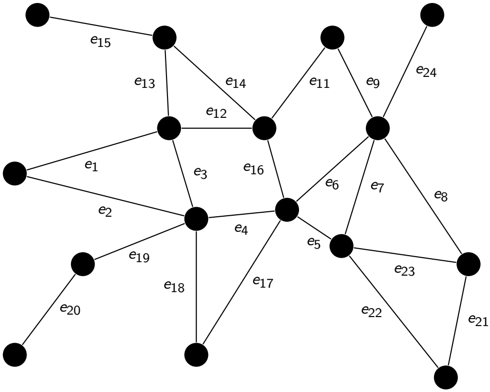
Presentation for CMPT 843 Class

Outline

Motivation

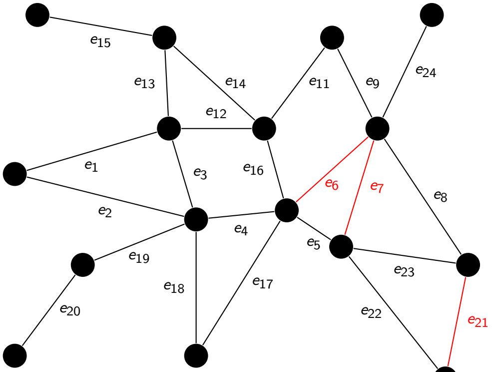
- Triangle Counting is important in:
  - social networks
  - spam and fraud detection
  - link classification and recommendation
- Why use a streaming algorithm?
  - real-time processing of live data
  - analysis of **large** disk-resident graph data

The Adjacency Stream Model
1



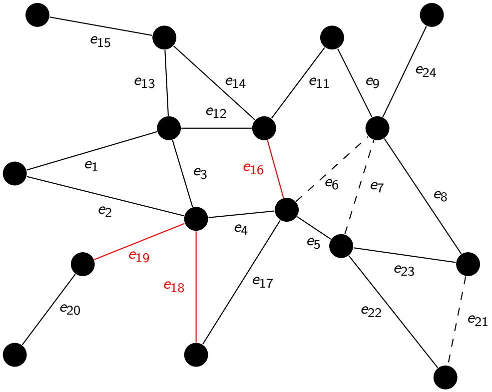
$\cdots\ e_{23}\ e_{17}\ e_{4}\ e_{11}\ e_{12}\ e_{1}\ e_{2}\ e_{5}\ e_{16}\ e_{18}\ e_{19}\ e_{6}\ e_{21}\ e_{7}$

The Adjacency Stream Model
2



$\cdots\ e_{23}\ e_{17}\ e_4\ e_{11}\ e_{12}\ e_1\ e_2\ e_5\ e_{16}\ e_{18}\ e_{19}\ \boxed{e_6\ e_{21}\ e_7}$

The Adjacency Stream Model
3



$\cdots\ e_{24}\ e_{14}\ e_{22}\ e_{23}\ e_{17}\ e_4\ e_{11}\ e_{12}\ e_1\ e_2\ e_5\ \boxed{e_{16}\ e_{18}\ e_{19}}$

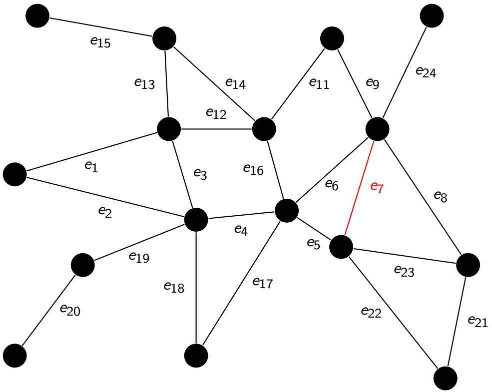
Neighborhood Sampling

- first sample a random edge r_1 from the edge stream
- then sample a random edge r_2 from those edges that appear after r_1 and are adjacent to r_1
- try to close r_1 and r_2 with a subsequent edge, to form a triangle t
- simple data structure: (r_1, r_2, t, c)
- sample r_1 and r_2 using *reservoir sampling*

