Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 15, no. 6, pp. 703–726 (2011)

Triangle Sparsifiers

Charalampos E. Tsourakakis[1] *Mihail N. Kolountzakis*[2]
Gary L. Miller[3]

[1]Department of Mathematical Sciences, Carnegie Mellon University, USA
[2]Department of Mathematics, University of Crete, Greece
[3]School of Computer Science, Carnegie Mellon University, USA

Abstract

In this work, we introduce the notion of *triangle sparsifiers*, i.e., sparse graphs which are approximately the same to the original graph with respect to the triangle count. This results in a practical triangle counting method with strong theoretical guarantees. For instance, for unweighted graphs we show a randomized algorithm for approximately counting the number of triangles in a graph G, which proceeds as follows: keep each edge independently with probability p, enumerate the triangles in the sparsified graph G' and return the number of triangles found in G' multiplied by p^{-3}. We prove that under mild assumptions on G and p our algorithm returns a good approximation for the number of triangles with high probability. Specifically, we show that if $p \geq \max\left(\frac{\text{polylog}(n)\Delta}{t}, \frac{\text{polylog}(n)}{t^{1/3}}\right)$, where n, t, Δ, and T denote the number of vertices in G, the number of triangles in G, the maximum number of triangles an edge of G is contained and our triangle count estimate respectively, then T is strongly concentrated around t:

$$\mathbf{Pr}\left[|T-t| \geq \epsilon t\right] \leq n^{-K}.$$

We illustrate the efficiency of our algorithm on various large real-world datasets where we obtain significant speedups. Finally, we investigate cut and spectral sparsifiers with respect to triangle counting and show that they are not optimal.

Submitted:	Reviewed:	Revised:	Accepted:
January 2011	July 2011	September 2011	October 2011
	Final:	Published:	
	October 2011	October 2011	
Article type:		Communicated by:	
Regular paper		D. Wagner	

We would like to thank Prof. Alan Frieze and Richard Peng for helpful discussions and the anonymous referees for their valuable comments. Research supported by NSF Grants No. CCF-1013110, No. IIS-0705359, No. CCF-0635257 and by the University of Crete Grant No. 2569.

E-mail addresses: ctsourak@math.cmu.edu (Charalampos E. Tsourakakis) kolount@math.uoc.gr (Mihail N. Kolountzakis) glmiller@cs.cmu.edu (Gary L. Miller)

1 Introduction

In recent years, a considerable amount of research has focused on the study of graph structures arising from technological, biological and sociological systems. Graphs are the tool of choice in modeling such systems since the latter are typically described as a set of pairwise interactions. Important examples of such datasets are the Internet graph (vertices are routers, edges correspond to physical links), the Web graph (vertices are web pages, edges correspond to hyperlinks), social networks (vertices are humans, edges correspond to friendships), information networks like Facebook and LinkedIn (vertices are accounts, edges correspond to online friendships), biological networks (vertices are proteins, edges correspond to protein interactions), math collaboration network (vertices are mathematicians, edges correspond to collaborations) and many more.

The number of triangles is a computationally expensive, crucial graph statistic in complex network analysis, in random graph models and in various important applications. In Section 2.2 we provide an extensive list of applications involving triangles. It is worth mentioning that triangles play an important role in theoretical computer science. Specifically, the problem of detecting whether a graph is triangle-free [3] arises in numerous combinatorial problems such as the minimum cycle detection problem [26], the approximate minimum vertex cover in planar graphs [5] and recognition of median graphs [25]. Recently, it was shown that detecting a triangle is equivalent to Boolean matrix multiplication under the notion of subcubic reductions [60] contrary to common belief [48].

