Under consideration for publication in Knowledge and Information
Systems

Counting Triangles in Real-World Networks using Projections

Charalampos E. Tsourakakis

Machine Learning Department, SCS Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213-3891, USA

Abstract. Triangle counting is an important problem in graph mining. Two frequently used metrics in complex network analysis which require the count of triangles are the clustering coefficients and the transitivity ratio of the graph. Triangles have been used successfully in several real-world applications, such as detection of spamming activity, uncovering the hidden thematic structure of the web and link recommendation in online social networks. Furthermore, the count of triangles is a frequently used network statistic in exponential random graph models. However, counting the number of triangles in a graph is computationally expensive.

In this paper, we propose the EigenTriangle and EigenTriangleLocal algorithms to estimate the number of triangles in a graph. The efficiency of our algorithms is based on the special spectral properties of real-world networks, which allow us to approximate accurately the number of triangles. We verify the efficacy of our method experimentally in almost 160 experiments using several Web Graphs, social, co-authorship, information and Internet networks where we obtain significant speedups with respect to a straight-forward triangle counting algorithm.

Furthermore, we propose FastSVD, an algorithm which allows us to apply the core idea of the EigenTriangle algorithm on graphs which do not fit in the main memory. The main idea is a simple node sampling process according to which node i is selected with probability $\frac{d_i}{2m}$ where d_i is the degree of node i and m is the total number of edges in the graph. Our theoretical contributions also include a theorem which gives a closed formula for the number of triangles in Kronecker graphs, a model of networks which mimics several properties of real-world networks.

Keywords: Triangles; Network Analysis; SVD; Algorithms;

Received xxx
Revised xxx
Accepted xxx

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 (e.g., intrusion detection, spamming), in web applications (e.g., community detection, blog analysis), in social networks such as Facebook and LinkedIn (e.g., for link prediction) and many more. One of the operations of interest in such a setting is the estimation of the clustering coefficients and the transitivity ratio of the graph, which effectively translates in computing the number of triangles that each node participates in or the total number of triangles in the graph respectively. Furthermore, triangles are a frequently used network statistic in the exponential random graph model (Ove et al, 1986; Fienberg et al, 2009) and naturally appear in models of real-world network evolution (Leskovec et al, 2008). Furthermore, triangles have been used in several applications such as spam detection (Becchetti et al, 2008), uncovering the hidden thematic structure of the web (Eckmann et al, 2002) and for link recommendation in online social networks (Tsourakakis et al, 2009). It is worth noting that in social networks triangles have a natural interpretation: friends of friends are frequently friends themselves (Wasserman et al, 1994).

However, triangle counting is computationally expensive. In this paper, we propose the EigenTriangle and EigenTriangleLocal algorithms to compute the total number of triangles and the number of triangles that each node participates in respectively, in an undirected graph. Our algorithms work for any type of graph but they are effective when the graph possesses certain spectral properties. Real-world networks empirically exhibit such properties, making our algorithms a viable option for counting triangles therein. We verify this claim experimentally, by performing 160 experiments on different types of real-world networks (Web Graphs, social, co-authorship, information and Internet networks). We observe significant speedups, i.e., between 34× to 1075× faster performance, for accuracy at least 95% compared to a straight-forward counting algorithm.

We use Lanczos method to compute the low rank eigendecomposition, and we explain how the spectral properties of real-world networks allow Lanczos to converge fast. Viewing the adjacency representation of the graph as a set of n points in the n-dimensional Euclidean space $\mathbb{R}^n$ and observing that EigenTriangle performs an optimal (in the least squares sense) projection on a k-dimensional hyperplane, we show that at the cost of some accuracy fast SVD algorithms can be used instead to estimate the number of triangles. Finally we give two new laws related to triangles and a theorem providing a closed formula for the number of triangles in Kronecker graphs (Leskovec et al, 2005), a model for generating graphs which mimic properties of real-world networks.

The rest of the paper is organized as follows: Section 2, presents briefly existing triangle-counting methods and the Singular value Decomposition. In Section 3 we present the EigenTriangle and EigenTriangleLocal algorithms, for global and local triangle counting respectively and we explain why they are efficient. Section 4 presents the experimental results on several real data sets. In Section 5 we present a simple sampling algorithm which allows us to improve further the underlying idea of the EigenTriangle and several other theoretical ramifications. We conclude in Section 6.

2. Related work

In this section we briefly present previous work related to the triangle counting problem and basic background knowledge on the Singular Value Decomposition.

2.1. Counting Triangles

Let $G(V, E)$, n=$|V|$, m=$|E|$ be an undirected, unweighted, simple graph. 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 or not and *(ii)* for each triangle in G, list the participating nodes.

Exact Counting: The brute force approach enumerates all possible triples of nodes resulting in a naive algorithm of $O(n^3)$ time complexity. Using this naive algorithm we can list exactly the triangles in G. Other listing methods include the *Node Iterator* and the *Edge Iterator*. The *Node Iterator* considers each one of the n nodes and examines which pairs of its neighbors are connected. The *Edge Iterator* algorithm computes for each edge the number of triangles that contain it. Asymptotically, both methods have the same time complexity $O(\sum_{v \in V} d_v^2)$ (Schank et al, 2004), which in the case of a dense graph are eventually $O(n^3)$. For sparse graphs, these methods are significant improvements over the naive algorithm. In (Schank et al, 2004) the $forward$ algorithm is proposed, which is an improvement of the *Edge Iterator* algorithm, with running time $\Theta(m^{\frac{3}{2}})$. In (Latapy, 2008), a further improvement of the $forward$ algorithm is proposed, called the *compact-*$forward$ algorithm.

The algorithms with the lowest time complexity for counting triangles rely on fast matrix multiplication. The asymptotically fastest matrix multiplication algorithm to date is $O(n^{2.376})$ (Coppersmith et al, 1987). In (Alon et al, 1997) 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. In practice, listing methods (Schank et al, 2004) are preferred against matrix-based methods because of the prohibitive memory requirements of the latter.

Approximate Counting: In many applications such as the ones mentioned in Section 1 the exact number of triangles is not crucial. Thus approximating algorithms which are faster and output a high quality estimate are desirable. Most of the approximate triangle counting algorithms have been developed in the streaming setting. In this scenario, the graph is represented as a stream. Two main representations of a graph as a stream are the edge stream and the incidence stream. In the former, edges are arriving one at a time. In the latter scenario all edges incident to the same vertex appear successively in the stream. The ordering of the vertices is assumed to be arbitrary. A streaming algorithm produces a relative ϵ-approximation of the number of triangles with high probability, making a constant number of passes over the stream. However, sampling algorithms developed in the streaming literature can be applied in the setting where the graph fits in the memory as well.

Monte Carlo sampling techniques have been proposed to give a fast estimate of the number of triangles. According to such an approach, a.k.a. naive sampling, we choose three nodes at random repetitively and check if they form a triangle or not. If one makes

$$r = \log(\frac{1}{\delta})\frac{1}{\epsilon^2}(1+\frac{T_0+T_1+T_2}{T_3})$$

independent trials where $T_i = \#$triples with i edges and outputs as the estimate of triangles the random variable $T_3' = \binom{n}{3}\frac{\sum_{i=1}^{r} X_i}{r}$ then

$$(1-\epsilon)T_3 < T_3' < (1+\epsilon)T_3$$

with probability at least $1-\delta$. For graphs that have $T_3 = o(n^2)$ triangles this approach is not suitable. This is the typical case, when dealing with real-world networks. This sampling approach is presented in (Schank et al, 2005).