The Algorithm

Upon receiving edge e **at time step** m
if `coin(1/m)`="head" **then**
    $(r_1, r_2, t, c) = (e, \emptyset, \emptyset, 0)$
else
    if e is adjacent to r_1 **then**
        $c \leftarrow c + 1$
        if `coin(1/c)`="head" **then**
            $(r_2, t) \leftarrow (e, \emptyset)$
        else
            if e forms a triangle with r_1 and r_2 **then**
                $t \leftarrow \{r_1, r_2, e\}$
            end if
        end if
    end if
end if

Example 1

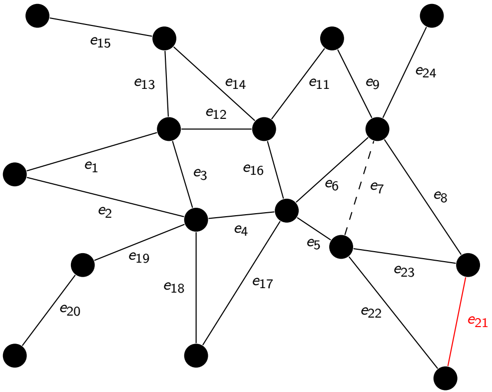


$\cdots\ e_{23}\ e_{17}\ e_4\ e_{11}\ e_{12}\ e_1\ e_2\ e_3\ e_8\ e_5\ e_{14}\ e_{23}\ e_{21}\ \boxed{e_7} \cdots$

$m = 1042$

$estimator : r_1 = e_{15}\ r_2 = e_{39}\ t = \emptyset\ c = 15$

Example 2

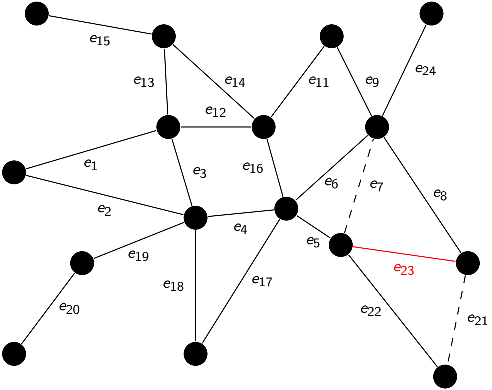


$$\cdots\ e_{13}\ e_{23}\ e_{17}\ e_4\ e_{11}\ e_{12}\ e_1\ e_2\ e_3\ e_8\ e_5\ e_{14}\ e_{23}\ \boxed{e_{21}}\cdots$$

$m = 1043$

$estimator : r_1 = e_7\ r_2 = \emptyset\ t = \emptyset\ c = 0$

Example 3

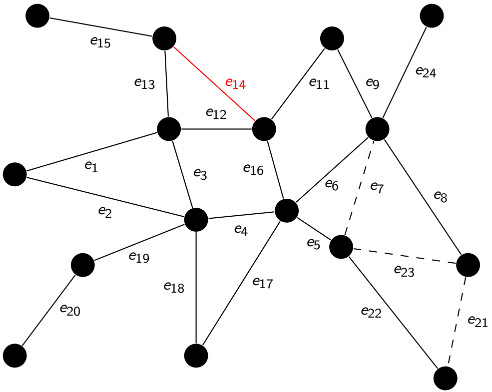


$$\cdots \; e_9 \; e_{13} \; e_{23} \; e_{17} \; e_4 \; e_{11} \; e_{12} \; e_1 \; e_2 \; e_3 \; e_8 \; e_5 \; e_{14} \; \boxed{e_{23}} \cdots$$