In this paper, we contribute to the problem of counting triangles in large graphs. Specifically, we present a new randomized algorithm for approximately counting the number of triangles in a graph G. The algorithm proceeds as follows: keep each edge independently with probability p, enumerate the triangles in the sparsified graph G' and return the number of triangles found in G' multiplied by p^{-3}. We prove that under mild assumptions on G and p our algorithm returns a good approximation for the number of triangles with high probability. Specifically, we show that if $p \geq \max\left(\frac{\text{polylog}(n)\Delta}{t}, \frac{\text{polylog}(n)}{t^{1/3}}\right)$, where n, t, Δ, and T denote the number of vertices in G, the number of triangles in G, the maximum number of triangles an edge of G is contained and our triangle count estimate respectively, then T is strongly concentrated around t:

$$\mathbf{Pr}\left[|T - t| \geq \epsilon t\right] \leq n^{-K}$$

We illustrate the efficiency of our algorithm on various large real-world datasets where we achieve significant speedups. Furthermore, we investigate the performance of existing sparsification procedures namely the Spielman-Srivastava spectral sparsifier [46] and the the Benczúr-Karger cut sparsifier [8, 9] and show that they are not optimal/suitable with respect to triangle counting.

Our paper is organized as follows: Section 2 presents work related to triangle counting and provides the necessary theoretical preliminaries for the analysis of our algorithm. Section 3 presents our algorithm and our main theoretical result.

Section 4 shows the efficiency of our method on several large networks. Section 5 discusses properties of existing sparsification methods (cut sparsifiers [9] and spectral sparsifiers [46]) with respect to triangle counting. Finally, Section 6 concludes and provides future research directions.

2 Background

In Section 2.1 we introduce the notation that we use in this work. In Section 2.2 we present an extensive list of applications which involves triangles. In Section 2.3 we present existing work on the triangle counting problem. Finally in Section 2.4 we briefly present theoretical preliminaries for our work. Namely, the Kim-Vu concentration theorem, the Benczúr-Karger and the Spielman-Srivastava sparsifier and finally Foster's theorem.

2.1 Notation

For the rest of the paper we use the following symbols: $G([n], E)$ stands for an undirected simple graph with n vertices labeled as $1, 2, .., n$ and edge set E. Let $m = |E|$ be the number of edges in G. Furthermore, let t be the number of triangles, $deg(u)$ be the degree of vertex u, $\Delta(e)$ be the number of triangles edge e is contained in and $\Delta(v)$ be the number of triangles vertex v participates in. From the context, it shall always be clear whether we refer to a vertex or an edge. We define Δ to be equal to the maximum number of triangles that any edge e is contained in, i.e., $\Delta = \max_{e \in E(G)} \Delta(e)$. Finally, p is the sparsification parameter.

2.2 Applications

There are two main processes that generate triangles in a social network: *homophily* and *transitivity*. According to the former, people tend to choose friends with similar characteristics to themselves (e.g., race, education) [61, 55] and according to the latter friends of friends tend to become friends themselves [55]. These facts have several implications which we present in the following together with other applications of triangle counting. For example, recently Bonato, Hadi, Horn, Prałat and Wang [13] proposed the iterated local transitivity model which has several properties matching empirical properties of "real-world" networks such as skewed degree distribution, communities etc. In the following we provide an extensive list of applications which involves triangles and ranges from social networks to Computer Aided Design applications.

Clustering Coefficients and Transitivity of a Graph

Watts and Strogatz [57] in their influential paper proposed a simple model which explains several properties in social networks such as the abundance in triangles and the short paths among any pair of nodes. Their model combines the idea of homophily which leads to the wealth of triangles in the network and the idea of weak ties which create short paths. In order to quantify the homophily, they introduce the definitions of the clustering coefficient of a vertex and the graph, see Definition 1. The definition of the transitivity $T(G)$ of a graph G, introduced by Newman et al. [39] is closely related to the clustering coefficient and measures the probability that two neighbors of any vertex are connected. It is worth pointing out that the authors of [39] erroneously claim that $C(G)$ is the same with $T(G)$, see also [45].

