A Comparison of Two Triangle Counting Approximation Methods

Antonia Calia-Bogan, Richard Massimilla, Gabriel Orlanski

December 2, 2020

Outline

- Counting Triangles
  - Problem Description
  - Existing Solutions
- Discussion of the two Papers
  - "Fast Counting of Triangles in Large Real Networks: Algorithms and Laws"[11]
  - "Counting Triangles in Large Graphs using Randomized Matrix Trace Estimation"[1]
- Empirical Results
  - Analysis of the proposed algorithms
  - Comparison between the two papers

Counting Triangles

- Common task in graph-mining
- Many real world applications
  - Social Networks, Link Recommendations, etc.
- Exact counting is expensive with respect to time and memory
  - Total nodes on the internet is in the order of 10^{10}
  - Each webpage has approximately 20-30 links on it [3]

Widely Used Solutions

- `NodeIterator`:

  - Considers each one of the nodes and examines which pairs of its neighbors are connected

- `EdgeIterator`:

  - Algorithm computes for each edge the number of triangles that contain it

- Both have asymptotic time complexity of $O\left(\frac{|E|^2}{|V|}\right)$ and $\Theta(\sum_{v \in V} \deg(v)^2)$ [10]

First Paper

- In 2008, Charalampos Tsourakakis from Carnegie Mellon University published the paper "Fast Counting of Triangles in Large Real Networks: Algorithms and Laws" in *2008 Eighth IEEE International Conference on Data Mining*.
- He proposed `SpectralCount` to get the **exact** $\#$ of triangles in an undirected graph G
- To reduce the time and memory requirements of `SpectralCount`, he proposed the algorithms `EigenTriangle` and `EigenTriangleLocal` to approximate the number of triangles

SpectralCount

Given an adjacency matrix $\mathbf{A}$ for an undirected graph G:

- Every diagonal element $\alpha_{ii} \in \mathbf{A}^3$ is number of paths of length 3 that begin and end at V_i, which will be exactly the paths from V_i to itself that are triangles
- The trace of $\mathbf{A}^3$ is $3 * \triangle(G)$, because each triangle must have 3 distinct nodes
- Because G is undirected, each triangle is counted as 2.
  - Traingle $\triangle_{ijk}$ is counted both as $i \to k \to j \to i$ and $i \to j \to k \to i$

Therefore we finally end up with

$$\triangle(G) = \frac{1}{6}\mathrm{trace}(\mathbf{A}^3) \tag{1}$$

This will give the **exact** number of triangles, although it is very expensive both in terms of time and memory. [11]

EigenTriangle

Because of `SpectralCount`'s expensive nature, Tsourakakis proposed `EigenTriangle` to estimate $\triangle(G)$ using

$$\text{If } \lambda \text{ is an eigenvalue of } \mathbf{A} \text{ then } \lambda^k \text{ is an eigenvalue of } \mathbf{A}^k \text{ if } k \geq 1 \tag{2}$$

$$\text{trace}(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i \tag{3}$$

Combining both (2) and (3) with (1) we get

$$\triangle(G) = \frac{1}{6}\text{trace}(\mathbf{A}^3) = \frac{1}{6}\sum_{i=1}^{n} \lambda_i^3$$

Lanczos Algorithm

Iterative algorithm used for estimating extreme (very large or very small) eigenvalues and corresponding eigenvectors of large sparse and symmetric matrices. Created in 1950 by Cornelius Lanczos [8].
Given a sparse symmetric matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$ and number of iterations $k \in \mathbb{Z}, k > 0$ return

- $\lambda_i, i \in [1, \ldots, k]$. The largest k eigenvalues of $\mathbf{X}$.
- $\mathbf{u}_i, i \in [1, \ldots, k]$. The corresponding eigenvectors vectors.

Note

This only works if $k \ll n$
These are only approximations of the true eigenvalues $\lambda_1^*, \ldots, \lambda_N^*$ and eigenvectors $\mathbf{u}_1^*, \ldots, \mathbf{u}_N^*$ for the matrix $\mathbf{X}$.

