

# Spectral Counting of Triangles via Element-Wise Sparsification and Triangle-Based Link Recommendation

Charalampos E. Tsourakakis  
School of Computer Science  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA 15213-3891  
ctsourak@cs.cmu.edu

Eirinaios Michelakis  
EECS University of California, Berkeley  
387 Soda Hall, Berkeley, CA 94720-1776  
ireneos@cs.berkeley.edu

Petros Drineas  
Department of Computer Science  
Rensselaer Polytechnic Institute  
110 8th Street, Troy, NY 12180-3590  
drinep@cs.rpi.edu

Ioannis Koutis  
School of Computer Science  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA 15213-3891  
jkoutis@cs.cmu.edu

Christos Faloutsos  
School of Computer Science  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA 15213-3891  
christos@cs.cmu.edu

## Abstract

*Triangle counting is an important problem in graph mining. The clustering coefficient and the transitivity ratio, two commonly used measures effectively quantify the triangle density in order to quantify the fact that friends of friends tend to be friends themselves. Furthermore, several successful graph mining applications rely on the number of triangles in the graph.*

*In this paper, we study the problem of counting triangles in large, power-law networks. Our algorithm, SPARSIFYINGEIGENTRIANGLE, relies on the spectral properties of power-law networks and the Achlioptas-McSherry sparsification process. SPARSIFYINGEIGENTRIANGLE is easy to parallelize, fast, and accurate.*

*We verify the validity of our approach with several experiments in real-world graphs, where we achieve at the same time high accuracy and considerable speedup versus a straight-forward exact counting competitor.*

*Finally, our contributions include a simple method for making link recommendations in online social networks based on the number of triangles as well as a procedure for estimating triangles using sketches.*

## 1 Introduction

It is a well known fact in social network analysis that friends of friends tend to be friends themselves [29]. Triangles are an important indicator of this property. Two measures that quantify the triangle density of a graph are the clustering coefficient and the transitivity ratio [23].

Besides the significance of triangles in network analysis statistics, they also play an important role in graph mining applications: Eckmann and Moses showed how one can use triangles in order to uncover the hidden thematic structure of the web [12] and Becchetti et al. in [6] used the local distribution of triangles and the clustering coefficient to detect spamming activity. Furthermore, triangle-related power laws [25] can be used to define outliers in a graph with respect to triangles.

In this paper we focus on the problem of counting triangles in large networks. The main contribution of this work is a novel method for counting triangles in a large power-law network: we show how one can sparsify the graph by converting it into another weighted graph, with significantly smaller number of edges, in order to count the number of triangles in the sparse graph with a recently introduced method, called EIGENTRIANGLE[25]. Furthermore, our method is easy to parallelize since it uses only matrix-

Symbol	Definition
$G$	Undirected simple graph
$d_{max}$	maximum node degree
$\Delta$	total number of triangles
$\Delta'$	EIGENTRIANGLE's estimation of $\Delta$
$m, n$	Number of edges and nodes.
$[n] = (1..n)$	Node ids
$\mathbf{A}$	Adjacency matrix
$\mathbf{A}^{(k)}$	optimal k-rank approximation of matrix A
$\lambda_i$	top- $i$ -th eigenvalue (absolute value)
$\vec{\Lambda}_k = [\lambda_i]_{i=1..k}$	$k$ top eigenvalues
$p$	sparsification parameter (probability of keeping an edge)

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

vector multiplications, easy to implement, and most importantly gives considerable speedups versus a straight-forward competitor. Finally, we validate the validity of our approach in several real world networks, where we achieve important speedups while being very accurate.

The outline of the paper is as follows: in section 2 we present briefly the related work, in section 3 we describe the proposed algorithm and in section 4 we show the experimental results. In section 5 we present a procedure for estimating the number of triangles using sketches and in section 6 an application of triangles for link recommendation in social networks. We conclude in section 7.

## 2 Background and Related Work

In this section we describe briefly existing work on the problem of counting triangles and the Achlioptas-McSherry low rank approximation algorithm. In the rest of the paper, we use the notation of Table 1.