$m = 1044$

$estimator : r_1 = e_7 \;\; r_2 = e_{23} \;\; t = \emptyset \;\; c = 1$

Example 4

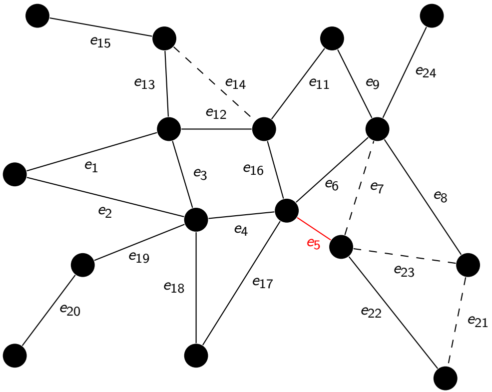


$\cdots\ e_{16}\ e_9\ e_{13}\ e_{23}\ e_{17}\ e_4\ e_{11}\ e_{12}\ e_1\ e_2\ e_3\ e_8\ e_5\ \boxed{e_{14}} \cdots$

$m = 1045$

$estimator : r_1 = e_7\ r_2 = e_{23}\ t = \emptyset\ c = 1$

Example 5

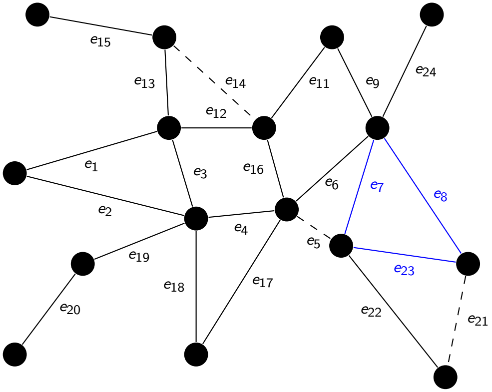


$\cdots\ e_{18}\ e_{16}\ e_{9}\ e_{13}\ e_{23}\ e_{17}\ e_{4}\ e_{11}\ e_{12}\ e_{1}\ e_{2}\ e_{3}\ e_{8}\ \boxed{e_{5}}\cdots$

$m = 1046$

$estimator : r_1 = e_7\ r_2 = e_{23}\ t = \emptyset\ c = 2$

Example 6



$$\cdots\ e_{20}\ e_{18}\ e_{16}\ e_9\ e_{13}\ e_{23}\ e_{17}\ e_4\ e_{11}\ e_{12}\ e_1\ e_2\ e_3\ \boxed{e_8}\ \cdots$$

$m = 1047$

estimator : $r_1 = e_7\ r_2 = e_{23}\ t = \{e_7, e_{23}, e_8\}\ c = 3$

Expectation value of τ

- We want to find the number of triangles $\tau(G)$
- Let t and c be the values the neighborhood sampling algorithm maintains and m be the number of edges observed so far. Define:

$$\tilde{\tau} = \begin{cases} c \times m & \text{if } t \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Then $\mathbf{E}[\tilde{\tau}] = \tau(G)$

An (ϵ, δ)-approximation to the triangle count

- Take the average of r such estimators with
- With probability $1 - \delta$ the average of the estimators is in

$$[(1-\epsilon)\tau(G), (1+\epsilon)\tau(G)]$$

An (ϵ, δ)-approximation to the triangle count

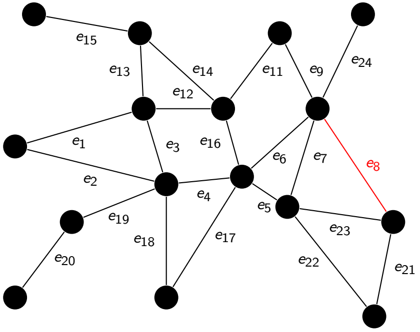
- Take the average of r such estimators with
- With probability $1 - \delta$ the average of the estimators is in

$$[(1-\epsilon)\tau(G), (1+\epsilon)\tau(G)]$$

- Relation between number of estimators and error of the approximation:

$$r \geq \frac{6}{\epsilon^2} \frac{m\Delta}{\tau(G)} log(\frac{2}{\delta})$$

- LiveJournal:
  - given $m = 34.7M, \Delta = 14815, \tau(G) = 177,820,130$
  - chose $\epsilon = 0.1, \delta = 0.05$
  - then $r \geq 6.3M$