Full `EigenTriangle` Algorithm

- The absolute value of the top few eigenvalues are skewed and follow a power law [5, 4]
- Their signs alternate [6]
- The time complexity is $O(c|E|)$
  - c is # of matrix multiplications done by `LanczosMethod` [11]

Algorithm 1: `EigenTriangle`

Input: `tol` $\longrightarrow$ Tolerance
Output: Estimation of $\triangle(G)$

Form the Adjacency Matrix $\mathbf{A}$
$\lambda_1 \longleftarrow \texttt{LanczosMethod}(\mathbf{A}, 1)$
$\Lambda \longleftarrow [\lambda_1]$
$i \longleftarrow 1$
repeat
  $i \longleftarrow i + 1$
  $\lambda_i \longleftarrow \texttt{LanczosMethod}(A, i)$
  $\Lambda \longleftarrow [\Lambda \ \ \lambda_i]$
until $0 \leq \frac{|\lambda_i^3|}{\sum_{j=1}^{i} \lambda_j^3} \leq \texttt{tol}$
return $\frac{1}{6} \sum_{\ell \in \Lambda} \ell^3$

Second Paper

- In 2010, Haim Avron from Tel-Aviv University published the paper "Counting Triangles in Large Graphs using Randomized Matrix Trace Estimation" in *Workshop on Large-scale Data Mining: Theory and Applications*.
- Proposes $\texttt{TraceTriangle}_{\texttt{N}}$, $\texttt{TraceTriangle}_{\texttt{R}}$, and $\texttt{TraceTriangle}_{\texttt{M}}$ as a faster and more accurate triangle approximation than `EigenTriangle`
- $\texttt{TraceTriangle}_{\texttt{N}}$ and $\texttt{TraceTriangle}_{\texttt{R}}$ use random sampling and an unbiased estimators instead of `LanczosMethod`

Gaussian and Hutchinson Trace Estimators

For a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, both the Gaussian and Hutchinson Trace Estimators are defined as

$$\mathbb{E}\left[\frac{1}{M}\sum_{i=1}^{M}\mathbf{z}_i^T\mathbf{A}\mathbf{z}_i\right] = \mathrm{trace}(\mathbf{A})$$

where $\mathbf{z}_1, \ldots, \mathbf{z}_M$ are independent random vectors following a distribution with zero mean and unity variance and whose entries are i.i.d and have $||\mathbf{z}_i||_2 = n$.

Proof of Unbiased Estimation

Let $\mathbf{B}$ be an $n \times n$ symmetric matrix and let $\mathbf{u} = (u_1, \ldots, u_n)^\top$ be a vector of n independent samples from random variable U with zero mean and variance σ^2. Then $\mathbb{E}\left[\mathbf{u}^\top \mathbf{B}\mathbf{u}\right] = \sigma^2 \mathrm{trace}(\mathbf{B})$. This makes $\frac{1}{\sigma^2}\mathbf{u}^\top \mathbf{B}\mathbf{u}$ an unbiased estimator of $\mathrm{trace}(\mathbf{B})$.

$$
\begin{aligned}
\mathbb{E}\left[\mathbf{u}^\top \mathbf{B}\mathbf{u}\right] &= \mathbb{E}\left[\sum_{i,j} u_i u_j B_{ij}\right] \\
&= \sum_{\substack{i,j \\ i \neq j}} \mathbb{E}[u_i]\mathbb{E}[u_j] B_{ij} + \sum_{\substack{i,j \\ i = j}} \mathbb{E}[u_i^2] B_{ii} \\
&= \sigma^2 \sum_i B_{ii} \\
&= \sigma^2 \mathrm{trace}(\mathbf{B})
\end{aligned}
$$

Differences Between the Two Estimators

- The **Gaussian Trace Estimator** uses $\mathcal{N}(0,1)$.[1]
- The **Hutchinson Trace Estimator** uses a Rademacher distribution with equal probability[7]

This leads to significant differences in the variances

- $\mathrm{Var}(\textbf{Gaussian Trace Estimator}) = 2\left\|\mathbf{A}\right\|_F^2$. [1, 2]
- $\mathrm{Var}(\textbf{Hutchinson Trace Estimator}) = 2(\left\|\mathbf{A}\right\|_F^2 - \sum_{i=1}^{n} \mathbf{A}_{ii}^2)$. [2, 7]

$$\left\|\mathbf{X}\right\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |\mathbf{X}_{ij}|^2}, \text{ Where } \mathbf{X} \in \mathbb{R}^{m\times n}$$

$\texttt{TraceTriangle}_{\texttt{N}}$

- Overall Runtime of $O(|E| \log^2 |V|)$
- Parallelization Reduces Space used per machine from $O(|V| \log^2 |V|)$ to $O(|V|)$ using $O(\log^2 |V|)$ independent machines

Algorithm 2: $\texttt{TraceTriangle}_{\texttt{N}}$

Input: $\gamma \longleftarrow$ a scalar
Output: Estimation of $\triangle(G)$

Form the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
$M = \lceil \gamma \ln^2 n \rceil$
for $i \in 1, \ldots, M$ **do**
  Form the vector $\mathbf{x} = [x_0, \ldots, x_n]$, where $x_k \sim \mathcal{N}(0, 1)$ are i.i.d. $k \in 1, \ldots, n$
  $y \longleftarrow \mathbf{Ax}$
  $T_i \longleftarrow (y^T \mathbf{A} y)/6$
$\triangle \longleftarrow \frac{1}{M} \sum_{i=1}^{M} T_i$