In the seminal paper (Bar-Yosseff et al, 2002) the authors reduce the problem of triangle counting efficiently to estimating moments for a stream of node triples. Then they use the Alon-Matias-Szegedy algorithms (Alon et al, 1996) (a.k.a. AMS algorithms) to proceed. Along the same lines, Buriol et al. in (Buriol et al, 2006) proposed two space-bounded sampling algorithms to estimate the number of triangles. Again, the underlying sampling procedures are simple. E.g., for the case of the edge stream representation, they sample randomly an edge and a node in the stream and check if they form a triangle. Their algorithms are the state-of-the-art algorithms to our knowledge. In their three-pass algorithm, in the first pass they count the number of edges, in the second pass they sample uniformly at random an edge (i,j) and a node $k \in V - \{i,j\}$ and in the third pass they test whether the edges $(i,k),(k,j)$ are present in the stream. The number of draws that have to be done in order to get concentration (of course these draws are done in parallel), is of the order

$$r = \log(\frac{1}{\delta})\frac{2}{\epsilon^2}(3+\frac{T_1+2T_2}{T_3})$$

Even if the term T_0 is missing compared to the naive sampling, the graph still has to be fairly dense with respect to the number of triangles in order to get an ϵ approximation with high probability. In (Becchetti et al, 2008) the semi-streaming model for counting triangles is introduced. The authors observed that since counting triangles reduces to computing the intersection of two sets, namely the induced neighborhoods of two adjacent nodes, ideas from the locality sensitivity hashing (Broder et al, 1998) are applicable to the problem of counting triangles. They relax the constraint of a constant number of passes over the edges, by allowing $\log n$ passes.

Doulion (Tsourakakis et al, KDD, 2009) proposed a new sampling procedure which is used in the Peta-Scale graph mining project (Kang et al, 2009). The approach of Doulion is the combinatorial perspective of the sparsification procedure proposed by (Achlioptas et al, 2001) and by (Tsourakakis, 2010) in the multilinear setting, which has been used to speed up spectral counting approach of (Tsourakakis, 2008) in (Tsourakakis et al, ASONAM , 2009). The algorithm tosses a coin independently for each edge with probability p to keep the edge and probability $q = 1-p$ to throw it away. In case the edge "survives", it gets reweighed with weight equal to $\frac{1}{p}$. Then, any triangle counting algorithm, such as the node- or edge- iterator, is used to count the number of triangles t' in G'.

The estimate of the algorithm is the random variable $T = \frac{t'}{p^3}$. The following facts -among others- were shown in (Tsourakakis et al, KDD, 2009):a) The estimator T is unbiased, i.e., $E[T] = t$ and the expected speedup when a simple exact counting algorithm as the node iterator is used, is $1/p^2$. The authors however did not answer the critical question, of how small can p be? Therefore (Tsourakakis et al, KDD, 2009) provides constant factor speedups leaving the question as a research topic. The answer concerning p was given recently in (Tsourakakis et al, Arxiv, 2009).

2.2. Singular Value Decomposition (SVD)

The Singular Value Decomposition (SVD) (Golub et al, 1989; **?**) is a powerful matrix decomposition frequently used for dimensionality reduction (Xiang et al, 2009; Song et al, 2009). SVD is widely used in problems involving least squares problems, linear systems and finding a low rank representation of a matrix. Furthermore, a wide range of applications uses SVD as its main algorithmic tool. Notable applications of the SVD are the HITS algorithm (Kleinberg, 1999), Latent Semantic Indexing (Deerwester et al, 1990; Papadimitriou et al, 1998), and image compression (Demmel, 1997).

The SVD theorem states that any matrix $A \in \mathbb{R}^{m \times n}$ can be written as a sum of rank one matrices, i.e., $A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$, where $u_i, i = 1 \ldots r$ (left singular vectors) and $v_i, i = 1 \ldots r$ (right singular vectors) are orthonormal and the singular values are ordered in decreasing order $\sigma_1 \geq \ldots \geq \sigma_r > 0$. Here r is the rank of A. We denote with A_k the k-rank approximation of A, i.e., $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$. Among all matrices $C \in \mathbb{R}^{m \times n}$ of rank at most k, A_k is the one that minimizes $||A - C||_F$.

An exhaustive listing of the work related to the SVD is impossible. We report here briefly the main result of (Drineas et al, 2004), since it is related to our work. Therein, a fast randomized algorithm is presented to approximate the SVD of a given matrix A. Specifically, the authors approximate the left singular vectors and the singular values of the SVD using an appropriately sampled set of columns of the matrix. Similarly, the right singular vectors can be approximated via a row sampling procedure. The probability of choosing a specific column $A^{(i)}$ is equal to $p_i = \frac{||A^{(i)}||^2}{||A||_F^2}$. They prove that their k-rank approximation $\hat{A}_k$ satisfies the following form of inequality with probability at least 1-δ when the sampling procedure picks c columns of A: $||A - \hat{A}_k||_F^2 \leq ||A - A_k||_F^2 + f(\delta, k, c)||A||_F^2$, where $f(\cdot)$ is a function of the three parameters k, c, δ as described in (Drineas et al, 2004).

3. Proposed Method

In this section we present the proposed algorithms for the triangle counting problem and explain why they are efficient when applied to a real-world network. Table 1 gives a list of symbols and their definitions.

Sym.	Definition
G	Undirected graph (no self-edges)
d_{max}	maximum node degree
Δ	total number of triangles
Δ'	EigenTriangle's estimation of Δ
$\vec{\Delta}(G) = [\Delta_i]_{i=1..n}$	Δ_i number of triangles node i participates
$\vec{\Delta}'(G) = \left[\Delta_i'\right]_{i=1..n}$	Δ_i' EigenTriangleLocal's estimation of Δ_i
m, n	Number of edges and nodes.
$[n] = (1..n)$	Node ids
A	Adjacency matrix
$A^{(i)}$	i-th column of A
λ_i	top-i-th eigenvalue (absolute value)
u_i	top-i-th eigenvector corresponding to eigenvalue λ_i
$\Lambda_k = [\lambda_i]_{i=1..k}$	vector containing k top eigenvalues
$U_k = [u_1\|\dots\|u_k]$	matrix containing the k top eigenvectors as its columns
$u_{i,j}$	the i-th entry of the j-th eigenvector

Table 1. Definitions of symbols used.

3.1. Theorems and proofs

The following theorem connects the number of triangles in which node i participates with the eigenvalues and eigenvectors of the adjacency matrix.

Theorem 3.1. Let G be an undirected, simple graph and A is adjacency matrix representation. The number of triangles Δ_i that node i participates in satisfies the following equation:

$$\Delta_i = \frac{\sum_j \lambda_j^3 u_{i,j}^2}{2} \tag{1}$$

where $u_{i,j}$ is the i-th entry of the j-th eigenvector and λ_j is the j-th eigenvalue of the adjacency matrix.

Proof. Since G is undirected, A is a real, symmetric matrix. Thus, by the spectral theorem we can diagonalize A using its eigenvalues and eigenvectors. Therefore $A = U\Lambda U^T$, where Λ is a diagonal matrix containing the eigenvalues of A and $U = [u_1|\dots|u_n]$ is the orthonormal matrix containing in its i-th column the eigenvector u_i corresponding to the i-th eigenvalue λ_i, $i = 1, \dots, n$. By the orthonormality of U, it follows that $A^3 = U\Lambda^3U^T$ $(\diamond)$.

