# Fast Counting of Triangles in Large Real Networks: Algorithms and Laws

Charalampos E. Tsourakakis
Machine Learning Department, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213-3891, USA
ctsourak@cs.cmu.edu

## Abstract

*How can we quickly find the number of triangles in a large graph, without actually counting them? Triangles are important for real world social networks, lying at the heart of the clustering coefficient and of the transitivity ratio. However, straight-forward and even approximate counting algorithms can be slow, trying to execute or approximate the equivalent of a 3-way database join.*

*In this paper, we provide two algorithms, the* EigenTriangle *for counting the total number of triangles in a graph, and the* EigenTriangleLocal *algorithm that gives the count of triangles that contain a desired node. Additional contributions include the following: (a) We show that both algorithms achieve excellent accuracy, with up to* $\approx$ *1000x faster execution time, on several, real graphs and (b) we discover two new power laws (*Degree-Triangle *and* TrianglePartication *laws) with surprising properties.*

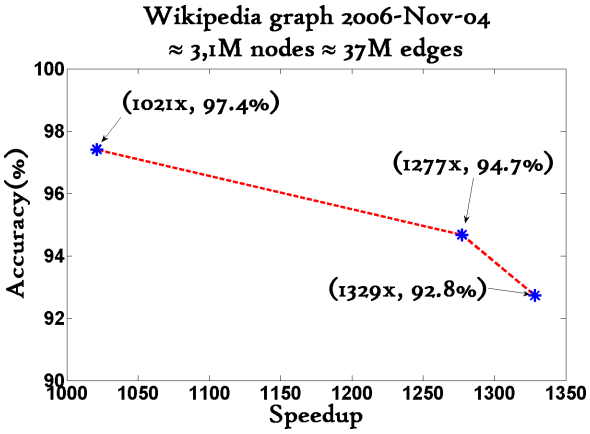



**Figure 1. Speed-up ratio versus accuracy for the Wikipedia web graph (**$\approx 3,1M$ **nodes,** $\approx 37M$ **edges ). Proposed method achieves 1021x faster time, for 97.4% accuracy, compared to a typical competitor, the** *Node Iterator* **method.**

## 1 Introduction

Finding patterns in large scale graphs, with millions and billions of edges is attracting increasing interest, with numerous applications in computer network security (intrusion detection, spamming), in web applications (community detection, blog analysis) in social networks (facebook, linkedin, for link prediction), and many more. One of the operations of interest in such a setting is the estimation of the clustering coefficient and the transitivity ratio, which effectively translates to the number of triangles in the graph, or the number of triangles that a node participates in.

It is known that in social networks there is a higher-than-random number of triangles ([27]). The reason is that friends of friends are typically friends themselves. Thus, the number of triangles can help us spot abnormal graphs and abnormal nodes (see, e.g. [4]).

More-than-expected number of triangles also appear in biological networks,such as protein-protein interaction networks (see, e.g [28]).

A very recent work ([4]) shows that the distribution of the local number of triangles can be used to create successful spam filters and also provide useful features to assess content quality in social networks. In [11] the distribution of triangles is used to uncover hidden thematic structure in the World Wide Web. Therefore, counting triangles is a significant problem in graph mining, with several important applications.

The asymptotically fastest existing methods (lowest time complexity) suffer from space complexity. Specifically, they have $\Theta(n^2)$ space complexity, where $n$ is

the number of nodes in the network. For large or huge networks this is prohibitive. Therefore, in practice it is preferred to list the triangles ([20]). Other approaches, instead of counting exactly the triangles, adopt the streaming model ([3],[5]) or even more recently a semi-streaming model([4]).

The main contribution of this paper is the EIGENTRIANGLE algorithm, based on Theorem 3.1 saying that the number of triangles is exactly one sixth of the sum of cubes of eigenvalues and the properties of "real-world" network spectra. This is a completely novel view point, which opens the door to the vast machinery of eigenvalue algorithms and fine-tunings. Eigenvalues can be easily computed for sparse graphs, and can be applied on a map/reduce ('hadoop') architecture, which is extremely promising for Peta-byte scale graphs.

The additional contributions are the following

1. **Fast total triangle count** An algorithm for the fast estimation of the number of triangles, with excellent accuracy: Figure 1 shows the performance of our algorithm for a web graph (Wiped, Nov. '06) with approximately $\approx 3.1M$ nodes and $\approx 37M$ edges. We achieve about 1000x faster performance respectively than a straightforward, exact-counting competitor with more than 97% accuracy.

2. **Fast local triangle count** A theorem and an algorithm for the fast estimation of local triangle count, that is, the number of triangles $\Delta_i$ that the $i$-Th node participates in. Again, the speedups and the accuracy are excellent, as we show in section 4.

3. **Extensive experimentation** We used almost 160 real-world data sets; the speed-ups were between 34x to 1075x,for accuracy at least 95%.

4. **Laws**: New power laws in real networks with surprising properties.

The rest of the paper is organized as follows: Section 2, surveys earlier triangle-counting methods. In Section 3 we present the EIGENTRIANGLE and EIGENTRIANGLELOCAL theorems and algorithms, for global and local triangle counting, respectively. Section 4 gives the experimental results on several real data sets. Section 5 lists some surprising laws that govern the count of triangles in real graphs. In Section 6 we present some theoretical ramifications of the previous sections and we conclude in Section 7.

## 2 Related work

Let $G(V, E)$, n=$|V|$, m=$|E|$ be an undirected graph without self-edges. A triangle is a set of three fully connected nodes. In this section we briefly review the state-of-the-art work related to the problems of global and local triangle counting. By global we refer to the problem of counting the total number of triangles in $G$ and by local to the problem of counting the number of triangles per each node. Two other problems related to triangles are *(i)* deciding whether $G$ contains a triangle and *(ii)* for each triangle in $G$, list the participating nodes. Before we make the overview, we state a few facts about the spectrum of a graph.