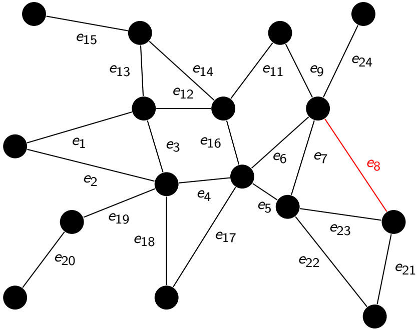
$estimator1 : r_1 = e_{101}\ r_2 = \emptyset\ t = \emptyset\ c = 0$
$estimator2 : r_1 = e_9\ r_2 = e_{23}\ t = \emptyset\ c = 13$
$estimator3 : r_1 = e_8\ r_2 = \emptyset\ t = \emptyset\ c = 33$
$estimator4 : r_1 = e_{99}\ r_2 = e_{42}\ t = \{e_{99}, e_{42}, e_{101}\}\ c = 33$
$estimator5 : r_1 = e_7\ r_2 = \emptyset\ t = \emptyset\ c = 0$
$estimator6 : r_1 = e_8\ r_2 = e_{23}\ t = \{e_7, e_{23}, e_8\}\ c = 31$
$estimator7 : r_1 = e_{42}\ r_2 = \emptyset\ t = \emptyset\ c = 0$
$estimator8 : r_1 = e_{17}\ r_2 = e_{42}\ t = \{e_{17}, e_{42}, e_{23}\}\ c = 92$

$\vdots$

$\vdots$

$m = 1042$

$\cdots\ e_{20}\ e_{18}\ e_{16}\ e_9\ e_{13}\ e_{23}\ e_{17}\ e_4\ e_{11}\ e_{12}\ e_1\ e_2\ e_3\ \boxed{e_8}\cdots$



$estimator1 : r_1 = e_{101}\ \ r_2 = \emptyset\ \ t = \emptyset\ \ c = 0$
$estimator2 : r_1 = e_9\ \ r_2 = e_{23}\ \ t = \emptyset\ \ c = 13$
$estimator3 : r_1 = e_8\ \ r_2 = \emptyset\ \ t = \emptyset\ \ c = 33$
$estimator4 : r_1 = e_{99}\ \ r_2 = e_{42}\ \ t = \{e_{99}, e_{42}, e_{101}\}\ \ c = 33$
$estimator5 : r_1 = e_7\ \ r_2 = \emptyset\ \ t = \emptyset\ \ c = 0$
$estimator6 : r_1 = e_8\ \ r_2 = e_{23}\ \ t = \{e_7, e_{23}, e_8\}\ \ c = 31$
$estimator7 : r_1 = e_{42}\ \ r_2 = \emptyset\ \ t = \emptyset\ \ c = 0$
$estimator8 : r_1 = e_{17}\ \ r_2 = e_{42}\ \ t = \{e_{17}, e_{42}, e_{23}\}\ \ c = 92$

$\vdots$

$\vdots$

$m = 1042$

$$\tilde{\tau} = \begin{cases} c \times m & \text{if } t \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

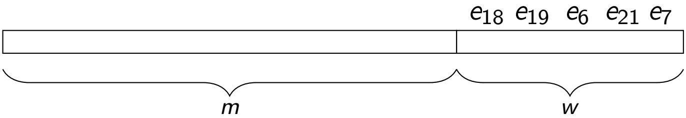
Nearly-Linear Time Triangle Counting

- so far we have a $O(mr)$-time implemenation
- with a small constant factor increase in space, we are able to achieve $O(m + r)$-time bound
- **Bulk Processing** Process edges in bulk. Read w edges at a time and update all estimators simultaneously.

$$\cdots\ e_{23}\ e_{17}\ e_4\ e_{11}\ e_{12}\ e_1\ e_2\ e_5\ e_{16}\ \boxed{e_{18}\ e_{19}\ e_6\ e_{21}\ e_7}$$

Bulk Processing Step 1: Resample Level 1 edges

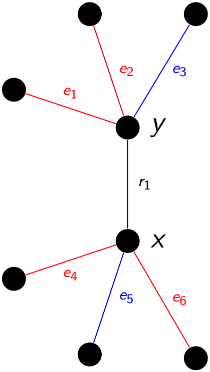
- sample new r_1 edges from the batch of size w. Keep current edge with probability $\frac{m}{w+m}$ and with remaining probability, replace it with edge uniformly chosen from B.
- for each estimator: draw random number between 1 and $m + w$.