Consider now α_{ii} the i-th diagonal element of A^3. α_{ii} is equal to twice (each triangle ijk is counted twice as $i \to j \to k \to i$ and $i \to k \to j \to i$) the number of closed walks of length three, i.e., the number of triangles in which node i participates. From equation $(\diamond)$ follows that $\alpha_{ii} = \sum_j \lambda_j^3 u_{i,j}^2$. Combining these two facts we obtain for equation 1. $\square$

The following lemma holds, see (Godsil et al, 2001; Tsourakakis, 2008):

Lemma 3.2. The total number of triangles $\Delta(G)$ in the graph is given by the

Algorithm 1 The EigenTriangle algorithm

```
Require: Adjacency matrix $A$ $(n \times n)$
Require: Tolerance $tol$
Output: $\Delta'(G)$ global triangle estimation
  $\lambda_1 \leftarrow LanczosMethod(A, 1)$
  $\vec{\Lambda} \leftarrow [\lambda_1]$
  $i \leftarrow 1$ {initialize $i$, $\vec{\Lambda}$}
  repeat
    $i \leftarrow i + 1$
    $\lambda_i \leftarrow LanczosMethod(A, i)$
    $\vec{\Lambda} \leftarrow \left[\vec{\Lambda}\ \lambda_i\right]$
  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 $\Delta'(G)$
```

sum of the cubes of the eigenvalues of the adjacency matrix divided by six, i.e.,:

$$\Delta(G) = \frac{1}{6} \sum_{i=1}^{n} \lambda_i^3 \qquad (2)$$

3.2. Proposed algorithms

We propose algorithms 1 and 2, the EigenTriangle and EigenTriangleLocal algorithms respectively. The former is based on Lemma 3.2, whereas the latter on Theorem 3.1. Both take as input the $n \times n$ adjacency matrix A and a tolerance parameter *tol*. EigenTriangle keeps computing eigenvalues until the contribution of the cube of the current eigenvalue is considered to be significantly smaller than the sum of the cubes of the previously computed eigenvalues. The tolerance parameter determines when the algorithm will stop looping, i.e., when we consider that the currently computed eigenvalue contributes little to the total number of triangles. The idea behind them is that due to the special spectral properties of real-world networks few iterations suffice to output a good approximation.

Specifically, EigenTriangle starts by computing the first eigenvalue λ_1. It then computes the second eigenvalue λ_2, and checks using the condition in the *repeat* loop if λ_2 contributes significantly or not to the current estimate of triangles, i.e., $\sum_{j=1}^{2} \lambda_j^3$. In the former case, the algorithm keeps iterating and computing eigenvalues until the stopping criterion is satisfied. Then, it outputs the estimate of the total number of triangles $\Delta'(G)$ using the computed eigenvalues and equation 2. EigenTriangleLocal additionally stores the eigenvectors corresponding to the top eigenvalues in order to make an estimate of Δ_i using equation 1. The *repeat* loop as in EigenTriangle computes eigenvalue-eigenvector pairs until the stopping criterion is met and the *for* loop computes the estimates Δ_i' of Δ_i, $i = 1, \ldots, n$.

Both algorithms use the subroutine *LanczosMethod* (Golub et al, 1989; Dem-

Algorithm 2 The EigenTriangleLocal algorithm

```
Require: Adjacency matrix $A$ $(n \times n)$
Require: Tolerance $tol$
Output: $\vec{\Delta}'(G)$ per node triangle estimation
  $\langle \lambda_1, \vec{u_1} \rangle \leftarrow LanczosMethod(A, 1)$
  $\vec{\Lambda} \leftarrow [\lambda_1]$
  $\mathbf{U} \leftarrow [\vec{u_1}]$
  $i \leftarrow 1$
  {initialize $i$, $\vec{\Lambda}, \mathbf{U}$}
  repeat
    $i \leftarrow i + 1$
    $\langle \lambda_i, \vec{u_i} \rangle \leftarrow LanczosMethod(A, i)$
    $\vec{\Lambda} \leftarrow \left[\vec{\Lambda}\ \lambda_i\right]$
    $\mathbf{U} \leftarrow [\mathbf{U}\ \vec{u_i}]$
  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 $\vec{\Delta}'(G)$
```

mel, 1997; Meurant, 2006) as a black box[1] to compute a low-rank eigendecomposition of the adjacency matrix. Lanczos method is a well studied projection based method for solving the symmetric eigenvalue problem using Krylov subspaces. It is based on simple matrix-vector multiplications. Furthermore, high quality software implementing Lanczos method is publicly available (ARPACK, Parallel ARPACK, MATLAB etc.). It is worth noting how easy it is to implement our algorithm in a programming language that offers routines for eigenvalue computation. For example, assuming that a k-rank approximation of the adjacency matrix gives good results, the piece of MATLAB code described in algorithm 3 will output an accurate estimate of the number of triangles. This function takes two input arguments, A and k which are the adjacency matrix representation of the graph and the desired rank of the low rank approximation respectively.

3.3. Why is EigenTriangle successful?

Real-world networks have several special properties, such as small-worldness, scale-freeness and self-similarity characteristics. For our work, the special spectral properties are crucial. Figure 1(a) and Figure 1(b) show the spectra of two real-world networks. Both are representative of the typical spectrum of a real-world network. These figures plot the value of the eigenvalue vs. its rank. The spectrum of Figure 1(a) corresponds to the Political Blogs network (Adamic et al, 2005), a small network with approximately 1,2K nodes and 17K edges. The spectrum

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

Algorithm 3 MATLAB implementation, k-rank approximation

```
function $\Delta'$ = EigenTriangleLocal(A,k) {A is the adjacency matrix, k is the required rank approximation}
n = size(A,1);
$\Delta'$ = zeros(n,1); {Preallocate space for $\Delta'$}
opts.isreal=1; opts.issym=1; {Specify that the matrix is real and symmetric}
[u l] = eigs(A,k,'LM',opts); {Compute top k eigenvalues and eigenvectors of A}
l = diag(l)';
for j=1:n do
    $\Delta'(j)$ = sum( l.^3.*u(j,:).^2)/2
end for
```

of Figure 1(b) corresponds to an anonymous social network with approximately 404K nodes and 2,1M edges. Notice that in the latter network, only the 800 top eigenvalues out of the approximately 404K eigenvalues are plotted.

The following two facts which are apparent in the two figures, play a crucial role in the effectiveness of our proposed algorithms:

1. The absolute values of the few top eigenvalues are skewed, typically following a power law (Faloutsos et al, 1999)[2],(Mihail et al, 2002),(Chung et al, 2003).
2. Moreover, the signs of the eigenvalues tend to alternate (Farkas et al, 2001) and thus their cubes roughly cancel out.

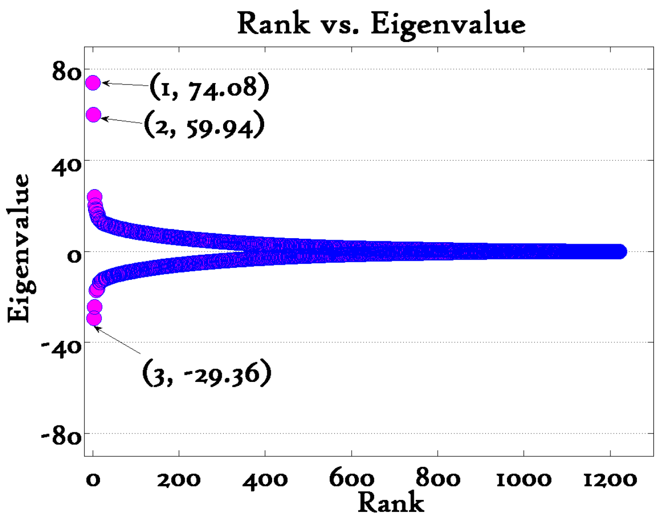
In other words, the contribution of the bulk of the eigenvalues is negligible compared to the contribution of the few top eigenvalues to the total number of triangles. This fact allows us to discard the largest part of the spectrum. Therefore we can keep just a handful of eigenvalues and approximate fast and well the number of triangles. Experimentally 1 to 25 eigenvalues, see Figure2(a), lead to a satisfactory approximation. The time complexity of our proposed algorithms is $O(c\text{nnz})$ where nnz is the number of non zeros in the adjacency matrix, i.e., twice the number of edges, and c is the total number of matrix vector multiplications Lanczos method performs. As we explain in the next subsection, the computation of a handful of the top eigenvalues results in a small number of iterations c and therefore the performance of our methods is fast.