**Eigenvalues** Depending on whether the graph is represented as an adjacency or as a Laplacian matrix ([7]), the eigenvalues receive different meaning: in the former case, they indicate the path capacity of the graph ([18]) whereas in the latter the connectivity of the graph ([7]). A classical method for finding the eigenvalues of a matrix is the QR method ([17]). A huge literature, which is impossible to list here, exists for the eigenvalue problem.

**Non-streaming algorithms** The brute-force approach enumerates all possible triples of nodes ($O(n^3)$). The algorithms with the lowest time complexity for counting triangles rely on fast matrix multiplication. The asymptotically fastest algorithm to date is $O(n^{2.376})$ [8]. In [2], an algorithm of $O(m^{\frac{2\omega}{\omega+1}}) \subset O(m^{1.41})$ time complexity and of $\Theta(n^2)$ space complexity is proposed to find and count triangles in a graph. However, these methods suffer from $\Theta(n^2)$ space complexity. Listing methods ([26]) are preferred against matrix-based methods. Even if these methods solve problem *(ii)* which is more general than the global and local triangle counting, they are more efficient. Two straightforward listing methods are the *Node Iterator* and the *Edge Iterator* algorithms. The *Node Iterator* considers each one of the $n$ nodes and examines which pairs of its neighbors are connected. The time complexity of the *Node Iterator* is $nd_{max}^2$. This is a significant improvement over the brute-force approach when the graph is sparse. The *Edge Iterator* algorithm computes for each edge the number of triangles that contain it. The time complexity of this algorithm is $O(\sum_{v \in V} d_v^2)$. Asymptotically, both methods have the same time complexity ([26]). In [2], a listing algorithm of time complexity $\Theta(m^{\frac{3}{2}})$ is proposed. However, the space complexity is $\Theta(n^2)$. In [26] the $forward$ algorithm is proposed, which is an improvement of the *Edge Iterator* algorithm, with running time $\Theta(m^{\frac{3}{2}})$. In

[20], a further improvement of the $forward$ algorithm is proposed, called the *compact-forward* algorithm.

**Streaming algorithms** The memory restrictions when dealing with huge graphs lead us to the streaming approach. In the streaming model, the goal is to find a randomized algorithm that outputs an $\epsilon$-approximation of the number of triangles with probability at least 1-$\delta$ with one pass access to the graph data stream. The main advantage of this approach in comparison to the non-streaming scenario is that it requires a single pass over the data. Representative work on the streaming case are [3] and [5]. In [3], rigorous theory supports the algorithms making them attractive for practical applications. Still, there are open issues according to the authors such as justifying when their adjacency stream model is superior to the naive sampling method.In [5], accuracy in certain cases can be an issue. Recently, the semi-streaming model was introduced by [4] to solve the local counting problem. This model relaxes the strict restriction of the single pass over the data, and instead it uses an amount of main memory ($O(n)$) and performs $O(log(n))$ sequential scans over the edge file. The authors do not make a comparison to existing approaches and report overflow problems in the implementation as the number of passes increases.

## 3 Proposed Method

The goal of this work is to propose a new method for counting triangles approximately in large, real-world networks while being accurate, fast, and *easily* parallelizable. The last goal is also of great importance, since it will provide a way to mine huge graphs using parallel architectures such as the map/reduce ('hadoop') [10]. Furthermore,if all these goals are met at once, the "trade-off" described in section 2 will be destroyed. The method we propose achieves all the above characteristics when the graph has some special properties. Real-world networks appear to have them very frequently.

Table 1 gives a list of symbols and their definitions.

### 3.1 Theorems and proofs

Using simple linear algebra arguments, we prove two theorems on the top of which our methods are built.

**Theorem 3.1 (EigenTriangle)** *The total number of triangles in a graph is proportional to the sum of cubes of eigenvalues, namely:*

$$\Delta(G) = \frac{1}{6}\sum_{i=1}^{n} \lambda_i^3 \quad (1)$$

| Sym. | Definition |
|---|---|
| $G$ | Undirected graph (no self-edges) |
| $d_{max}$ | maximum node degree |
| $\Delta$ | total number of triangles |
| $\Delta'$ | EigenTriangle's estimation of $\Delta$ |
| $\Delta_{avg}^{d_m}$ | average number of triangles over all nodes with degree $d_m$ |
| $\vec{\Delta}(G) = [\Delta_i]_{i=1..n}$ | $\Delta_i$ number of triangles node i participates |
| $\vec{\Delta}'(G) = [\Delta'_i]_{i=1..n}$ | $\Delta'_i$ EigenTriangleLocal's estimation of $\Delta_i$ |
| $m$, $n$ | Number of edges and nodes. |
| $[n] = (1..n)$ | Node ids |
| $\mathbf{C}'$ ($\vec{x}'$) | transpose of matrix $\mathbf{C}$ (vector $\vec{x}$) |
| $\mathbf{A}$ | Adjacency matrix |
| $\lambda_i$ | top-$i$-th eigenvalue (absolute value) |
| $\vec{u_i}=[u_{ij}]'_{j=1..n}$ | $i$-th top eigenvector (eigenvector corresponding to $\lambda_i$) |
| $\vec{\Lambda_k} = [\lambda_i]_{i=1..k}$ | $k$ top eigenvalues |
| $\mathbf{U_k} = [\vec{u_i}]_{i=1..k}$ | $k$ top eigenvectors |
| TPPL | triangles per node power law |
| DTPL | degree triangle power law |