Definition 1 (Clustering Coefficient) *The clustering coefficient $C(v)$ of vertex $v \in V(G)$ is defined as*

$$C(v) = \frac{\Delta(v)}{\binom{deg(v)}{2}}. \tag{1}$$

The clustering coefficient $C(G)$ of graph G is the average of $C(v)$ over all $v \in V(G)$.

Definition 2 (Transitivity) *The transitivity $T(G)$ of a graph G is defined as*

$$T(G) = \frac{3 \times t}{\sum_{v \in V(G)} \binom{deg(v)}{2}}. \tag{2}$$

Uncovering Hidden Thematic Structures

Eckmann and Moses [20] propose the use of the clustering coefficient for detecting subsets of web pages with a common topic. The key idea is that reciprocal links between pages indicate a mutual recognition/respect and then triangles due to their transitivity properties can be used to extend "seeds" to larger subsets of vertices with similar thematic structure in the Web graph. In other words, regions of the World Wide Web with high triangle density indicate common thematic structure, allowing the authors to extract useful meta-information. This idea has found applications in other scientific domains as well, e.g., in bioinformatics [43].

Exponential Random Graph Model

Frank and Strauss [21] proved under the assumption that two edges are dependent only if they share a common vertex that the sufficient statistics for Markov graphs are the counts of triangles and stars. Wasserman and Pattison [56] proposed the exponential random graph (ERG) model which generalized the Markov graphs [42]. In this model, the probability space is the set of all possible graphs on n vertices and the probability of a given graph g is given by

$$\mathbf{Pr}\left[g\right] \propto \exp\Big(\theta \cdot s(g)\Big)$$

where $s(g)$ is a vector of graph statistics, typically counts of certain subgraphs (e.g., $s(g) = (m, t)$) and θ is a vector of parameters. Triangles are frequently used as one of the graph statistics of the ERG model and counting them is necessary for parameter estimation, e.g., using MCMC procedures [11].

Spam Detection

Becchetti et al. [7] show that the distribution of triangles among spam hosts and non-spam hosts can be used as a feature for classifying a given host as spam or non-spam. The same result holds also for web pages, i.e., the spam and non-spam triangle distributions differ at a detectable level using standard statistical tests from each other.

Content Quality and Role Behavior Identification

Nowadays, there exist many online forums where acknowledged scientists participate, e.g., MathOverflow, CStheory Stackexchange and discuss problems of their fields. This yields significant information for researchers. Several interesting questions arise such as which participants comment on each other. This question including several others were studied in [58]. The number of triangles that a user participates was shown to play a critical role in answering these questions. For further applications in assessing the role behavior of users see [7].

Structural Balance and Status Theory

Balance theory appeared first in Heider's seminal work [24] and is based on the concept "the friend of my friend is my friend", "the enemy of my friend is my enemy" etc. [55]. To quantify this concept edges become signed, i.e., there is a function $c : E(G) \to \{+, -\}$. If all triangles are positive, i.e., the product of the signs of the edges is $+$, then the graph is balanced. Status theory is based on interpreting a positive edge (u, v) as u having lower status that v, while the negative edge (u, v) means that u regards v as having a lower status than himself/herself. Recently, Leskovec et al. [35] have performed experiments to quantify which of the two theories better apply to online social networks and predict signs of incoming links. Their algorithms require counts of signed triangles in the graph.

Microscopic Evolution of networks

Leskovec et al. [36] present an extensive experimental study of network evolution using detailed temporal information. One of their findings is that as edges arrive in the network, they tend to close triangles, i.e., connect people with common friends.

Community Detection

Counting triangles is also used in community detection algorithms. Specifically Berry et al. use triangle counting to deduce the edge support measure in their community detection algorithm [10].

Link Recommendation

Online social networks (e.g.,Facebook, LinkedIn) typically have link recommendation applications. Proposing edges which create as many triangles as possible is a possible link recommendation mechanism [53].