Bulk Processing Step 2: Identify Level-2 candidates and sample from them

- The sample space we want to sample r_2 edges from, is those edges that are adjacent to r_1 and arrive after r_1.
- Let $N(r_1)$ be those edges that arrive after r_1 in the stream.
- Suppose for each estimator we have the following values $(r_1 = (x, y))$:
  - $|N(r_1)| = c^- + c^+$
  - c^-: # edges in $N(r_1)$ before the badge.
  - $c^+ = a + b$: # of edges in $N(r_1)$ inside the badge.
  - a: # of edges in $N(r_1)$, sharing endpoint x with r_1, inside badge.
  - b: # of edges in $N(r_1)$, sharing endpoint y with r_1, inside badge.

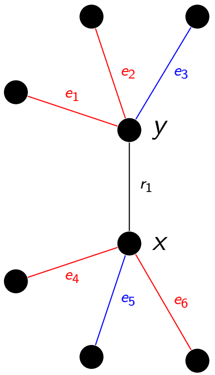
Example: c^-, c^+, a, b



- $|N(r_1)| = c^- + c^+ = 6$
- $c^- = 2$
- $c^+ = a + b = 4$
- $a = 2$
- $b = 2$

$\boxed{e_6 \quad e_4 \quad e_{23} \quad e_1 \quad e_{13} \quad e_2}$ e_{42} e_{17} e_5 e_3 r_1

Example: c^-, c^+, a, b

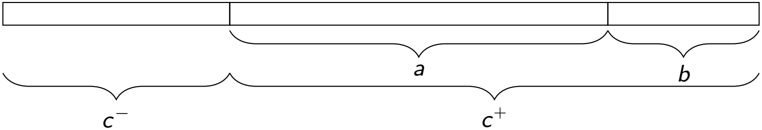


- $|N(r_1)| = c^- + c^+ = 6$
- $c^- = 2$
- $c^+ = a + b = 4$
- $a = 2$
- $b = 2$
- with probability $\frac{c^+}{c^+ + c^-}$ sample a r_1 adjacent edge from the badge as new r_2

$\boxed{e_6 \quad e_4 \quad e_{23} \quad e_1 \quad e_{13} \quad e_2}$ e_{42} e_{17} e_5 e_3 r_1

Sampling new r_2 edges from the badge

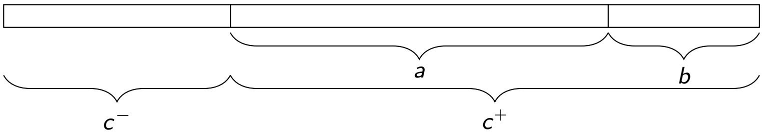
- For each estimator draw random number φ between 1 and $c^- + c^+$



- if $\varphi \leq c^-$ don't sample new r_2 edge
- if $c^- \leq \varphi \leq c^- + a$ sample one of the edges that share endpoint x
- else sample one of the edges that share endpoint y

Sampling r_2 edges summary

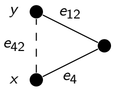
- paper presents simple algorithm to get c^-, c^+, a, b which goes over the badge once
- then we go over all estimators to sample the r_2 edges
- then go over badge again to find edge that corresponds to offset in range $c^-, c^- + a$ or $c^- + a, c^- + a + b$