Let  $G(V, E)$ ,  $n=|V|$ ,  $m=|E|$  be an undirected graph without self-edges. A triangle is defined as a three node fully connected subgraph of  $G$ .

**Exact Counting Methods.** The obvious way to count the number of triangles in a graph is to examine each of the  $\binom{n}{3}$  combinations of nodes and check whether they form a triangle or not. As the procedure suggests, the time complexity is  $O(n^3)$ .

Since the problem of counting triangles can be reduced

to matrix multiplication, the complexity of counting triangles can be reduced to  $O(n^{2.376})$  [9], which is the lowest time complexity. Alon, Yuster and Zwick in [4] described an algorithm running in  $O(m^{\frac{2\omega}{\omega+1}}) \subset O(m^{1.41})$  time and  $\Theta(n^2)$  space, which is prohibitive even for graphs of moderate size.

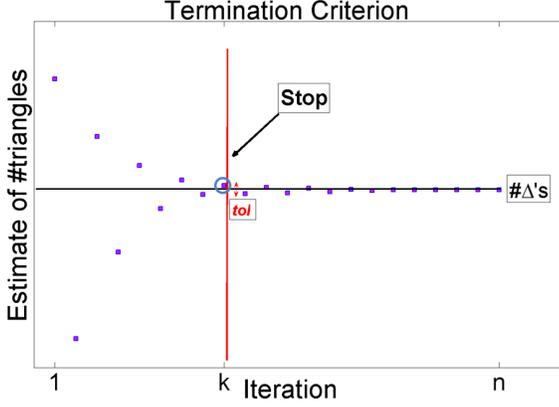
Therefore, listing methods [24] are preferred instead of matrix-multiplication based methods. Such methods are the NODEITERATOR and the EDGEITERATOR. NODEITERATOR considers each one of the  $n$  nodes and examines which pairs of its neighbors are connected. The time complexity of the NODEITERATOR is  $O(nd_{max}^2)$ . This is a significant improvement over the brute-force approach when the graph is sparse. The EDGEITERATOR algorithm computes for each edge the number of triangles that contain it. The time complexity of this algorithm is  $O(md_{max})$ . Both methods are equivalent asymptotically [24]. Schank and Wagner in [24] propose the *forward* algorithm with running time  $\Theta(m^{\frac{3}{2}})$  and space complexity  $O(m)$ . A nice survey and the state-of-the-art algorithms are described in [19].

**Approximate Counting Algorithms.** In the streaming approach, we restrict ourselves to one or at most a constant number of passes over the data. The goal is to output an accurate estimate of the number of triangles with high probability. Bar-Yossef, Kumar and Sivakumar showed in [5] how one can approximate the number of triangles by using the Alon-Matias-Szegedy [3] method for approximating frequency moments. New streaming algorithms were introduced in [7].

In [6], Becchetti, Boldi, Castillo and Gionis introduced the semi-streaming model to solve the local triangle counting problem. Their method relies on the locality sensitivity hashing concept. In contrast to the streaming model, this model relaxes the strict restriction of the constant number of passes over the data. Instead it performs  $O(\log(n))$  sequential scans over the edge file. Finally, in [26, 27] an efficient combinatorial algorithm which tosses a coin for each edge was introduced and analyzed.

**EIGENTRIANGLE.** Recently, Tsourakakis gave two approximation algorithms in [25] for counting the total number of triangles and the triangles per node. It was observed that a low-rank approximation of the adjacency matrix yields in many real-world networks a fast, accurate, and parallelizable method for counting triangles in power-law networks. The first theorem in [25], which is of interest to us in this work, is the following:

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



**Figure 1. An ideal scenario for SPARSIFYINGEIGENTRIANGLE : Figure shows the termination criterion for SPARSIFYINGEIGENTRIANGLE . Plot of the triangle estimate at the  $i$ -th iteration versus  $i$ . The algorithm decides at the  $k$ -th iteration to stop since its estimate did not change significantly from the  $k$ -th to the  $(k + 1)$ -st iteration. As we see from the plot, the estimate in the circle is close to the actual value of the number of triangles.**