3.4. Lanczos method and Real-World Networks

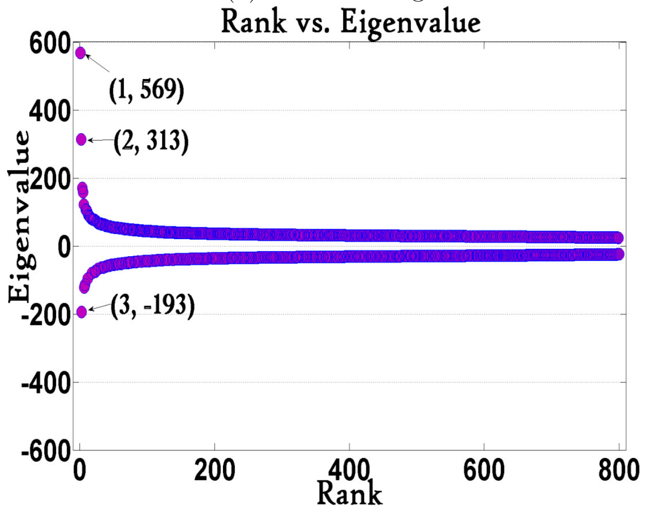
First we give a brief description of Lanczos method for computing the eigenvalues of a symmetric matrix and then we explain why it converges fast in the case of real-world networks.

Short Description of Lanczos Method: Consider a symmetric $n \times n$ matrix A whose eigenvalues and eigenvectors are sought and let $u \in \mathbb{R}^n$ be a given unit vector. Lanczos method is based on the subspace spanned by the vectors

[2] Even if the least squares fitting used in (Faloutsos et al, 1999) has been questioned as a methodology of fitting power laws and better methodologies have been developed (Clauset et al, 2009), the key property is the skewness observed in the values of the top eigenvalues rather than the exact distribution that they follow.



(a) Political Blogs



(b) Anonymous Social Network

Fig. 1. Spectra of two real-world networks, representative of the typical spectrum of networks with skewed degree distributions. Both figures (a) and (b) plot the value λ_i versus the rank i. Political blogs is a small network with ≈17K edges and ≈1,2K nodes. The Anonymous Social Network has ≈404K nodes and ≈2,1M edges. Figure (b) plots only the 800 top eigenvalues. Notice that (1) the first few eigenvalues are significantly larger than the rest, (2) which are almost symmetric around zero and (3) cubing amplifies these effects.

$u, Au, \dots, A^{k-1}u$, also known as the Krylov subspace. Let K be the $n \times k$ matrix $K = [u|Au|\dots|A^{k-1}u]$. For $k \leq m \leq n$, where m is the order of the minimal polynomial of u with respect to A, matrix K has full column rank. However, since the successive multiplications of matrix A lead the terms $A^j u$ for large j to being almost equal to the first eigenvector, it is necessary to get a numerically better base for this subspace. Using the Gram-Schmidt orthogonalization procedure we produce an orthonormal sequence of vectors $u = q_1, \ldots, q_k$ such that the following three term recurrence equation holds:

$$Aq_j = b_{j-1}q_{j-1} + a_jq_j + b_jq_{j+1} \tag{3}$$

The coefficients a_j, b_j can be found by using the orthogonality properties of the q_j vectors. Let Q be the matrix $Q = [q_1|\ldots|q_k]$. The matrix Q^TAQ is a small $k \times k$, tridiagonal matrix (containing the coefficients $a_1, \ldots, a_k$ in its main diagonal, and the coefficients $b_1, \ldots, b_{k-1}$ in the first diagonal above and below the main one) whose eigenvalues typically approximate well the top k eigenvalues of A. It is also worth noting that Lanczos method performs only matrix-vector multiplications making it a good option for a low rank approximation of a sparse matrix A. For more details see one of the following excellent references (Golub et al, 1989; Demmel, 1997; Edwards et al, 1979; Cullum et al, 2002).

Convergence of Lanczos method: As we know, the eigenvalues of matrix A are the roots of its characteristic polynomial. The latter is also known as the secular function. When the roots of the secular function are very close, Lanczos needs several iterations to find them. Even if there exist sophisticated methods for finding the roots of the secular function, e.g., (Cuppen, 1981), they run into similar problems with Newton's method when the two roots we are trying to find are very close (Meurant, 2006).

Since real-world networks tend to have skewed degree distributions which imply a skewed eigenvalue distribution too, Lanczos converges fast to the top eigenvalues because they correspond to roots of the secular function which are well separated. Therefore, assuming that the top eigenvalues provide us a satisfactory approximation to the total number of triangles implies that we can find fast a good estimate of the total number of triangles.

4. Experimental Results

We conduct numerous experiments in order to answer the following question: for at least 95% accuracy what are the speedups we can achieve for the triangle counting problem using EigenTriangle? First, we describe the experimental setup, and then we provide the experimental results.

4.1. Experimental set up

Each directed graph was converted into an undirected graph by ignoring the direction of the edges. Multiple edges and self-loops were removed. The number of nodes and edges of the networks used after the preprocessing are summarized in table 2. [3] As the competitor for our method we chose the *Node Iterator* (see section 2), a basic, non-trivial exact listing algorithm which allows us to directly evaluate the quality of EigenTriangle and EigenTriangleLocalby comparing the outputs. We ran the experiments in a machine with a quad-processor Intel Xeon 3GHz with 16GB of RAM. We express the experimental results as

[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/`

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)

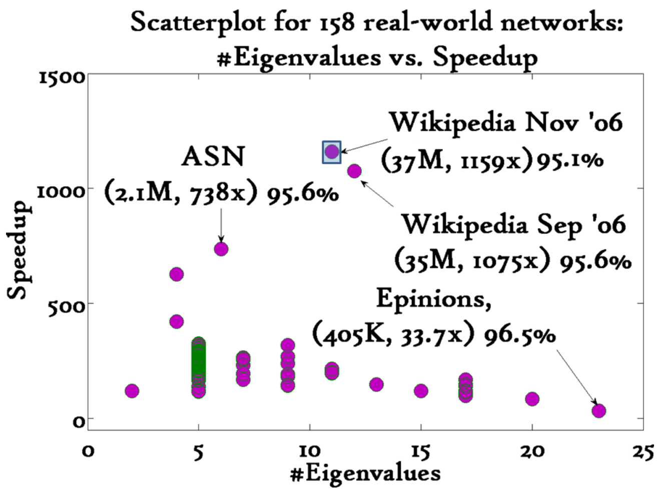
Table 2. Summary of real-world networks used.

the ratio of the clock-work times of the *Node Iterator* to the EigenTriangle (speedup). All algorithms were implemented in MATLAB. For the eigenvalue computation, we used the command *eigs* to which we passed a struct *opts*, specifying that our matrices are symmetric and real, as shown in Algorithm 3.