Motif Detection

Triangles are abundant not only in social networks but in biological networks [37, 62]. This fact can be used e.g., to correlate the topological and functional properties of protein interaction networks [62].

CAD applications

Fudos and Hoffman [23] introduced a graph-constructive approach to solving systems of geometric constraints, a problem which arises frequently in Computer Aided Design (CAD) applications. One of the steps of their algorithm computes the number of triangles in an appropriately defined graph.

2.3 Existing work

There exist two categories of triangle counting algorithms, exact and approximate ones. It is worth noting that for most of the applications described in Section 2.2 the exact number of triangles in not crucial. Hence, approximate counting algorithms which are fast and output a high quality estimate are desirable for the practical applications in which we are interested in this work.

2.3.1 Exact Counting

Naive triangle counting by checking all triples of vertices takes $O(n^3)$ units of time. The state of the art algorithm is due to Alon, Yuster and Zwick [2] and runs in $O(m^{\frac{2\omega}{\omega+1}})$, where currently the fast matrix multiplication exponent ω is 2.371 [18]. Thus, the Alon, Yuster, Zwick (AYZ) algorithm currently runs in $O(m^{1.41})$ time. It is worth mentioning that from a practical point of view algorithms based on matrix multiplication are not used due to the high memory requirements. Even for medium sized networks, matrix-multiplication based algorithms are not applicable. We use this partitioning idea in Section 3 to obtain state of the art results on approximate triangle counting. Itai and Rodeh in 1978 showed an algorithm which finds a triangle in any graph in $O(m^{\frac{3}{2}})$ [26]. This algorithm can be extended to list the triangles in the graph with the same time complexity. Chiba and Nishizeki showed that triangles can be found in time $O(m\alpha(G))$ where $\alpha(G)$ is the *arboricity* of the graph. Since $\alpha(G)$ is at most $O(\sqrt{m})$ their algorithm runs in $O(m^{3/2})$ in the worst case [16]. For special types of graphs more efficient triangle counting algorithms exist. For instance in planar graphs, triangles can be found in $O(n)$ time [16, 26, 41].

Even if listing algorithms solve a more general problem than the counting one, they are preferred in practice for large graphs, due to the smaller memory requirements compared to the matrix multiplication based algorithms. Simple representative algorithms are the node- and the edge-iterator algorithms. The former counts for each node number of triangles it is involved in, which is equivalent to the number of edges among its neighbors, whereas in the latter, the algorithm counts for each edge (i, j) the common neighbors of nodes i, j. Both of these algorithms have the same asymptotic complexity $O(mn)$, which in dense graphs results in $O(n^3)$ time, the complexity of the naive counting algorithm. Practical improvements over this family of algorithms have been achieved using various techniques, such as hashing and sorting by the degree [34, 44].

2.3.2 Approximate Counting

On the approximate counting side, most of the 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 arrive 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 $(1+\epsilon)$ approximation of the number of triangles with high probability by making only 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 [45], 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 is the number of triples with i edges and outputs as the estimate of triangles the random variable T_3' equaling to the fractions of triples picked that form triangles times the total number of triples $(\binom{n}{3})$, then

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

with probability at least $1 - \delta$. This is not suitable when $T_3 = o(n^2)$.

In [6] 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 [1] (a.k.a. AMS algorithms) to proceed. The key is that the triangle computation reduces to estimating the zero-th, first and second frequency moments, which can be done efficiently. Furthermore, as the authors suggest their algorithm is efficient only on graphs with $\Omega(n^2/\log\log n)$ triangles, i.e., triangle dense graphs as in the naive sampling. The AMS algorithms are also used by [28], where simple sampling techniques are used, such as choosing an edge from the stream at random and checking how many common neighbors its two endpoints share considering the subsequent edges in the stream.