**Table 1. Definitions of symbols and acronyms**

**Proof** The diagonal element $\alpha_{ii}$ of the square matrix $\mathbf{A}^3$ contains the number of paths of length 3 that begin and end at the same node $i$. The only way this can happen is to have a triangle in which node $i$ participates. Therefore the trace of $\mathbf{A}^3$ is three times the total number of triangles (since we are triple counting them because each triangle has 3 participating nodes). Furthermore, since the graph is undirected and we are counting each triangle as two (triangle $ikj$ is counted as $i \to k \to j$ and $i \to j \to k$). Therefore the following equality holds: $\Delta(G) = \frac{1}{6}trace(\mathbf{A}^3)$. Furthermore, if $\lambda$ is an eigenvalue of $\mathbf{A}$ then $\lambda^k$ is an eigenvalue of $A^k$ ($k \geq 1$). Finally, we know that $\sum_{i=1}^{n} \lambda_i = trace(\mathbf{A})$. Combining the above equations, we get that $\Delta(G) = \frac{1}{6}\sum_{i=1}^{n} \lambda_i^3$. *Q.E.D*

In case the graph $G$ is directed, theorem 3.1 still holds but with a slight modification: as it can be easily seen by the proof instead of multiplying the sum of the right side with $\frac{1}{6}$ we multiply by $\frac{1}{3}$. In this case $\Delta(G)$ is the total number of undirected triangles.

**Theorem 3.2 (EigenTriangleLocal)** *The number of triangles $\Delta_i$ that node i participates in, can be computed from the cubes of the eigenvalues of the*

**Algorithm 1** The EIGENTRIANGLE algorithm

**Require:** Adjacency matrix $A$ ($nxn$)
**Require:** Tolerance $tol$
**Output:** $\Delta'(G)$ global triangle estimation
  $\lambda_1 \leftarrow LanczosMethod(A, 1)$
  $\vec{\Lambda} \leftarrow [\lambda_1]$
  $i \leftarrow 2$ {initialize $i$, $\vec{\Lambda}$}
  **repeat**
    $\lambda_i \leftarrow LanczosMethod(A, i)$
    $\vec{\Lambda} \leftarrow \left[\vec{\Lambda}\ \lambda_i\right]$
    $i \leftarrow i + 1$
  **until** $0 \leq \frac{|\lambda_i^3|}{\sum_{j=1}^{i} \lambda_j^3} \leq tol$
  $\Delta'(G) \leftarrow \frac{1}{6} \sum_{j=1}^{i} \lambda_j^3$
  **return** $\Delta'(G)$

*adjacency matrix*

$$\Delta_i = \frac{\sum_j \lambda_j^3 u_{i,j}^2}{2} \tag{2}$$

*where $u_{i,j}$ is the $j$-th entry of the $i$-th eigenvector.*

**Proof** Easy extension of 3.1. It follows from the facts that since $\mathbf{A}_{nxn}$ is symmetric, $\mathbf{A} = \mathbf{U_n \Sigma U_n'}$, where $\mathbf{\Sigma}$ is a diagonal matrix with $diag(\mathbf{\Sigma}) = \vec{\Lambda}_n$ (all eigenvalues are real and $\mathbf{U_n}$ is an orthogonal matrix and therefore $\mathbf{A}^3 = \mathbf{U_n \Sigma^3 U_n'}$ according to [25]) and that each triangle is counted twice. *Q.E.D*

### 3.2 Proposed algorithms

We can see the pseudocode of the EIGENTRIANGLE and EIGENTRIANGLELOCAL algorithms. Both take only a tolerance parameter: *tol*. The intuition behind the tolerance parameter is to stop looping when the smallest eigenvalue contributes very little to the total number of triangles.

Both algorithms use the subroutine $LanczosMethod$ as a black box[1]. The Lanczos method is a well studied projection method for solving the symmetric eigenvalue problem using Krylov subspaces. Our choice is due to the following reasons: a) It is based only on matrix-vector products, which are easy to parallelize. b) "..with minimal memory requirements very large problems can be handled on not very large computers, and huge problems can be handled on large computers" (quote from [9]). c) High quality software is available (ARPACK,Parallel ARPACK etc.). More details about the Lanczos method can be found in [17], [19].

[1]For simplifying presentation, depending on the number of output arguments, Lanczos returns either $\lambda_i$ only or $\vec{u_i}$ too. The required time is the same in both cases.

**Algorithm 2** The EIGENTRIANGLELOCAL algorithm

**Require:** Adjacency matrix $A$ ($nxn$)
**Require:** Tolerance $tol$
**Output:** $\vec{\Delta}'(G)$ per node triangle estimation
  $\langle \lambda_1, \vec{u_1} \rangle \leftarrow LanczosMethod(A, 1)$
  $\vec{\Lambda} \leftarrow [\lambda_1]$
  $\mathbf{U} \leftarrow [\vec{u_1}]$
  $i \leftarrow 2$
  {initialize $i$, $\vec{\Lambda}, \mathbf{U}$}
  **repeat**
    $\langle \lambda_i, \vec{u_i} \rangle \leftarrow LanczosMethod(A, i)$
    $\vec{\Lambda} \leftarrow \left[\vec{\Lambda}\ \lambda_i\right]$
    $\mathbf{U} \leftarrow [\mathbf{U}\ \vec{u_i}]$
    $i \leftarrow i + 1$
  **until** $0 \leq \frac{|\lambda_i^3|}{\sum_{j=1}^{i} \lambda_j^3} \leq tol$
  **for** $j = 1$ to $n$ **do**
    $\Delta'_j = \frac{\sum_{k=1}^{i} u_{jk}^2 \lambda_k^3}{2}$
  **end for**
  $\vec{\Delta}'(G) \leftarrow [\Delta'_1, .., \Delta'_n]$
  **return** $\vec{\Delta}'(G)$

The idea of both algorithms could not be more simple and clear: *find the diagonal of a low rank approximation of* $\mathbf{A^3}$.