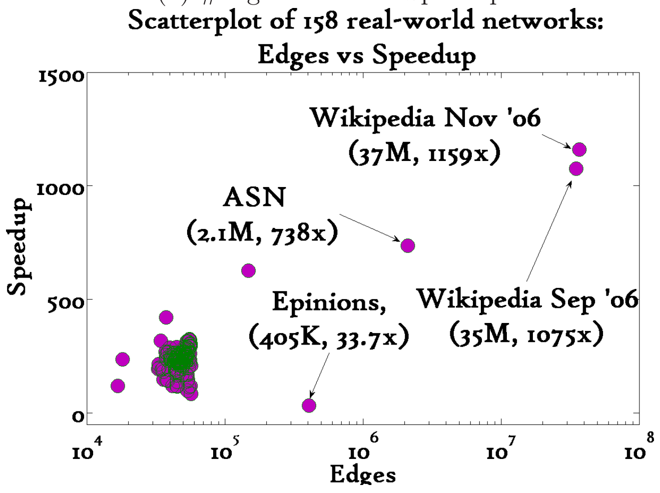
4.2. Total Triangle Counting

Figures 2(a), 2(b) summarize the results of the EigenTriangle algorithm when applied to 158 real world networks. Specifically, Figure 2(a) plots the achieved speedup versus the number of eigenvalues required to get at least 95% accuracy. Figure 2(b) plots the speedup versus the number of edges in the graph. The following facts are worth noting:

1. The mean number of eigenvalues required to achieve more than 95% is 6.2 with standard deviation equal to 3.2. The mean speedup is 250× with the standard deviation equal to 123. The maximum speedup is 1159× whereas the minimum speedup is 33.7×.
2. The speedup appears 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. According to (Mihail et al, 2002), the top eigenvalues exhibit the same behavior, i.e., get more detached from the bulk. Therefore, with a handful of eigenvalues, we get high accuracy, since their cubes dominate the total sum of the cubes of the eigenvalues. Furthermore,



(a) #Eigenvalues vs. Speedup



(b) Edges vs. Speedup

Fig. 2. 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).

due to the fast convergence of Lanczos method, EigenTriangle ouputs fast its estimate.
An exception to the observation above is the performance of our method on the Epinions graph. EigenTriangle needs to compute more than 20 eigenvalues to ouput a high quality estimate, due to the specific spectrum of this graph. This fact has as a consequence the smallest speedup observed (33.7$\times$) which is still significant.

3. An important issue in EigenTriangle and EigenTriangleLocal is the choice of the tolerance parameter *tol*. Clearly, if the parameter is set to $\epsilon \rightarrow 0$,

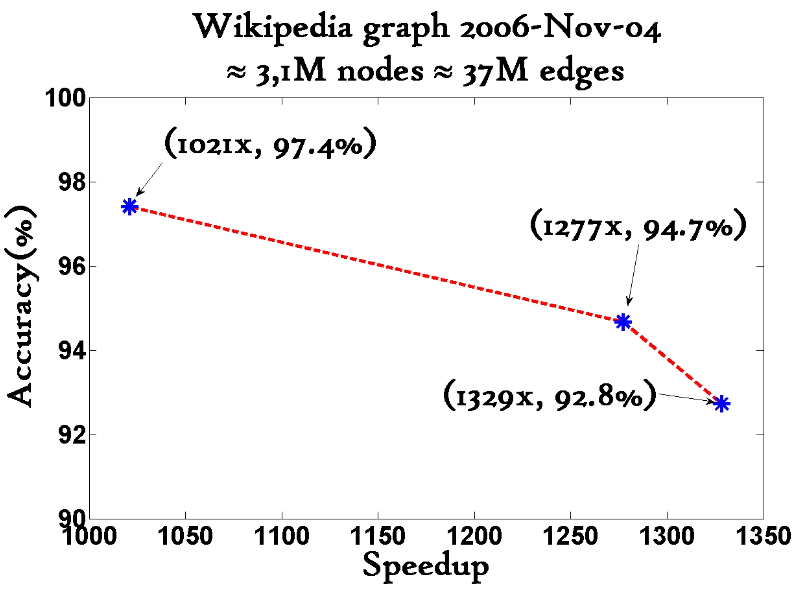


Fig. 3. Zooming in the point enclosed by a rectange of figure 2(a). This figure plots the accuracy obtained versus the speed-up ratio for the Wikipedia web graph ($\approx 3,1M$ nodes, $\approx 37M$ edges). Proposed method achieves 1021x faster time, for 97.4% accuracy, compared to a typical competitor, the *Node Iterator* method.

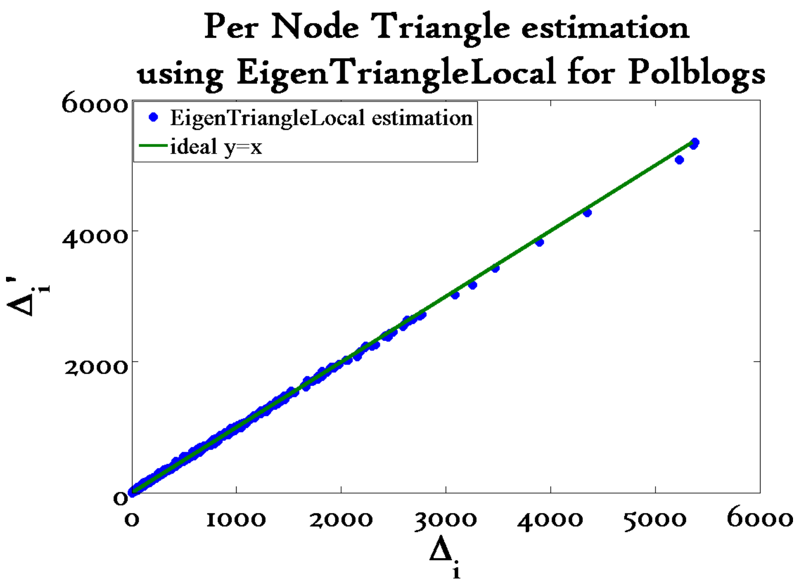


Fig. 4. Scatterplot of Δ_i' (estimated #triangles of node i) vs. Δ_i (actual number) for Polblogs using a rank 10 approximation. Relative reconstruction error is $7 * 10^{-4}$ and the Pearson's correlation coefficient is 99.97%.

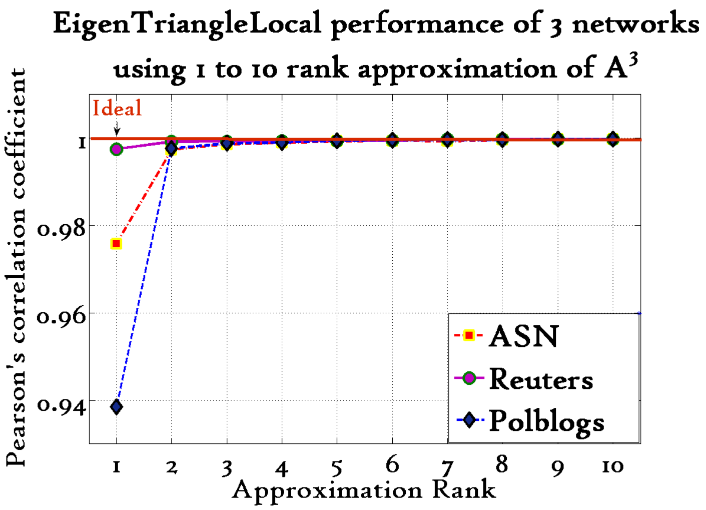


Fig. 5. Local triangle reconstruction for three real-world networks using rank 1 to 10 approximation of the diagonal of $\mathbf{A}^3$. Pearson's correlation coefficient ρ vs. approximation rank.Notice that after rank 2 ρ is greater than 99.9% for all three networks.

both algorithms will have to compute many eigenvalues slowing down significantly their performance. An extremely small value for the parameter *tol* is likely to turn the proposed algorithms into slower than other exact counting algorithms, since computing the whole spectrum of a square $n \times n$ matrix has time complexity $O(n^3)$ with potential convergence and numerical problems. On the other hand, if the tolerance parameter is set to a high value, then the accuracy of the estimate can be unsatisfactory. It is not clear how to decide the *tol* parameter a priori. However, this does not render EigenTriangle useless. A useful "rule of thumb" for practitioners based on Figure 2(a) is to compute 5-15 eigenvalues and see how well does the sum S_i of the cubes of the eigenvalues from 1 to i compare to S_{i+1}. This is essentially the same criterion with the stopping criterion of the algorithms we propose. However, using this "rule of thumb" is a practical way of running the algorithms without depending on the parameter *tol*.If one wants to run the algorithm as is, a choice of *tol* that was satisfactory in many experiments was 0.05.