**Achlioptas-McSherry Low Rank Approximation Algorithm** Approximating a matrix with a low rank matrix is a desired task in many applications; see [28, 10]. Since computing the optimal solution is expensive, approximate solutions are used in practice; see [11] and [15]. Achlioptas and McSherry showed in [1] how to obtain a matrix  $\hat{A}$  has the following properties with high probability:

$$\left\|A - \hat{A}^{(k)}\right\|_2 \leq \left\|A - A^{(k)}\right\|_2 + O\left(\left(\frac{n}{p}\right)^{\frac{1}{2}}\right) \quad (2)$$

$$\left\|A - \hat{A}^{(k)}\right\|_F \leq \left\|A - A^{(k)}\right\|_F + O\left(\left(\frac{n}{p}\right)^{\frac{1}{4}}\left\|\hat{A}^{(k)}\right\|_F^{\frac{1}{2}}\right) \quad (3)$$

These equation reveal the existence of a matrix  $\hat{A}^{(k)}$  for a given  $k$  “close” to the optimal  $A^{(k)}$  with respect to both the 2-norm and the Frobenius norm. Furthermore, they showed how one can obtain the  $\hat{A}$  in a very simple way: toss a biased coin for each entry  $A_{ij}$  of the matrix and with probability  $p$  keep that specific entry. If  $A_{ij}$  is kept, it is set equal to  $\frac{A_{ij}}{p}$ , otherwise it is set to 0.

### 3 Proposed Method

The method builds on the top of the EIGENTRIANGLE and Achlioptas-McSherry ideas. These ideas are: (a) A low

```

Require: Adjacency matrix  $A$  ( $n \times n$ )
Require: Tolerance  $tol$ 
Require: Sparsification parameter  $p$ 
Output:  $\Delta'(G)$  global triangle estimation
{ Stage 1: Achlioptas-McSherry Sparsification }
for all  $(i, j)$  s.t  $A(i, j) \neq 0$  do
    Toss a biased coin with success probability  $p$ 
    if success then
         $\hat{A}(i, j) \leftarrow \frac{A(i, j)}{p}$ 
    end if
end for
{ Stage 2: EIGENTRIANGLE }
 $\lambda_1 \leftarrow LanczosMethod(\hat{A}, 1)$ 
 $\vec{\Lambda} \leftarrow [\lambda_1]$ 
 $i \leftarrow 2$  {initialize  $i, \vec{\Lambda}$ }
repeat
     $\lambda_i \leftarrow LanczosMethod(\hat{A}, i)$ 
     $\vec{\Lambda} \leftarrow [\vec{\Lambda} \lambda_i]$ 
     $i \leftarrow i + 1$ 
until  $0 \leq \frac{|\lambda_i^3|}{\sum_{j=1}^i \lambda_j^3} \leq tol$ 
 $\Delta'(G) \leftarrow \frac{1}{6} \sum_{j=1}^{i-1} \lambda_j^3$ 
return  $\Delta'(G)$ 

```

**Algorithm 1:** The SPARSIFYINGEIGENTRIANGLE algorithm

rank approximation of the adjacency matrix gives a good estimate of the number of triangles in the graph. (b) We can keep a small percentage of the total edges of the graph and keep the top eigenvalues of the sparsified graph very close to the ones of the initial graph.

Our algorithm, SPARSIFYINGEIGENTRIANGLE, takes two parameters, the  $tol$  parameter and the sparsification parameter  $p$ , as can be seen from the pseudocode. As we see the algorithm works in two stages. First it performs one pass over the edges of the graph (non-zero entries of the adjacency matrix). For each edge we toss a biased coin with probability  $p$  of keeping an edge. If the edge is kept we attach a weight of  $\frac{1}{p}$  to that edge. Therefore, in expectation, at the end of the pass, we keep  $pm$  edges in total, all of them weighted with value equal to  $\frac{1}{p}$ . After the sparsification stage, the algorithm moves into the EIGENTRIANGLE stage. In this phase, we perform an iterative eigen-computation on the sparse adjacency matrix  $\hat{A}$  until we observe that the cube of the absolute value of the eigenvalue being computed is significantly smaller than the estimate of triangles made until then. This is exactly the intuition behind the  $tol$  parameter: stop iterating when the eigenvalue just computed does not contribute significantly to the estimate. The intuition behind this stopping criterion is shown in figure 1.