Along the same lines, Buriol et al. [15] proposed two space-bounded sampling algorithms to estimate the number of triangles. Again, the underlying sampling procedures are simple. For instance, in 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 the best of our knowledge. The three-pass algorithm presented therein, counts in the first pass the number of edges, in the second pass it samples uniformly at random an edge (i, j) and a node $k \in V - \{i, j\}$ and in the third pass it tests whether the edges $(i, k), (k, j)$ are present in the stream. The number of samples r needed to obtain an ϵ-approximation with probability $1 - \delta$ is

$$r = O\left(\log\left(\frac{1}{\delta}\right) \frac{T_1 + 2T_2 + 3T_3}{T_3 \epsilon^2} \right) = O\left(\log\left(\frac{1}{\delta}\right) \frac{mn}{t\epsilon^2} \right).$$

Even if the term T_0 in the nominator is missing[1] compared to the naive sampling, the graph has still to be fairly dense with respect to the number of triangles in order to get an ϵ approximation with high probability. Buriol et al. [15] show how to turn the three-pass algorithm into a single pass algorithm for the edge stream representation and similarly they provide a three- and one-pass algorithm for the incidence stream representation. In [52] an algorithm which tosses a coin independently for each edge with probability p to keep the edge and probability $q = 1-p$ to throw it away is proposed, which however sets p to a constant. The sampling scheme of [52] can be combined with existing sampling schemes [15] via a degree based partitioning to yield improved theoretical and practical performance as it was shown in [31]. Recently, a more efficient sampling scheme was proposed by Pagh and Tsourakakis [40].

Another line of work is based on linear algebraic arguments. Specifically, in the case of "power-law" networks it was shown in [50] that the spectral counting of triangles can be efficient due to their special spectral properties [17]. This idea was further extended in [51] using the randomized SVD approximation algorithm by [19]. In [7] the semi-streaming model for counting triangles is introduced, which allows $\log n$ passes over the edges. The key observation is that since counting triangles reduces to computing the intersection of two sets, namely the induced neighborhoods of two adjacent nodes, ideas from locality sensitivity hashing [14] are applicable to the problem. More recently, Avron proposed a new approximate triangle counting method based on a randomized algorithm for trace estimation [4].

2.4 Theoretical Preliminaries

2.4.1 Kim-Vu Concentration of Measure

For the purposes of this work, let $Y = Y(t_1, \ldots, t_m)$ be a positive polynomial of m Boolean variables $[t_i]_{i=1..m}$ which are independent. A common task in combinatorics is to show that Y is concentrated around its expected value,

[1]Notice that $m(n-2) = T_1 + 2T_2 + 3T_3$ and $t = T_3$.

e.g., [30]. In the following we state the necessary definitions and the main concentration result which we will use in our method. Y is totally positive if all of its coefficients are non-negative. Y is homogeneous if all of its monomials have the same degree and we call this value the degree of the polynomial. Given any multi-index $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbb{Z}_+^m$, define the partial derivative $\partial^\alpha Y = (\frac{\partial}{\partial t_1})^{\alpha_1} \dots (\frac{\partial}{\partial t_m})^{\alpha_m} Y(t_1, \dots, t_m)$ and denote by $|\alpha| = \alpha_1 + \cdots \alpha_m$ the order of α. For any order $d \geq 0$, define $\mathbb{E}_d(Y) = \max_{\alpha:|\alpha|=d} \mathbb{E}(\partial^\alpha Y)$ and $\mathbb{E}_{\geq d}(Y) = \max_{d' \geq d} \mathbb{E}_{d'}(Y)$.

Now, we refer to the main theorem of Kim and Vu of [29, §1.2] as phrased in Theorem 1.1 of [54] or as Theorem 1.36 of [49].