4. Figure 3 is zooming in the point enclosed with a rectangle of Figure 2(a). This point corresponds to the Wikipedia Web graph (4 Nov. 2006 with approximately 3,1M nodes, and 37M edges). We observe that with a single eigenvalue we get 92.8% accuracy and 1329× speedup. When the algorithm terminates, the accuracy is 97.4%, the speedup 1021× and the rank of the required approximation equal to 7.

4.3. Local Triangle Counting

To measure the performance of the EigenTriangleLocal algorithm, we use Pearson's correlation coefficient ρ and the relative reconstruction error, as in (Becchetti et al, 2008).

$$RRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\Delta_i - \Delta_i'|}{\Delta_i} \quad (4)$$

In figure 4 we see how well $\vec{\Delta}'(G)$, i.e., the vector which contains in its i-th coordinate our estimate of the number of triangles in which node i participates in, approximates $\vec{\Delta}(G)$ using the top 10 eigenvalues and eigenvectors for the Political blogs dataset. The RRE we obtain is $7 * 10^{-4}$ and ρ is equal to 0.9997, close to the ideal value 1. Figure 5 explains why our proposed methods work well in practice. It plots ρ versus the rank of the approximation. We observe that after the two rank approximation, for all three networks the approximation is excellent: ρ 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. Finally, it is worth noting that figure 5 suggests that the rank-10 approximation of the adjacency matrix used to produce Figure 4 is significantly larger than the minimum one needed to obtain satisfactory results.

5. Theoretical Ramifications

In this section we extend our theoretical results in the following three ways. First, we show a simple sampling procedure allows us to apply the core idea of EigenTriangle on large graphs which do not fit into the main memory. The resulting algorithm is the FastSVD and is based on the seminal work of (Drineas et al, 2004). Secondly, using the spectral counting idea, we prove a theorem which provides a closed formula for the number of triangles in Kronecker graphs. Finally, we discuss about cases where the EigenTriangle algorithm still works, even if the graph is not a "real-world" network.

5.1. Counting Triangles via Fast SVD

We consider the following simple randomized procedure to speedup further the performance of our proposed algorithms: Given our $n \times n$ adjacency matrix A, integers c, k such that $c \leq n$, $k \leq c$, we sample c integers from 1 to n, with the probability of choosing integer i equal to $Pr(i) = p_i = \frac{d_i}{2m}$, where d_i is the degree of node i and m is the total number of edges in the graph. Let $\{i_1, \ldots, i_c\}$ be the indices sampled. We create a $n \times c$ matrix $A' = [\frac{A^{(i_1)}}{\sqrt{cp_{i_1}}}|\frac{A^{(i_2)}}{\sqrt{cp_{i_2}}}|\ldots|\frac{A^{(i_c)}}{\sqrt{cp_{i_c}}}]$. We use A' to approximate the k top eigenvalues and eigenvectors of A, where k is assumed to be the required rank of the approximation of the adjacency matrix which gives us a good estimate of the number of triangles in the graph. The top k left singular vectors $\hat{u}^{(i)}_{i=1\ldots k}$ of A' define a subspace which is close to the optimal k dimensional subspace spanned by the top k left singular vectors $u^{(i)}_{i=1\ldots k}$ of A. In order to approximate the right singular vectors as suggested by (Drineas et al, 2004) one should sample rows of A. Instead, we choose to approximate the right singular vectors using the equation $\hat{V}^T = \hat{\Sigma}^{-1}\hat{U}^T A$ assuming that $\hat{\Sigma}^{-1}\hat{U}^T A \approx \Sigma U^T A$. The signs of the eigenvalue λ_i can be recovered by multiplying the corresponding left and right singular vectors. For example if we had the exact SVD of A we could determine the i-th eigenvalue by $\lambda_i = \sigma_i (v^{(i)})^T u^{(i)}$. We approximate λ_i by $\hat{\lambda}_i$

Algorithm 4 The FastSVD Triangle Counting algorithm

```
Require: Adjacency matrix $A$ ($n$x$n$)
Require: c, $c \leq n$
Require: k, $k \leq c$
Output: $\Delta'(G)$ global triangle estimation
  for $j = 1$ to $c$ do
    Pick an integer from $\{1, \ldots, n\}$, where $p_i = \frac{d_i}{2m}$
    Include $\frac{A^{(i)}}{\sqrt{cp_i}}$ as a column of $A'$
  end for
  Compute the top k left singular vectors $\hat{u}^{(1)}, \ldots, \hat{u}^{(k)}$ and the top k singular
  values $\hat{\sigma}_1 > \ldots > \hat{\sigma_k} > 0$ of $A'$
  $\hat{U} \leftarrow [\hat{u}^{(1)}| \ldots |\hat{u}^{(k)}]$
  $\hat{\Sigma} \leftarrow \text{diag}(\hat{\sigma}_1, \ldots, \hat{\sigma_k})$
  $\hat{V}^T \leftarrow \hat{\Sigma}^{-1} \hat{U}^T A$
  for $j = 1$ to $k$ do
    $\hat{\lambda}_j \leftarrow \hat{\sigma}_j \text{sgn}((\hat{v}^{(j)})^T \hat{u}^{(j)})$
  end for
  $\Delta'(G) \leftarrow \frac{1}{6} \sum_{i=1}^{k} \hat{\lambda}_i^3$
  return $\Delta'(G)$
```

where $\hat{\lambda}_i \leftarrow \hat{\sigma}_i \text{sgn}((\hat{v}^{(i)})^T \hat{u}^{(i)})$. The reason that the sign function appears[4] is that the ideal situation where the inner product $(v^{(i)})^T u^{(i)}$ should equal either +1 or -1 does not occur in practice. This procedure results in algorithm 4. The reason that this procedure is theoretically sound is the seminal work of (Drineas et al, 2004). Specifically, since our matrix is a square, symmetric matrix containing only zeros and ones, the probabilities $p_i = \frac{||A^{(i)}||^2}{||A||_F^2}$ defined in (Drineas et al, 2004) are simplified to the expression $\frac{d_i}{2m}$. Intuitively, by favoring nodes of high degree we can recover the number of triangles approximately.

We apply Algorithm 4 on the anonymous social network, for which with 6 eigenvalues we obtain a 95.6% accuracy using Lanczos method. The obtained accuracy using Algorithm 4 is 95.46% using k equal to 6 and c equal to 100. With both algorithms we are able to compute with high accuracy an estimate of the 38036823 total triangles which exist in the graph. The speedup is not apparent due to the overhead of the sampling procedure and the necessary multiplications we make to find the signs of the singular values. Combined with the overall small amount of time needed to compute the top six eigenvalues (less than 4 seconds) the performance of EigenTriangle and Algorithm 4 are comparable. Nonetheless, algorithm 4 is useful, allowing us to apply the core idea of EigenTriangle on graphs which do not fit into the main memory.