$\texttt{TraceTriangle}_\texttt{R}$

- Same runtime and space costs as $\texttt{TraceTriangle}_\texttt{N}$
- It is less expensive with respect to Implementation [1] due to the lower cost of sampling from a Rademacher distribution when compared to something continuous like a Gaussian

Algorithm 3: $\texttt{TraceTriangle}_\texttt{R}$

Input: $\gamma \longleftarrow$ a scalar
Output: Estimation of $\triangle(G)$

Form the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
$M = \lceil \gamma \ln^2 n \rceil$
for $i \in 1, \ldots, M$ **do**
    Form the vector $\mathbf{x} = [x_0, \ldots, x_n]$, where $x_k = \pm 1$ with equal probability and are i.i.d. $k \in 1, \ldots, n$
    $y \longleftarrow \mathbf{Ax}$
    $T_i \longleftarrow (y^T \mathbf{A} y)/6$
$\triangle \longleftarrow \frac{1}{M} \sum_{i=1}^{M} T_i$

Analytical advantages

- `TraceTriangle` does not depend on a tolerance
- `TraceTriangle` is embarrassingly parallelizable
- `TraceTriangle` has guarantees of convergence

LEMMA 4. *Let $\delta > 0$ be a failure probability and let $\epsilon > 0$ be a relative error. For $M \geq 20\epsilon^{-2}\rho(A)^2 \ln(4/\delta)$, the Gaussian trace estimator G_M of a symmetric matrix $A \in \mathbb{R}^{n\times n}$ satisfies*

$$\Pr\left(|G_M - \mathrm{trace}(A)| \leq \epsilon\,|\mathrm{trace}(A)|\right) \geq 1 - \delta\,.$$

LEMMA 5. *Let $\delta > 0$ be a failure probability and let $\epsilon > 0$ be a relative error For $M \geq 6\epsilon^{-2}\rho(A)^2 \ln(2\,\mathrm{rank}(A)/\delta)$, the Hutchinson trace estimator H_M of a symmetric matrix $A \in \mathbb{R}^{n\times n}$ satisfies*

$$\Pr\left(|H_M - \mathrm{trace}(A)| \leq \epsilon\,|\mathrm{trace}(A)|\right) \geq 1 - \delta\,.$$

Lemmas from [1]

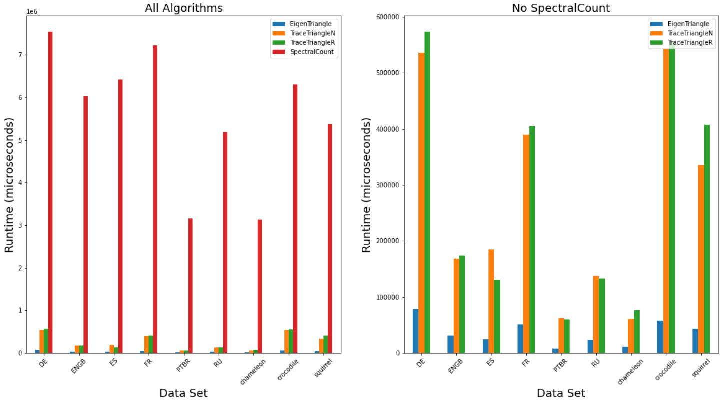
Our Results - Datasets

- **Wiki-[animals]:** Wikipedia page-page networks on chameleons, crocodiles, and squirrels. Each node is an article from the English Wikipedia, edges are mutual links. [9]
- **Twitch-[country]:** User-user network where each node is a user and edges represent mutual friendship. [9]

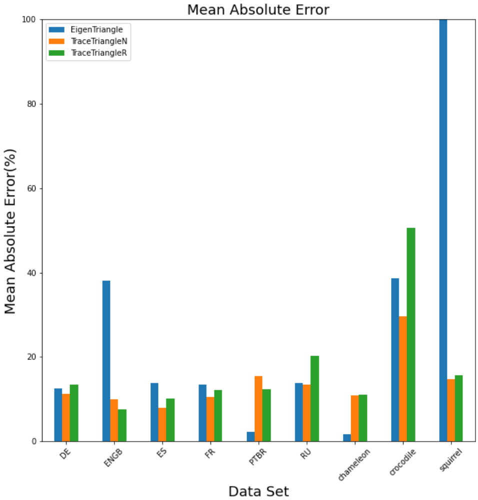
Our Results - Dataset Information

Name	Nodes	Edges	Density	Transitivity	Triangles
DE	9,498	153,138	0.003	0.047	603,088
EN	7,126	35,324	0.002	0.042	29,266
ES	4,648	59,382	0.006	0.084	200,144
FR	6,549	112,666	0.005	0.054	422,694
PT	1,912	31,299	0.017	0.131	173,510
RU	4,385	37,304	0.004	0.049	71,445
Chameleon	2,277	31,421	0.012	0.314	345,064
Crocodile	11,631	170,918	0.008	0.026	630,879
Squirrel	5,201	198,493	0.015	0.348	9,604,843