Theorem 1 *There is a constant c_k depending on k such that the following holds. Let $Y(t_1, \dots, t_m)$ be a totally positive polynomial of degree k, where t_i can have arbitrary distribution on the interval* $[0, 1]$. *Assume that:*

$$\mathbb{E}[Y] \geq \mathbb{E}_{\geq 1}(Y) \tag{3}$$

Then for any $\lambda \geq 1$:

$$\mathbf{Pr}\left[|Y - \mathbb{E}[Y]| \geq c_k \lambda^k (\mathbb{E}[Y]\, \mathbb{E}_{\geq 1}(Y))^{1/2}\right] \leq e^{-\lambda + (k-1)\log m}. \tag{4}$$

The Kim-Vu theorem is an important concentration result since it allows us to obtain strong concentration when the polynomial of interest is not *smooth*. Typically, when a polynomial Y is smooth, it is strongly concentrated. By smoothness one usually means a small Lipschitz coefficient or in other words, when one changes the value of one variable t_j, the value Y changes no more than a constant. However, as stated in [54] this is restrictive in many cases. Thus one can demand "average smoothness" as defined in [54] which is quantified via the expectation of partial derivatives of any order.

2.4.2 Graph Sparsifiers

A sparsifier of a graph $G(V, E, w)$ is a sparse graph H that is similar to G in some useful notion. In Section 2.4.2 we describe the Benczúr-Karger cut sparsifier [8, 9] and in Section 2.4.2.the Spielman-Srivastava spectral sparsifier [46].

Benczúr-Karger Sparsifier Benczúr and Karger introduced in [8] the notion of cut sparsification to accelerate cut algorithms whose running time depends on the number of edges. Using a non-uniform sampling scheme they show that given a graph $G(V, E, w)$ with $|V| = n, |E| = m$ and a parameter ϵ there exists a graph $H(V, E', w')$ with $O(n \log(n)/\epsilon^2)$ edges such that the weight of every cut in H is within a factor of $(1 \pm \epsilon)$ of its weight in G. Furthermore, they provide a nearly-linear time algorithm which constructs such a sparsifier. The key quantity used in the sampling scheme of Benczúr and Karger is the strong connectivity $c_{(u,v)}$ of an edge $(u, v) \in E$ [8, 9]. The latter quantity is defined to be the maximum value k such that there is an induced subgraph G_0 of G containing both u and v, and every cut in G_0 has weight at least k.

Spielman-Srivastava Sparsifier In [47] Spielman and Teng introduced the notion of a spectral sparsifier in order to strengthen the notion of a cut sparsifier. A quantity that plays a key role in spectral sparsifiers is the *effective resistance*. The term effective resistance comes from electrical network analysis, see Chapter IX in [12]. In a nutshell, let $G(V, E, w)$ be a weighted graph with vertex set V, edge set E and weight function w. We call the weight $w(e)$ *resistance* of the edge e. We define the *conductance* $r(e)$ of e to be the inverse of the resistance $w(e)$. Let $\mathcal{G}$ be the resistor network constructed from $G(V, E, w)$ by replacing each edge e with an electrical resistor whose electrical resistance is $w(e)$. Typically, in $\mathcal{G}$ vertices are called *terminals*, a convention that emphasizes the electrical network perspective of a graph G. The *effective resistance* $R(i, j)$ between two vertices i, j is the electrical resistance measured across vertices i and j in $\mathcal{G}$. Equivalently, the effective resistance is the potential difference that appears across terminals i and j when we apply a unit current source between them. Finally, *effective conductance* $C(i, j)$ between two vertices i, j is defined to by $C(i, j) = R(i, j)^{-1}$.

Spielman and Srivastava in their seminal work [46] proposed to include each edge of G in the sparsifier H with probability proportional to its effective resistance. They provide a nearly-linear time algorithm that produces spectral sparsifiers with $O(n \log n)$ edges.

2.4.3 Foster's theorem

In Section 5 we use the following theorem, proved by Foster [22].