5.2. Kronecker graphs

Kronecker graphs (Leskovec et al, 2005) have attracted recent interest, because they can be made to mimic real graphs well (Leskovec et al, 2007). In the fol-

[4] The sign function sgn(·) returns the sign of its argument.

lowing we give a closed formula that estimates the number of triangles for a Kronecker graph. Some definitions first:

Let A be the $n \times n$ adjacency matrix of an n-node graph G_A with $\Delta(G_A)$ triangles, and let $B = A^{[k]}$ be the k-th Kronecker power of it, that is, an $n^k \times n^k$ adjacency matrix (see (Leskovec et al, 2005) for the exact definition of the deterministic Kronecker graph). Let G_B denote the corresponding graph. Let $\vec{\lambda} = (\lambda_1, .., \lambda_n)$ be the eigenvalues of matrix A. The following theorem holds:

Theorem 5.1 (KroneckerTRC). The number of triangles $\Delta(G_B)$ of G_B can be computed from the n eigenvalues of A:

$$\Delta(G_B) = 6^k \Delta(G_A)^{k+1} \quad k \geq 0. \tag{5}$$

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

$$\Delta(G_C) = 6^r \Delta(G_A)^{r+1}$$

The eigenvalues of D are given by the Kronecker product $\vec{\lambda} \otimes \vec{\mu}$. Using these two facts, we will now show that KroneckerTRC holds for $r + 1$. By Lemma 3.2, we get that the number of triangles in G_D is given by the following equation: $\Delta(G_D) = \frac{\sum_{i=1}^{s} \sum_{j=1}^{n} \mu_i^3 \lambda_j^3}{6} = \frac{\sum_{i=1}^{s} \mu_i^3 \sum_{j=1}^{n} \lambda_j^3}{6} = \frac{\sum_{i=1}^{s} \mu_i^3 6\Delta(G_A)}{6} = 6\Delta(G_A) \frac{\sum_{i=1}^{s} \mu_i^3}{6} = 6\Delta(G_A) 6^r \Delta(G_A)^{r+1} = 6^{r+1} \Delta(G_A)^{r+2}$

Therefore KroneckerTRC holds for all $k \geq 0$. □

Timing results, and stochastic Kronecker graphs The above theorem results in tremendous time savings and perfect accuracy for deterministic Kronecker graphs. For example, 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 7, we get 10^6 faster performance. As the size of the Kronecker graph increases, we obtain arbitrarily large speedups.

It is interesting that the KroneckerTRC theorem also leads to a fast estimation of triangles, even for stochastic Kronecker graphs (Leskovec et al, 2007). Stochastic Kronecker graphs have been shown to mimic 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 graph show that these random edge manipulations have little effect on the accuracy. Specifically, our experiments with n=6,561 and m=2,202,808[5], show that we obtain $1.5 * 10^6 \times$ 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.

5.3. 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 (Bollobas, 2001). This

[5] Seed matrix (using MATLAB notation): [.99 .9 .9;.9 .99 .1;.9 .1 .99], depth of recursion: 7

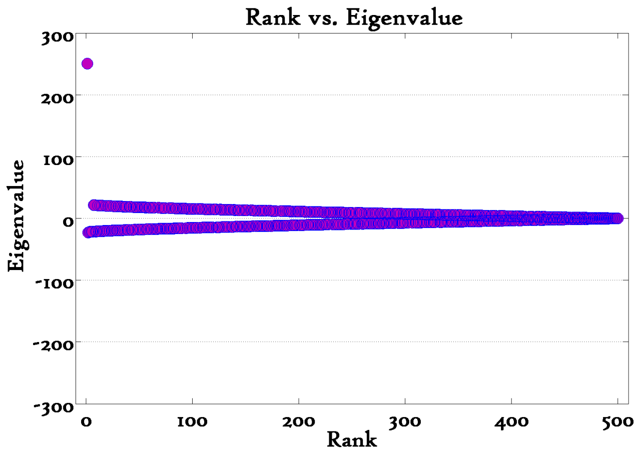


Fig. 6. Eigenvalue vs. rank plot of a random Erdős-Rényi graph $G_{n,p}$, with n =500 and $p = \frac{1}{2}$.

is due to Wigner's semi-circle law for all but the first eigenvalue (Furedi et al, 1981). In figure 6 we see the eigenvalue-rank plot for an Erdős-Rényi graph with n =500 and $p = \frac{1}{2}$, i.e., p constant.

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[6].

6. Conclusions

In this work, we propose the EigenTriangle and EigenTriangleLocal algorithms (Tsourakakis, 2008) to estimate the total number of triangles and the number of triangles per node respectively in an undirected, unweighted graph. The special spectral properties which real-world networks frequently possess make both algorithms efficient for the triangle counting problem. We showed experimentally that our method outperforms a straight-forward, exact triangle counting algorithm using different types of real-world networks. To our knowledge, the knowledge for the bulk of the spectrum is limited in contrast to the few, top eigenvalues (Mihail et al, 2002; Chung et al, 2003). An interesting theoretical problem is to find the distribution of the bulk of the eigenvalues of a random graph generated by a model which mimics real-world networks. As the underlying eigendecomposition algorithm we use Lanczos method, which converges fast as we explain in Section 3. In practice, EigenTriangle using in average a rank six approximation of the adjacency matrix results in at least 95% accuracy, for speedups ranging from 30× to 1000× compared to the *Node Iterator* algorithm. However, this behavior is empirical and requires further theoretical justification

[6] It makes no sense to apply EigenTriangle on Erdős-Rényi since we can approximate well the total number of triangles, i.e., $\binom{n}{3} p^3$.

and understanding. More experiments is another future direction, in order to establish to what extent real-world networks share similar spectral properties.

We also provide a simple randomized algorithm which allows us to use the core idea of EigenTriangle on graphs which do not fit in the main memory. The key idea behind this lies in the seminal work of (Drineas et al, 2004) and the fact that we can find the eigendecomposition of the adjacency matrix through its Singular Value Decomposition. Furthermore, we give a closed formula for the number of triangles in deterministic Kronecker graphs and show that the same formula can be used to approximate satisfactorily the number of triangles in a stochastic Kronecker graph as well.

It is worth noting that since (Tsourakakis, 2008) other combinatorial triangle counting algorithms have been developed (Tsourakakis et al, KDD, 2009) with strong theoretical guarantees (Tsourakakis et al, Arxiv, 2009). These algorithms are independent of any special spectral properties. Giving guarantees for the performance EigenTriangle algorithm under some random graph model, e.g., (Chung et al, 2003) is another research direction as already mentioned. Nonetheless, EigenTriangle is a viable option for computing triangles in real-world networks which also shows that restricting our input graphs to possess special properties like those possessed empirically by real-world networks can lead us in developing efficient algorithms. Investigating further properties of real-world networks and developing such algorithms is another broad research direction.

Acknowledgements. The author would also like to thank Petros Drineas, Ioannis Koutis and Christos Faloutsos for helpful discussions and the anonymous reviewers for their constructive comments.