The algorithm used in the eigen-computation is the Lanczos method, an efficient method for finding the top eigenvalues in sparse, symmetric matrices. Golub and Van Loan in [17] provide an excellent treatment of Lanczos method. One of the important properties of Lanczos is that the number of passes depends on the spectral gaps of the eigenvalues to be computed. In our case due to the power-law that holds for the top-eigenvalues [13, 21, 8] Lanczos converges fast [18].

When the iteration stops, the algorithm outputs the estimate of the number of triangles in the graph as the sum of the cubes of the computed eigenvalues divided by 6, in accordance with the EIGENTRIANGLEtheorem [25].

Our algorithm, SPARSIFYINGEIGENTRIANGLE works for many real-world networks very fast in practice, due to the following properties:

1. Top eigenvalues follow a power law which implies the following desirable properties:
  - Few eigenvalues contribute a lot to the number of triangles.
  - Cubes amplify this even more.
  - Lanczos method converges fast.

2. The rest of the eigenvalues are almost symmetric around zero and therefore they can be discarded since the sum of their cubes will not contribute significantly to the number of triangles.

These properties are illustrated in figure 2 where we see the Gershgorin circles which are simple upper bounds on the eigenvalues, and are in accordance with the observations made by Farkas, Derenyi, Barabasi, and Vicsek in [14].

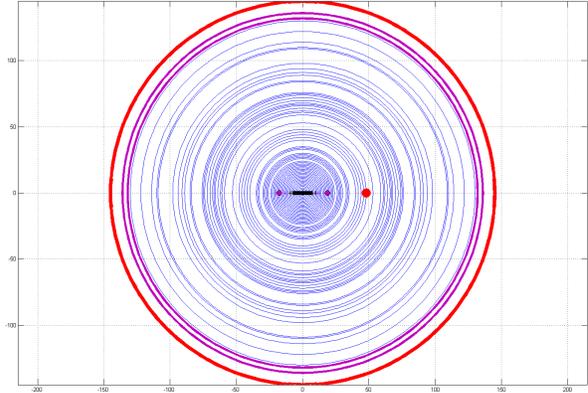
## 4 Experiments

The graphs we used in our experiments are described in Table 2. We implemented all our algorithms in MATLAB and the experiments ran on a 4GB RAM, Intel(R) Core(TM)2 Duo CPU at 2.4GHz Windows Vista machine. We report the results of our method in terms of the speedup vs. the NODEITERATOR algorithm.

Nodes	Edges	Description
404,733	2,110,078	Flickr
13,332	148,038	Reuters news, Sept 9-11,2001.
13,579	37,448	AS Oregon
23,389	47,448	CAIDA AS

**Table 2. Order and size of networks used.**

In order to avoid running into dilemmas regarding the choice of the tolerance parameter, we adapt the empirical



**Figure 2. Gershgorin circles for a small graph (airports) on the *complex* plane. As we see from the plot, all eigenvalues lie on the real axis due to the symmetry of the adjacency matrix, most of them are almost symmetric around zero, and a few ones are detached from the rest. The top eigenvalue is denoted with red, the second and third with purple and so are the corresponding circles.**

rule-of-thumb from [25], where it was observed in a wide range of experiments that a) typically a 6.2 rank approximation per average is good enough to acquire more than 95% accuracy and b) the maximum number of eigenvalues needed was 23. Therefore, in our experiments we compute for each graph the top-30 eigenvalues, even if less eigenvalues can provide an accurate estimation. The results are shown in figure 3. For each dataset, we plot the accuracy and the speedup vs. the NODEITERATOR for the estimation resulting after computing one to thirty top eigenvalues. Similar results are obtained for other graphs of about the same size as well, which are omitted here due to the limited space. The plots presented are representative of what we can see when SPARSIFYINGEIGENTRIANGLE is run (e.g., different scenarios that can occur in the convergence of the estimate towards the real value).