### 3.3 Why so successful?

Real-world networks have several special properties, such as small-worldness, scale-freeness and self-similarity characteristics. Two among the many special properties are the reason that our EIGENTRIANGLE and EIGENTRIANGLELOCAL algorithms achieve excellent accuracy, and excellent speedup at the same time:

- (a) The absolute values of their eigenvalues are skewed, typically following a power law ([13],[24],[6]).
- (b) Moreover, the signs of the eigenvalues tend to alternate ([14])) and thus their cubes roughly cancel out.

The combination of the two properties means that the first top strongest eigenvalues (experimentally 1-25, 3(a) lead to an excellent approximation.

Figure 2 shows the typical spectrum of a real-world network. It plots the rank of the eigenvalue vs. its value for the Political Blogs network ([1]). The two crucial properties described above are verified.

#### 3.3.1 Justifying the convergence speed of the Lanczos method

Lanczos algorithm can run in general into convergence problems. However, in the experiments we conducted, we never faced this problem. This interesting phenomenon happens because real-world networks have usually have a big spectral gap, which makes Lanczos robust. In more detail, we use the Kaniel-Paige convergence theory and the special properties of real-world networks. In particular, in [12] it is claimed that real-world networks have a big gap $\lambda_1$ - $|\lambda_2|$ [2]. This claim was experimentally verified and is in accordance with [13],[24],[6]. Also, according to Kaniel-Paige convergence theory ([17]) if $\theta_1 \geq ... \geq \theta_k$ are the eigenvalues of the tridiagonal $T_k$ (a small matrix internally constructed by Lanczos) obtained after $k$ steps of the Lanczos iteration, then the following inequality holds:

$$\lambda_1 \geq \theta_1 \geq \lambda_1 - \frac{(\lambda_1 - \lambda_n)tan(\phi_1)^2}{(c_{k-1}(1+2\rho_1))^2} \quad (3)$$

where $cos(\phi_1) = |\vec{q_1}'\vec{u_1}|$ (where $\vec{q_1}$ is the first Lanczos vector), $\rho_1 = \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}$, and $c_{k-1}$ is the Chebyshev polynomial of degree $k-1$. Therefore, in our case $\rho_1$ is larger than zero and since Chebyshev polynomials grow very fast outside [-1,1] ([23]), Lanczos converges very fast.

## 4 Experimental Results

We do experiments to answer the following questions: for at least 95% accuracy what are the speedups we can achieve for the triangle counting problem?

First we give the experimental setup, and then the results.

### 4.1 Experimental set up

Each graph was preprocessed by removing any self-edges, the direction of the edges and the weights whenever needed. The number of nodes and edges of the networks used after the preprocessing are summarized in table 2. [3] As a competitor we chose the *Node Iterator* (see section 2) since it is much superior to the naive $O(n^3)$, easy to implement and has asymptotically the same time and space complexity with the *Edge iterator*. We ran the experiments in a machine with a quad-processor Intel Xeon 3GHz with 16GB of RAM. We express the experimental results as the ratio of the clockwork times of the *Node Iterator* to the EigenTriangle (speedup). Our algorithms were implemented in MATLAB and the *Node Iterator* in JAVA.

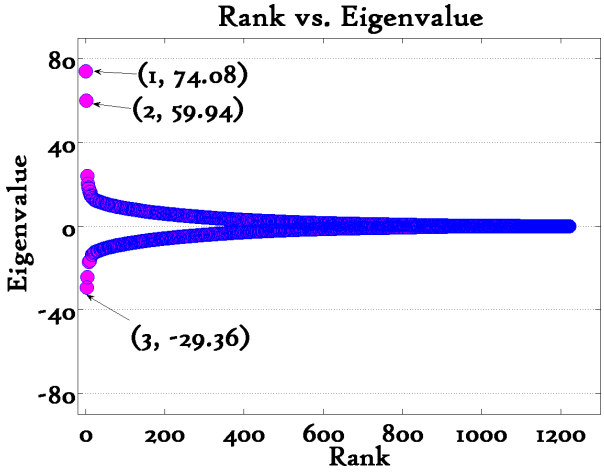



**Figure 2. Typical spectrum of a real-world network (Polblogs dataset). Value $\lambda_i$ versus rank $i$ (highest absolute value first). Notice that (1) the first few eigenvalues are much stronger than the rest, (2) which are almost symmetric around zero and (3) cubing amplifies these effects.**

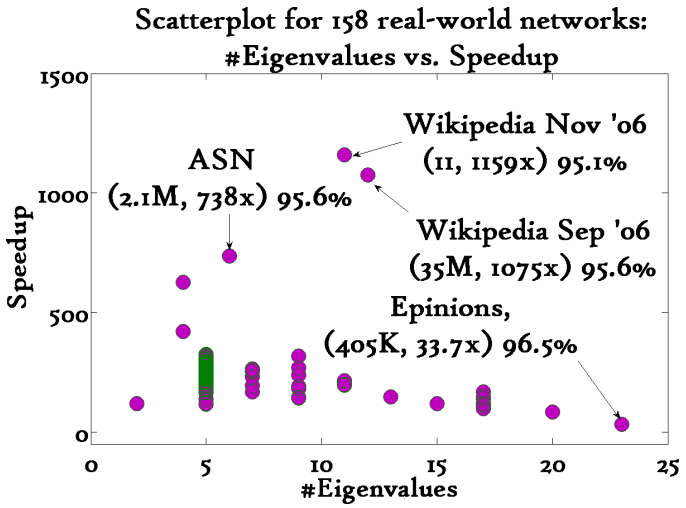
### 4.2 Global Triangle Count

The results of applying the EigenTriangle algorithm are summarized in figure 3. Figure 3(a) plots the number of eigenvalues required to get at least 95% versus the achieved speedup, whereas figure 3(b) the number of edges in the graph versus the speedup. The following facts are remarkable:

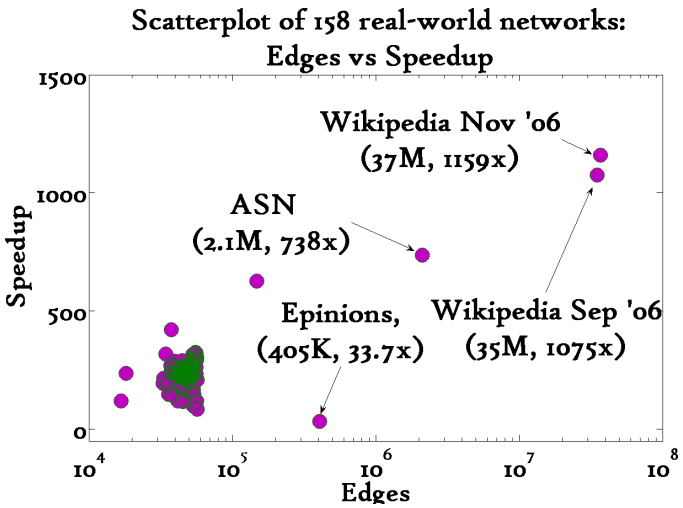
- The mean value of eigenvalues needed to achieve more than 95% is 6.2 with standard deviation 3.2. The mean speedup is 250x with standard deviation 123. The maximum speedup is 1159x whereas the minimum 33.7x.
- The speedup savings appear to increase as the size of the network grows. A possible explanation for this, assuming that our degree distribution follows approximately a power law, could be that as the network grows, the maximum degrees are getting more detached from the rest and according to [24], so do the top eigenvalues. Therefore, with a handful of eigenvalues, we get extremely high accuracy.

[2] Absolute value is not needed for $\lambda_1$ because according to the Perron-Frobenius theorem ([16]) it is always positive

[3] Most of the datasets we used are publicly available. Indicative sources are : `http://arxiv.org`, `http://www.cise.ufl.edu/research/sparse/mat/`, `http://www-personal.umich.edu/~mejn/netdata/`




(a) #Eigenvalues vs. Speedup




(b) Edges vs. Speedup

**Figure 3. Scatterplots of the results for 158 graphs. (a) Speedup vs. Eigenvalues: The mean required approximation rank for $\geq$ 95% accuracy is 6.2. Speedups are between 33.7x and 1159x, with mean 250.(b) Speedup vs. Edges: Notice the trend of increasing speedup as the network size grows (#edges).**

Finishing this section, we provide the following "rule of thumb": Follow the default *tol*=0.05 which gave the results reported here. Our experiments showed little sensitivity on the choice of *tol*. Alternatively an even easier rule of thumb is to pick the top 30 eigenvalues (since we got all our results with less than 25 eigenvalues, see figure 3).

### 4.3 Local Triangle Count

To measure of the performance of the EigenTriangleLocal algorithm, we use Pearson's correlation coefficient $\rho$ and the relative reconstruction error (as in [4]).

$$RRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\Delta_i - \Delta_i'|}{\Delta_i}$$

In figure 4 we see how well $\vec{\Delta}'(G)$ approximates $\vec{\Delta}(G)$ with the top 10 eigenvalues and eigenvectors. The RRE we obtain is $7 * 10^{-4}$ and $\rho$ almost 1.

Figure 5 explains why our proposed methods work well in practice. It plots the rank of the approximation vs. $\rho$. We observe that after the second rank approximation, for all three networks the approximation is excellent: $\rho$ is greater than 99.9% whereas the RRE has always order of magnitude between $10^{-7}$ and $10^{-4}$. Similar results hold for the rest of the datasets we experimented with.

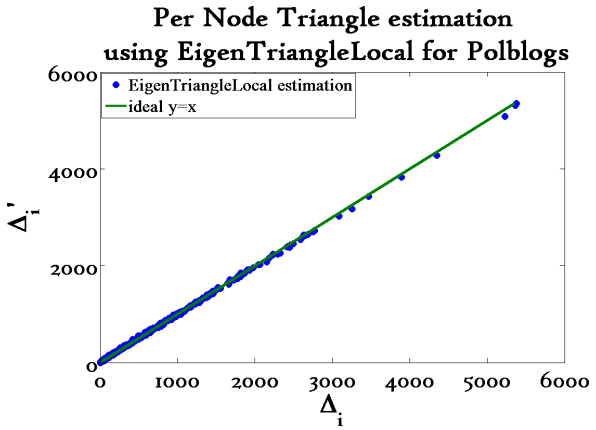



**Figure 4. Scatterplot of $\Delta_i'$ (estimated #triangles of node $i$) vs. $\Delta_i$ (actual number) for Polblogs using a rank 10 approximation. Relative reconstruction error is $7 * 10^{-4}$ and the Pearson's correlation coefficient is 99.97%.**