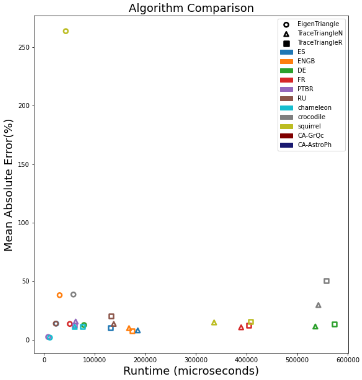
Our Results - Runtime

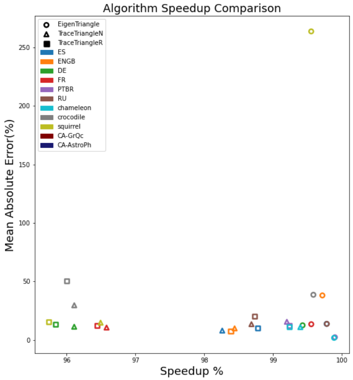


Our Results - Error

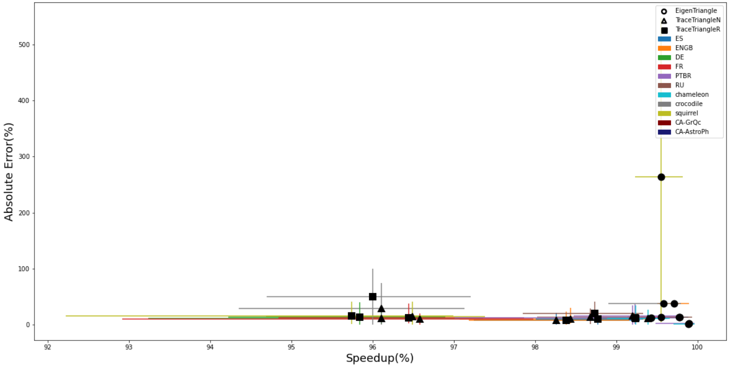


Our Results - Error Vs Time

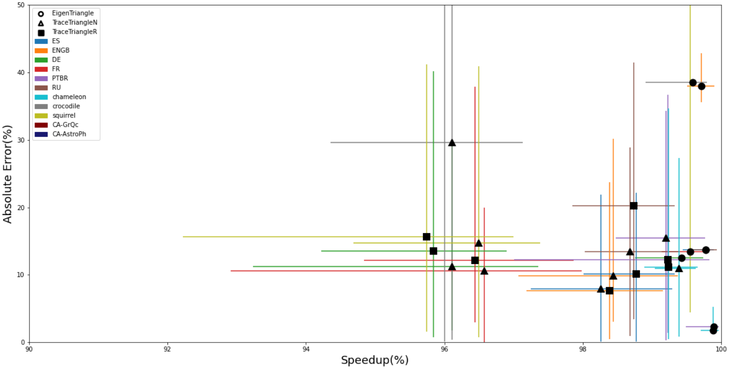




Our Results - Variances



Our Results - Variances - Excluding Outliers



References I

- H. Avron. Counting triangles in large graphs using randomized matrix trace estimation. In *Workshop on Large-scale Data Mining: Theory and Applications*, volume 10, 2010.
- H. Avron and S. Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):1–34, 2011.
- L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–24, 2008.
- F. Chung, L. Lu, and V. Vu. Eigenvalues of random power law graphs. *Annals of Combinatorics*, 7(1):21–33, 2003.

References II

M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *ACM SIGCOMM computer communication review*, 29(4):251–262, 1999.

I. J. Farkas, I. Derényi, A.-L. Barabási, and T. Vicsek. Spectra of "real-world" graphs: Beyond the semicircle law. *Physical Review E*, 64(2):026704, 2001.

M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

B. Rozemberczki, C. Allen, and R. Sarkar. Multi-scale attributed node embedding, 2019.

References III

T. Schank and D. Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *International workshop on experimental and efficient algorithms*, pages 606–609. Springer, 2005.

C. E. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *2008 Eighth IEEE International Conference on Data Mining*, pages 608–617. IEEE, 2008.