These plots reveal the following facts: 1) Even when we keep via the Achlioptas-McSherry sparsification a small percentage, e.g, 10%, of the graph edges the eigenvalues remain very close to the real ones. 2) Few top eigenvalues are enough to get a good estimate of the total number of triangles in the graph. 3) Speedups, even for graphs with few tenths of thousand or few million edges, are important. 4) The expected trend of significant savings as the number of non-zeros elements of the matrix gets smaller is not clearly observed and this is due to the implementation properties of MATLAB’s eigensolver. However, this phenomenon should be eliminated when our algorithm is

applied to larger graphs.

## 5 Theoretical Ramifications: Counting Triangles with Sketches

In this section, we describe how one can use existing theory from the streaming algorithms literature to estimate the number of triangles in the graph  $G$ .

**Theorem 1** *Given a graph  $G(V, E)$  the number of triangles that edge  $(i, j) \in E$  participates in, is given by the following equation:*

$$\delta_{(i,j)} = \sqrt{d_i d_j} \left( 1 - \frac{\| \frac{A^{(i)}}{\sqrt{d_i}} - \frac{A^{(j)}}{\sqrt{d_j}} \|^2}{2} \right) \quad (4)$$

where  $A^{(k)}$  is the  $k$ -th column of the adjacency matrix  $A$  and  $d_k$  is the degree of node  $k$ .

**Proof 1** *The following three observations hold:*

1. *The inner product  $\langle A^{(i)}, A^{(j)} \rangle$  when  $(i, j) \in E(G)$  is equal to  $\delta_{(i,j)}$ .*
2. *The following simple identity which holds for any two vectors  $a, b$ :*

$$\langle a, b \rangle = \|a\| \|b\| \left( 1 - \frac{\| \frac{a}{\|a\|} - \frac{b}{\|b\|} \|^2}{2} \right)$$

3.  *$\|A^{(i)}\|^2 = d_i$  for  $i = 1 \dots n$ .*

*The theorem follows directly by the above simple observations.*

We use the above observation and existing work on preserving inner products [2, 3, 16] to estimate the number of triangles that each edge participates in. We provide the necessary definitions, the full analysis can be found in [16]. Let  $\epsilon$  be the distortion parameter,  $\delta$  the probability of algorithm's failure and  $\eta$  a failure threshold. An atomic sketch of a vector  $a$  is the dot product  $\langle a, r \rangle$  where  $r$  is a random vector containing  $\pm 1$  valued random variables. A sketch of the vector  $a$  is a vector  $b = [b_1, \dots, b_k]$  where  $k = O(\log \frac{n}{\delta})$  and each  $b_i$  is an atomic sketch, i.e.,  $b_i = \langle a, r_i \rangle$  for  $i = 1 \dots k$ .

**Lemma 1** *Let  $X$  be the  $O(\log \frac{1}{\delta})$ -wise median of  $O(\frac{1}{\epsilon^2})$ -wise means of independent copies of*

$$\left( \sum_m A_{mi} r_m^k \right) \left( \sum_m A_{mj} r_m^k \right) \quad (5)$$

*Then, the following holds:*

$$|X - \delta_{ij}| \leq \epsilon \sqrt{d_i d_j} \quad (6)$$

*with probability  $1 - \delta$ .*

The above analysis suggests the following simple procedure:

1. For each column  $A^{(i)}$   $i = 1 \dots n$  create a sketch of size  $O(\log \frac{n}{\delta})$ .
2. For each edge  $(i, j)$  estimate the number of triangles  $\delta_{ij}$  using the inner product of the sketches.

The above procedure does not produce a  $(1 \pm \epsilon)$  guarantee for the number of triangles. However, it can produce a good estimate of the number of triangles for dense graphs where the cosines of the vector columns are large, i.e., the vectors are not close to being orthogonal (this intuition is quantified by parameter  $\eta$ , see [16]). The exact details of how to store and generate vectors  $r_j$  are presented in [16] and the references therein.