## 5 Laws and patterns

### 5.1 TrianglesParticipation law

Figure 6 shows the PDF of the number of triangles that a node participates in. That is, for a given graph, it plots the number of triangles (x-axis) versus

| Nodes | Edges | Description |
|---|---|---|
| **Social Networks** | | |
| 75,877 | 405,740 | Epinions network |
| 404,733 | 2,110,078 | Anonymous Social Network (ASN) |
| **Co-authorship networks** | | |
| 27,240 | 341,923 | Arxiv Hep-Th |
| **Information networks** | | |
| 1,222 | 16,714 | Political blogs |
| 13,332 | 148,038 | Reuters news, Sept 9-11,2001. |
| **Web graphs** | | |
| 2,983,494 | 35,048,116 | Wikipedia 2006-Sep-25 |
| 3,148,440 | 37,043,458 | Wikipedia 2006-Nov-04 |
| **Internet networks** | | |
| 13,579 | 37,448 | AS Oregon |
| 23,389 | 47,448 | CAIDA AS 2004 to 2008 (means over 151 timestamps) |

**Table 2. Summary of real-world networks used.**

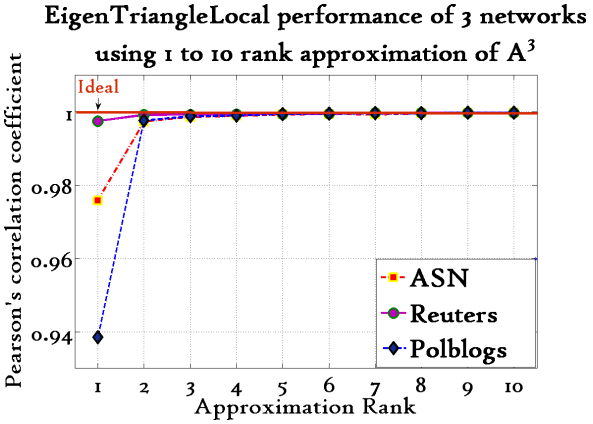



**Figure 5. Local triangle reconstruction for three real-world networks using rank 1 to 10 approximation of the diagonal of $A^3$. Pearson's correlation coefficient $\rho$ vs. approximation rank.Notice that after rank 2 $\rho$ is greater than 99.9% for all three networks.**

the count of nodes participating in that many triangles. Both scales are logarithmic. We show the results only for three of the datasets for brevity (Epinions, Anonymous social network and HEP-TH) , because the rest have similar behavior. The over-arching observation is that the number of participating triangles follows either a power law, or a lognormal-like distribution, with a power-law tail. The important point is that generating these plots can be from 30x to 1000x faster with our proposed algorithms with high accuracy.

### 5.2 DEGREE-TRIANGLE law

Is there a correlation between the $i$-th largest degree $d_i$, and the number of triangles? This is the focus of our exploration. The results are surprising, and shown in Figure 7. The Figure plots the degree $d_i$ vs. the mean number of triangles over all nodes with degree $d_i$ for the specified networks (the rest of the networks we used had similar behavior and are omitted for brevity). The Figure also has insets, showing the degree distribution (PDF), in log-log scales. We performed least square fitting.

We have the following observations from there.

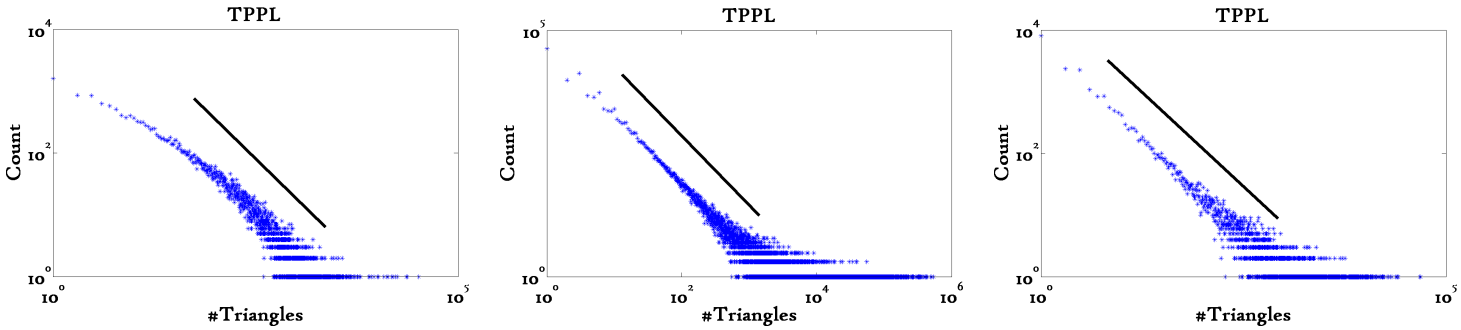
- DEGREE-TRIANGLE power law: $\Delta_{avg}^{d_i}$ (see table 1 for notation) follows a power-law with respect to the degree $d_i$.
- **Slope-complement:** it is surprising (at least to us) that the slope $\tau$ of the DEGREE-TRIANGLE power law is extremely close to the negative of the slope of the degree distribution, whenever the latter follows a power law (figure 7(a),(b) and (d)) or lognormal-like distribution(figure 7(c)). For the later, we performed a second least squares fitting, by fixing the slope of the fitted line to be the complementary of the slope of the degree distribution's fitted line. The result would have occured if we had done a manual-visual fitting.
- **High-degree deviation**: high degree nodes tend to deviate from the earlier slope. This is probably due to the phenomenon that high-degree nodes have a lot of degree-1 nodes, which, obviously, do not contribute to triangles.