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0705359. Also, under the auspices of the University of California Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-CONF-404625), subcontracts B579447, B580840. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

References

Achlioptas D, McSherry F (2001) Fast Computation of Low Rank Matrix Approximations. In Symposium on Theory of Computing (STOC), 2001

Adamic L, Glance N. (2005) The political blogosphere and the 2004 U.S. election: divided they blog. In Workshop on Link Discovery (LinkKDD), 2005

Alon N, Matias Y, Szegedy M. (1996) The space complexity of approximating the frequency moments. In Symposium on Theory of Computing (STOC), 1996

Alon N, Yuster R, Zwick U (1997). Finding and Counting Given Length Cycles. In Algorithmica, Volume 17, Number 3, pp 209-223

Bar-Yosseff Z, Kumar R, Sivakumar D (2002). Reductions in streaming algorithms, with an application to counting triangles in graphs. In Symposium on Discrete Algorithms (SODA), 2002

Becchetti L, Boldi P, Castillo C, Gionis A (2008) Efficient Semi-Streaming Algorithms for Local Triangle Counting in Massive Graphs. In Knowledge Discovery and Data Mining (KDD), 2008

Bollobas B (2001) Random Graphs. Publisher Cambridge University Press

Broder A Z, Charikar M, Frieze A, Mitzenmacher M (1998) Min-wise independent permutations. In Symposium on Theory of Computing (STOC), 1998

Buriol L, Frahling G, Leonardi S, Marchetti-Spaccamela A, Sohler C (2006) Counting Triangles in Data Streams. In Principles of database systems (PODS), 2006

Chung F, Lu L, Vu V (2003) Eigenvalues of Random Power law Graphs. In Annals of Combinatorics, Volume 7, pp 21-33.

Clauset A, Shalizi C R, Newman M E J (2009) Power-law distributions in empirical data. In SIAM Review, Vol. 51, No. 4

Coppersmith D, Winograd S (1987) Matrix multiplication via arithmetic progressions. In Symposium on Theory of Computing (STOC), 1987

Cullum J, Willoughby RA (2002) Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. 1. Publisher Society for Industrial and Applied Mathematics, 2002

Cuppen J J M (1981) A divide and conquer method for the symmetric tridiagonal eigenproblem. In Numer. Math., v. 36, pp 177-195

Deerwester S, Dumais S, Furnas G, Landauer T, Harshman R (1990) Indexing by latent semantic analysis. In Journal of the American Society for Information Science, 41(6), pp 391-407

Demmel J (1997) Applied Numerical Linear Algebra. Publisher Society for Industrial and Applied Mathematics, 2002

Drineas P, Frieze A, Kannan R, Vempala S, Vinay V (2004) Clustering Large Graphs via the Singular Value Decomposition. In Mach. Learning Journal '04, Volume 56, pp 9-33

Eckmann JP, Moses E (2002) Curvature of co-links uncovers hidden thematic layers in the World Wide Web. In Proceedings of the National Academy of Sciences (PNAS), Number 9, pp 5825-5829, Volume 99, 2002

Edwards J T, Licciardello D C, Thouless D J (1979) Use of Lanczos Methos for Finding Complete Sets of Eigenvalues of Large Sparse Symmetric Matrices. In IMA Journal of Applied Mathematics, Volume 23, pp 277-283

Faloutsos M, Faloutsos P, Faloutsos C (1999) On Power-law Relationshipds of the Internet Topology. In SIGCOMM, 1999

Farkas I, Derenyi I, Barabasi AL, Vicsek T (2001) Spectra of Real-World Graphs: Beyond the semicircle law. In Physical Review E, Volume 64, 2001

Fienberg S., Rinaldo A., Zhou Y (2009) On the Geometry of Discrete Exponential Families with Application to Exponential Random Graph Models. CMU Technical Report STAT-TR871, 2009

Furedi Z, Komlos J (1981) The eigenvalues of random symmetric matrices. In J. Combinatorica, Volume 1, Number 3, pp 233-241

Godsil C.D, Royle G (2001) Algebraic Graph Theory. Publisher Springer

Golub G.H, Van Loan C.F (1989) Matrix Computations. Publisher Johns Hopkins Press

Kang U, Tsourakakis C, Faloutsos C (2009) PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations. In IEEE International Conference on Data Mining (ICDM), 2009 Available at `http://www.cs.cmu.edu/~pegasus/`

Kleinberg J (1999) Authoritative sources in a hyperlinked environment. In J. ACM 1999, Volume 46, Number 5, pp 604-632

Latapy M (2008) Practical algorithms for triangle computations in very large (sparse (power-law)) graphs. In J. Theoretical Computer Science, vol. 407, pp 458-473, 2008

Leskovec J, Chakrabarti D, Kleinberg J, Faloutsos C (2005) Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. In Practice of Knowledge Discovery in Databases (PKDD), 2005

Leskovec J, Faloutsos C (2007) Scalable modeling of real graphs using Kronecker multiplication. In International Conference on Machine Learning (ICML), 2007

Leskovec J, Backstrom L, Kumar R, Tomkins A (2008) Microscopic Evolution of Networks. In Knowledge Discovery and Data Mining (KDD), 2008

Meurant G (2006) The Lanczos and Conjugate Gradient Algorithms, From Theory to Finite Precision Computations. Publisher Society for Industrial and Applied Mathematics, 2006

Mihail M, Papadimitriou C (2002) The eigenvalue power law. In RANDOM, 2002

Ove F., Strauss D. (1986) Markov Graph. In Journal of the American Statistical Association, Volume 81, pp 832-842

Papadimitriou C, Raghavan P, Tamaki H, Vempala S (1998) Latent Semantic Indexing: A Probabilistic Analysis. In Principles of Database Systems (PODS), 1998

Strang G (2003) Introduction to Linear Algebra. Publisher Society for Industrial and Applied Mathematics, 2003

Schank T, Wagner D (2004) DELIS-TR-0043 Finding, Counting and Listing all Triangles in Large Graphs, An Experimental Study. Tech Report 0043, 2004

Schank T, Wagner D (2005) Approximating Clustering Coefficient and Transitivity. In Journal of Graph Algorithms and Applications, 9, 265–275, 2005

Song G, Cui B, Zheng B, Xie K, Yang D (2009) Accelerating sequence searching: dimensionality reduction method. In Knowledge and Information Systems (KAIS), Volume 20, pp 301-322, 2009

Tsourakakis C (2010) MACH: Fast Randomized Tensor Decompositions. In SIAM Conference on Data Mining (SDM10), 2010

Tsourakakis C (2008) Fast Counting of Triangles in Large Real Networks without Counting: Algorithms and Laws. In IEEE International Conference on Data Mining (ICDM), 2008

Tsourakakis C, Kang U, Miller GL, Faloutsos C (2009) DOULION: counting triangles in massive graphs with a coin. In Knowledge Discovery and Data Mining (KDD), 2009

Tsourakakis C, Kolountzakis M, Miller GL Approximate Triangle Counting. In Arxiv 0904.3761, 2009

Tsourakakis C, Drineas P, Michelakis E, Koutis I, Faloutsos C Spectral Counting of Triangles in Power-Law Networks via Element-Wise Sparsification. In Advances in Social Networks Analysis and Mining (ASONAM), 2009

Tsourakakis C, Drineas P, Michelakis E, Koutis I, Faloutsos C Spectral Counting of Triangles in Power-Law Networks via Element-Wise Sparsification and Triangle-Based Link Recommendation. Invited Book Chapter in Advances in Social Networks Analysis and Mining, Submitted, 2010

Wasserman S, Faust K (1994) Social network analysis. Publisher Cambridge University Press, 1994

Xiang S, Nie F, Song Y, Zhang C, Zhang C (2009) Embedding new data points for manifold learning via coordinate propagation. In Knowledge and Information Systems (KAIS) Volume 19, pp 159-184, 2009

Charalampos Tsourakakis is currently a Ph.D. candidate in the Machine Learning Department, at Carnegie Mellon University, USA. He holds a Diploma in Electrical and Computer Engineering from the National Technical University of Athens. His main research interests lie in the fields of computational biology, machine learning and (multi)linear algebra.

Correspondence and offprint requests to: Charalampos Tsourakakis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3891, USA. Email: ctsourak@cs.cmu.edu