Bulk Processing Step 3: Detect edges that close the wedges

- For each estimator with r_1 and r_2 edges, check if badge contains an edge that comes after r_2 and closes r_1, r_2 to form a triangle
- Create hashtable that maps the needed edges to the estimators.
- Check if badge contains this edge and whether it comes after r_2

est.1: $r_1 = e_{12}, r_2 = e_4, t = \emptyset \dots$



x,y	*est*.1
c,d	*est*.3
u,v	*est*.12

e_2 e_{42} e_{17} e_5 e_3 e_4

Sample k uniformly chosen triangles with replacement

- so far we have an efficient algorithm to **count** triangles
- how do we **sample** triangles?
  - let t and c be the values that are maintained in the estimators
  - Define `unifTri(G)` $= \begin{cases} t & \text{with prob.} \frac{c}{2\Delta} \\ \emptyset & \text{otherwise} \end{cases}$
  - If `unifTri(G)` produces a triangle, each triangle in the graph is equally likely to be produced
  - run r copies of `unifTri(G)`, to sample k uniformly chosen triangles from the graph
  - success with probability at least $1 - \delta$ as long as

$$r \geq \frac{4mk\Delta ln(e/\delta)}{\tau(G)}$$

What we compare with

The space complexity of the presented method for triangle counting is

$$O(s(\epsilon, \delta) m\Delta / \tau(G))$$

- Jowhari and Ghodsi, 2005:
  - space and per-edge time: $O(s(\epsilon, \delta) m\Delta^2 / \tau(G))$
- Pagh and Tsourakakis, 2012:
  - space: $O(1/\epsilon^2 \cdot m\sigma / \tau(G) \cdot log(1/\delta))$
- Kane, 2012:
  - space: $O(s(\epsilon, \delta) \cdot m^3 / \tau(G)^2)$

- *Syn 3-reg* $n = 2000$, $m = 3000$, $\Delta = 3$, $\tau = 1000$

Algorithm	$r = 1,000$		$r = 10,000$		$r = 100,000$	
	MD	Time	MD	Time	MD	Time
JG [9]	7.20	0.04	2.08	0.44	0.27	5.26
Ours	4.28	0.004	1.52	0.01	0.93	0.07

- *Hep-Th* $n = 9877$, $m = 51971$, $\Delta = 130$, $\tau = 90649$

Algorithm	$r = 1,000$		$r = 10,000$		$r = 100,000$	
	MD	Time	MD	Time	MD	Time
JG [9]	79.33	0.71	86.86	7.17	86.66	86.02
Ours	92.69	0.05	81.25	0.08	0.68	0.17

Evaluation on real graphs

Dataset	r = 1K		r = 128K		r = 1M		I/O
	min/mean/max dev.	Time	min/mean/max dev.	Time	min/mean/max dev.	Time	
Amazon	1.60 / 6.28 / 12.45	0.41	0.11 / 0.84 / 1.52	1.06	0.08 / 0.25 / 0.40	3.72	0.26
DBLP	8.04 / 18.28 / 36.53	0.45	0.08 / 0.50 / 0.97	1.08	0.07 / 0.19 / 0.42	3.90	0.28
Youtube	12.56 / 59.45 / 79.76	1.25	9.37 / 21.46 / 38.49	2.39	1.75 / 4.42 / 10.18	5.26	0.79
LiveJournal	0.24 / 11.53 / 29.76	15.00	1.41 / 2.35 / 4.02	23.10	0.19 / 0.60 / 1.45	33.40	10.00
Orkut	4.61 / 31.93 / 58.93	52.40	2.13 / 4.69 / 12.69	75.20	1.48 / 3.55 / 5.93	103.00	33.40
Syn. ~d-regular	1.26 / 7.58 / 13.57	53.70	0.00 / 0.37 / 0.81	64.80	0.01 / 0.24 / 0.53	73.00	34.50

	r = 1K	r = 128K	r = 1M
Memory	36K	4.5M	36M

What I didn't talk about

- **Transitivity Coefficient**
  $\mathcal{K}(G) = \frac{3\tau(G)}{\zeta(G)}$, where $\zeta(G)$ is number of connected triplets
- Extension to counting and sampling higher-order-cliques (*4-Cliques*,...)