## 6 Theoretical Ramifications

### 6.1 Kronecker graphs

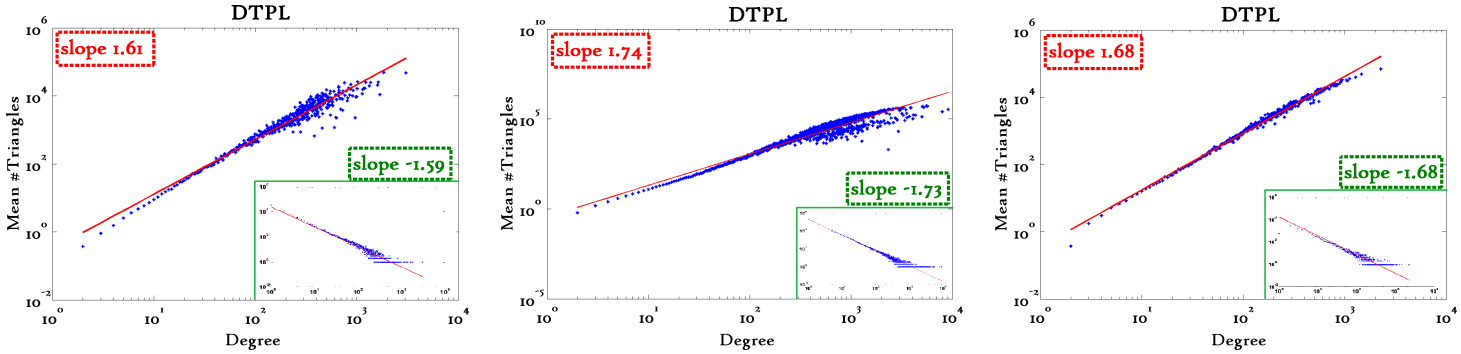
Kronecker graphs ([21]) have attracted recent interest, because they can be made to mimick real graphs well ([22]). Here we give a closed formula that estimates the number of triangles for a Kronecker graph. Some definitions first:

Let $\mathbf{A}$ be the $nxn$ adjacency matrix of an $n$-node graph $G_A$ with $\Delta(G_A)$ triangles, and let $\mathbf{B} = \mathbf{A}^{[\mathbf{k}]}$ be the $k$-th Kronecker power of it, that is, an $n^k \times n^k$ adjacency matrix (see [21] for the exact definition of




(a) **HEP-TH** (b) **ASN** (c) **Epinions**

**Figure 6. PDF of participating triangles: Figure plots the count of nodes with $\Delta$ triangles vs. $\Delta$ in log-log scale. We observe power laws or power law tails in the PDFs .**




(a) **Epinions** (b) **ASN** (c) **Reuters**

**Figure 7. Figure plots degree $\Delta_{avg}^{d_i}$, the mean number of triangles over all nodes with degree $d_i$ vs. $d_i$ for (a) Epinions (b) ASN (c) HEP-th and (d) Reuters networks. Two surprising observations: *(i)* A power law emerges. *(ii)* The slope of the fitted line is the complementary of the slope of the fitted line of the degree distribution in the insets (least squares fitting).**

the Kronecker matrix multiplication). Let $G_B$ denote the corresponding graph.

Let $\vec{\lambda} = (\lambda_1, .., \lambda_n)$ be the eigenvalues of matrix $\mathbf{A}$. Then we have:

**Theorem 6.1 (KroneckerTRC)** *The number of triangles $\Delta(G_B)$ of $G_B$ can be computed from the $n$ eigenvalues of $\mathbf{A}$:*

$$\Delta(G_B) = 6^{2^k-1}\Delta(G_A)^{2^k} = 6^{2^k-1}(\frac{\sum_{i=1}^{n}\lambda_i^3}{6})^{2^k}, k \geq 0. \quad (4)$$

**Proof** We use induction on the depth of the recursion $k$. For $k = 0$, KroneckerTRC trivially holds (see thrm. 3.1). So the base case is true. Let KroneckerTRC hold for some $r \geq 1$. For notation simplicity, let $\mathbf{C} = \mathbf{A}^{[r]}$ with eigenvalues $[\gamma_i]_{i=1..s}$ and $\mathbf{D} = \mathbf{A}^{[r+1]}$.

According to the induction assumption:

$$\delta(G_C) = 6^{2^r-1}\left(\frac{\sum_{i=1}^{n}\lambda_i^3}{6}\right)^{2^r}$$

Now, we will show that KroneckerTRC holds for $r+1$. The eigenvalues $\psi_1, .., \psi_{s^2}$ of $\mathbf{D}$ are the terms of the sum $(\sum_{i=1}^{s}\gamma_i)^2$. We see now that $\sum_{i=1}^{s^2}\psi_i^3 = (\sum_{i=1}^{s}\gamma_i^3)^2 \Leftrightarrow \delta(G_D)=6\delta(G_C)^2 \Leftrightarrow \delta(G_D)=6^{2^{r+1}-1}(\frac{\sum_{i=1}^{n}\lambda_i^3}{6})^{2^{r+1}}$. So, KroneckerTRC holds for all $k \geq 0$. The expression can be further simplified: $\Delta(G_B) = \frac{(\sum_{i=1}^{n}\lambda_i^3)^{2^k}}{6}$. *Q.E.D*

This results in tremendous speed savings, and *perfect* accuracy.

**Timing results, and stochastic Kronecker graphs** Experimenting on a small deterministic Kronecker graph with 6,561 nodes and 839,808 edges coming from the 3-clique initiator with depth of recursion equal to 3, we get $10^6$ faster performance. As the size of the Kronecker graph increases, we obtain arbitrarily huge speedups.

What is interesting is that the KroneckerTRC theorem also leads to fast estimation of triangles, even for stochastic Kronecker graphs (see [22] for the definitions). Stochastic Kronecker graphs have been shown to mimick real graphs very well. Intuitively, a stochastic Kronecker graph is like a deterministic one, with a few random edge deletions and additions. Our experiments with a stochastic Kronecker graphs show that these random edge manipulations have little effect on the accuracy. Specifically, our experiments with $n$=6,561 and $m$=2,202,808[4], show that we obtain $1.5 * 10^6$x faster execution, while maintaining 99.34% accuracy. Similar results hold for other experiments we conducted as well. Proving bounds for the accuracy for stochastic Kronecker graphs is an interesting research direction.