## 6 Application: Triangle-Based Link Recommendation in Online Social Networks

Online social networks, e.g, Facebook, LinkedIn, typically offer a utility/application which suggests a small number of links to the user that are of potential interest. A natural problem is the following, known as link recommendation in literature; see [22, 20] as well as references therein:

- **Input:** Graph  $G(V, E)$  (undirected, unweighted), node  $i$ , budget of  $k$  link recommendations.
- **Output:** Nodes  $i_1, \dots, i_k$

Nodes  $i_1, \dots, i_k$  will be the recommended links for node  $i$ . A natural, simple criterion for link recommendation in social networks which is based on the fact that friends of friends tend to make good recommendations is the following: *Recommend the links which create as many triangles as possible.*

A simple algorithm implementing the above principle is the following:

1. Find  $S = V - N$ , where  $N = \{j_1, \dots, j_{d_i}\}$  is the set of the  $d_i$  neighbors of node  $i$ .
2. For every node  $v \in S$  compute the inner product  $\langle A^{(i)}, A^{(v)} \rangle$ .
3. Sort the  $|S|$  inner products and choose nodes  $v_1, \dots, v_k$  which result in the top- $k$  inner products.

Intuitively, when the inner product of the columns  $A^{(i)}, A^{(v)}$  is large, then nodes  $i, v$  share many common neighbors and thus node  $v$  is according to our proposed principle a good candidate for link recommendation. Regarding the first step of the proposed method, if the graph has a large number of nodes then a more efficient algorithm would be to find two step away neighbors' set  $S'$ ,  $S' \subseteq S$  and apply the above procedure on  $S'$ .

## 7 Conclusions

In this follow-up work, we introduced the SPARSIFYINGEIGENTRIANGLE, a fast, parallelizable algorithm that can be used in cases where the graph of interest does not fit in the main memory. The main idea of the algorithm is to use a low-rank approximation of the matrix which is generated via the Achlioptas-McSherry [1] sparsification of the adjacency matrix to compute the number of triangles based on the EIGENTRIANGLE algorithm [25]. Furthermore, we show that even when keeping 10% of the graph edges one can compute the number of triangles in a very accurate and fast way.

Finally, we introduced a natural method of making link recommendations in online social networks and a procedure for estimating the number of triangles using sketches. The experimental evaluation of both is a topic for future research in the context of real-world networks.

## References

- [1] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximation. In *STOC*, 2001.
- [2] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. pages 10–20. Press, 2002.
- [3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29, New York, NY, USA, 1996. ACM.
- [4] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] F. Chung, L. Lu, and V. Vu. Eigenvalues of random power law graphs. *Annals of Combinatorics*, 7(1):21–33, June 2003.
- [9] 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.
- [10] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.
- [11] P. Drineas and R. Kannan. Pass efficient algorithms for approximating large matrices. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 223–232, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.
- [12] 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.
- [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] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. In *In Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 370–378, 1998.
- [16] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. One-pass wavelet decompositions of data streams. *IEEE TKDE*, 15:2003, 2003.
- [17] G. Golub and C. Van Loan. *Matrix Computations*. JohnsHopkinsPress, Baltimore, MD, second edition, 1989.
- [18] D. J. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.
- [19] M. Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theor. Comput. Sci.*, 407(1-3):458–473, 2008.

- [20] D. Liben-Nowell. *An Algorithmic Approach to Social Networks*. Ph.D. thesis, Massachusetts Institute of Technology, Electrical Engineering and Computer Science Department, June 2005.
- [21] M. Mihail and C. Papadimitriou. the eigenvalue power law, 2002.
- [22] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E*, 64, 2001.
- [23] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.
- [24] Thomas Schank and Dorothea Wagner . DELIS-TR-0043 - finding, counting and listing all triangles in large graphs, an experimental study. techreport 0043, submitted, 2004.
- [25] C. Tsourakakis. Fast counting of triangles in large real networks, without counting: Algorithms and laws. In *ICDM*, 2008.
- [26] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos. Doulion: counting triangles in massive graphs with a coin. In J. F. E. IV, F. Fogelman-Souli, P. A. Flach, and M. Zaki, editors, *KDD*, pages 837–846. ACM, 2009.
- [27] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller. Approximate triangle counting. *CoRR*, abs/0904.3761, 2009. informal publication.
- [28] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
- [29] S. Wasserman and K. Faust. *Social network analysis*. Cambridge University Press, Cambridge, 1994.