### 6.2 Erdős-Rényi graphs

It is interesting to notice that our algorithm is guaranteed to give high accuracy and speedup performance for random Erdős-Rényi graphs. This is due to the so-called Wigner's semi-circle law for all but the first eigenvalue [15]. For example, for a graph with $n = 20,000$ and $p = 0.6$, using EigenTriangleLocal with 0.05 tolerance parameter, we get 1600 faster performance compared to the *Node Iterator* with relative error $5 * 10^{-5}$ and Pearson's correlation coefficient almost equal to 1[5].

## 7 Conclusions

The main contribution of this work is the EigenTriangle algorithm. It uses a link between the number of triangles and the eigenvalues (Theorem 3.1) of the adjacency matrix and the observation that just the top eigenvalues contribute significantly to the total number of the triangles. This is a major observation opening the door to the vast machinery of readily available, highly fine-tuned eigenvalue algorithms and implementations. These algorithms are not only fast, but, most of them have been parallelized, or can be easily parallelized on the emerging map/reduce ('hadoop') architecture. Thus, our method can be trivially applied on huge, peta-byte scale graphs, as long as there is an eigenvalue implementation available, like Lanczos.

The main contributions of this work are the following:

- We give the EigenTriangle algorithm, which gives excellent accuracy, for huge speedups: over 95% accuracy, for 30x to 1000x speedups, for all the 158 real networks we tried.
- We give the EigenTriangleLocal algorithm based on Theorem 3.2, which can quickly estimate the number of triangles that a given node participates in. Again, the accuracy/speedup results are excellent, for all the datasets we tried.
- Both algorithms, as Figure 3(b) implies, appear to have a trend of increasing speedup savings as the network size grows. Furthermore in all datasets we experimented with, 30 eigenvalues are always enough to obtain high accuracy, no matter the size of the network. Figure 3(a) indicates strongly this fact.
- We were able to discover two new laws with certain surprising properties (at least to us): the TriangleParticipation and the Degree-Triangle laws. Thanks to our fast triangle-counting methods these laws can be found fast with high accuracy.

A very promising direction is mining huge graphs, using a map/reduce ('hadoop') architecture ([10]). Our algorithms are steps towards this direction.

[4]Initiator matrix (using MATLAB's notation): [.99 .9 .9;.9 .99 .1;.9 .1 .99],depth of recursion: 3

[5]It makes no sense to apply EigenTriangle on Erdős-Rényi since the total number of triangles is approximately $\binom{n}{3}p^3$.

## References


[1] L. A. Adamic and N. Glance. The political blogosphere and the 2004 u.s. election: divided they blog. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, New York, NY, USA, 2005. ACM.

[2] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

[3] Z. Bar-Yosseff, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 623–632, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[4] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of ACM KDD*, Las Vegas, NV, USA, August 2008.

[5] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. Counting triangles in data streams. In *PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 253–262, New York, NY, USA, 2006. ACM.

[6] F. Chung, L. Lu, and V. Vu. Eigenvalues of random power law graphs. *Annals of Combinatorics*, 7(1):21–33, June 2003.

[7] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society.

[8] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 1–6, New York, NY, USA, 1987. ACM.

[9] J. K. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. 1*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[10] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *OSDI '04*, pages 137–150, December 2004.

[11] J.-P. Eckmann and E. Moses. Curvature of co-links uncovers hidden thematic layers in the world wide web. *PNAS*, 99(9):5825–5829, April 2002.

[12] E. Estrada. Spectral scaling and good expansion properties in complex networks. *Europhysics Letters*, 73:649–655, 2006.

[13] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[14] I. J. Farkas, I. Derenyi, A.-L. Barabasi, and T. Vicsek. Spectra of "real-world" graphs: Beyond the semi-circle law. *Physical Review E*, 64:1, 2001.

[15] Z. Füredi and J. Komlós. The eigenvalues of random symmetric matrices. *Combinatorica*, 1(3):233–241, 1981.

[16] R. G. Godsil C.D. *Algebraic Graph Theory*. Springer, 2001.

[17] G. Golub and C. Van Loan. *Matrix Computations*. JohnsHopkinsPress, Baltimore, MD, second edition, 1989.

[18] S. A. J. Harary F. The spectral approach to determining the number of walks in a graph. *Pacific J. Math.*, 80:443–449, 1979.

[19] D. J. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[20] M. Latapy. Practical algorithms for triangle computations in very large (sparse (power-law)) graphs. Submitted, 2007.

[21] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. *Knowledge Discovery in Databases: PKDD 2005*, pages 133–145, 2005.

[22] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 497–504, New York, NY, USA, 2007. ACM.

[23] C. D. Mason J.C. *Chebyshev Polynomials*. CRC Press, 2002.

[24] M. Mihail and C. Papadimitriou. the eigenvalue power law, 2002.

[25] G. Strang. *Introduction to Linear Algebra*. SIAM, Philadelphia, PA, 2003.

[26] Thomas Schank and Dorothea Wagner . DELIS-TR-0043 - finding, counting and listing all triangles in large graphs, an experimental study. techreport 0043, submitted, 2004.

[27] S. Wasserman and K. Faust. *Social network analysis*. Cambridge University Press, Cambridge, 1994.

[28] P. Ye, B. D. Peyser, F. A. Spencer, and J. S. Bader. Commensurate distances and similar motifs in genetic congruence and protein interaction networks in yeast. *BMC Bioinformatics*, 6:270, 2005.