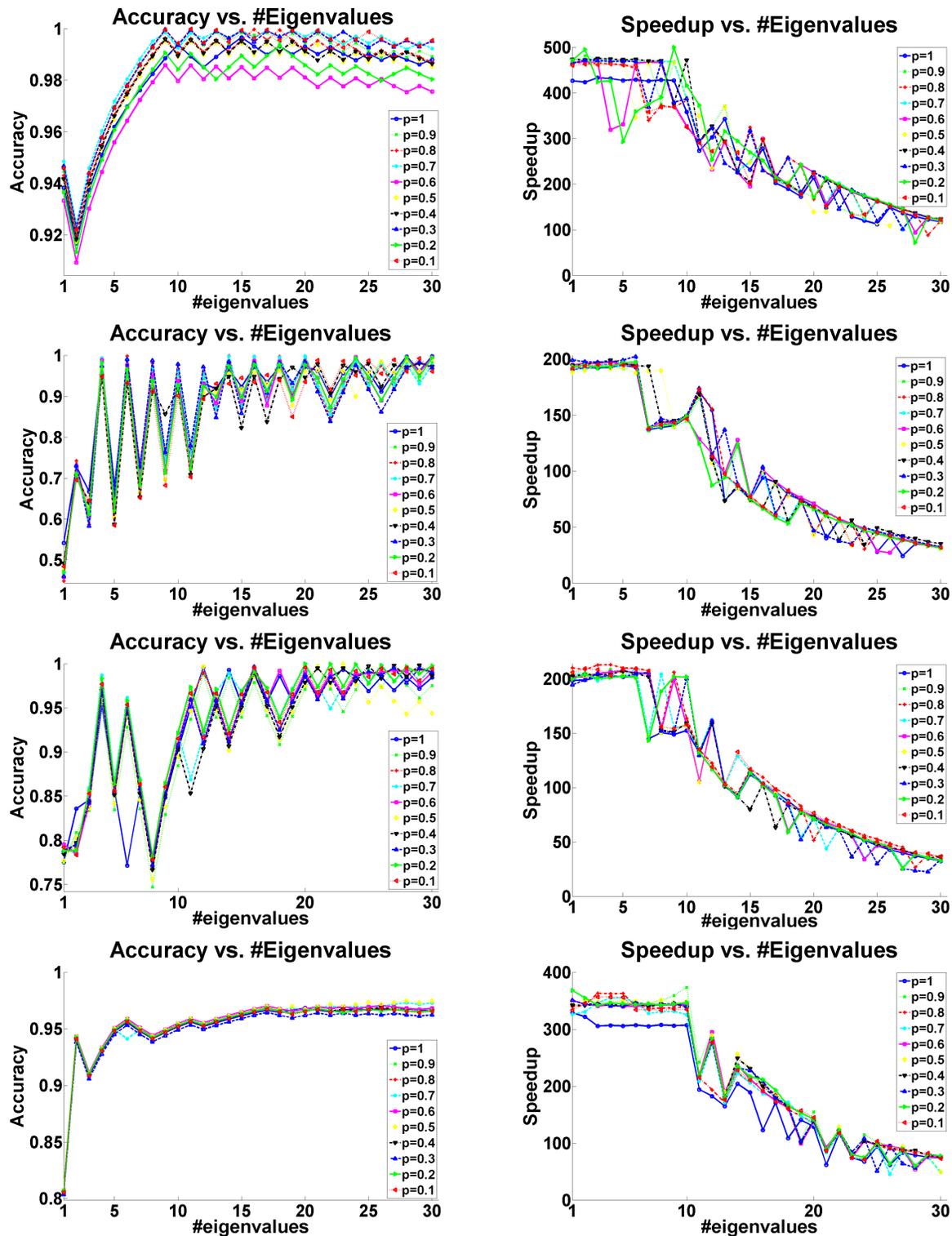


Figure 3. Experimental Results for four different datasets. (a) Reuters (b) AS CAIDA (c) AS Oregon (d) Flickr. Observe the following points: (1) the high accuracy obtained in the estimate after the top-5 eigenvalues, (2) the descending “oscillation” that moves towards the true value in different ways (3) the important speedups that we obtain while being accurate.