Theorem 2 *Let G be a connected graph of order n. Then*

$$\sum_{(u,v) \in E(G)} R(u, v) = n - 1.$$

3 Proposed Algorithm

Algorithm 1 Triangle Sparsifier

```
Require: Set of edges $E \subseteq \binom{[n]}{2}$ {Unweighted graph $G([n], E)$}
Require: Sparsification parameter $p$
  Pick a random subset $E'$ of edges such that the events $\{e \in E'\}$, for all $e \in E$
  are independent and the probability of each is equal to $p$.
  $t' \leftarrow$ count triangles on the graph $G'([n], E')$
  Return $T \leftarrow \frac{t'}{p^3}$
```

Our proposed algorithm *Triangle Sparsifier* is shown in Algorithm 1 (see also for a preliminary version [52]). The algorithm takes an unweighted, simple graph $G(V, E)$, where without loss of generality we assume that the nodes are numbered from $1, \ldots, n$, i.e., $V = [n]$ and a sparsification parameter $p \in (0, 1)$

as input. The algorithm first chooses a random subset E' of the set E of edges. The random subset is such that the events

$$\{e \in E'\}, \text{for all } e \in E,$$

are independent and the probability of each is equal to p. Then, any triangle counting algorithm can be used to count triangles on the sparsified graph with edge set E'. Clearly, the expected size of E' is pm where $m = |E|$. The output of our algorithm is the number of triangles in the sparsified graph multiplied by $\frac{1}{p^3}$, or equivalently we are counting the number of weighted triangles in G' where each edge has weight $\frac{1}{p}$. It follows immediately that the expected value $\mathbb{E}[T]$ of our estimate is the number of triangles in G, i.e., t. Our main theoretical result is the following theorem:

Theorem 3 *Suppose G is an undirected graph with n vertices, m edges and t triangles. Let also Δ denote the size of the largest collection of triangles with a common edge. Let G' be the random graph that arises from G if we keep every edge with probability p and write T for the number of triangles of G'. Suppose that $\gamma > 0$ is a constant and*

$$\frac{pt}{\Delta} \geq \log^{6+\gamma} n, \quad \textit{if } p^2\Delta \geq 1, \tag{5}$$

and

$$p^3 t \geq \log^{6+\gamma} n, \quad \textit{if } p^2\Delta < 1. \tag{6}$$

for $n \geq n_0$ sufficiently large. Then

$$\mathbf{Pr}\left[|T - \mathbb{E}[T]| \geq \epsilon \mathbb{E}[T]\right] \leq n^{-K}$$

for any constants $K, \epsilon > 0$ and all large enough n (depending on K, ϵ and n_0).

Proof: Write $X_e = 1$ or 0 depending on whether the edge e of graph G survives in G'. Then $T = \sum_{\Delta(e,f,g)} X_e X_f X_g$ where $\Delta(e,f,g) = \mathbf{1}$ (edges e, f, g form a triangle). Clearly $\mathbb{E}[T] = p^3 t$.

Refer to Theorem 1. We use T in place of Y, $k = 3$.

We have

$$\mathbb{E}\left[\frac{\partial T}{\partial X_e}\right] = \sum_{\Delta(e,f,g)} \mathbb{E}[X_f X_g] = p^2 |\Delta(e)|.$$

We first estimate the quantities $\mathbb{E}_j(T), j = 0,1,2,3$, defined before Theorem 1. We get

$$\mathbb{E}_1(T) = p^2\Delta. \tag{7}$$

We also have

$$\mathbb{E}\left[\frac{\partial^2 T}{\partial X_e \partial X_f}\right] = p\mathbf{1}\left(\exists g : \Delta(e,f,g)\right),$$

hence

$$\mathbb{E}_2(T) \leq p. \tag{8}$$

Obviously, $\mathbb{E}_3(T) \leq 1$.

Hence

$$\mathbb{E}_{\geq 3}(T) \leq 1,\ \mathbb{E}_{\geq 2}(T) \leq 1,$$

and

$$\mathbb{E}_{\geq 1}(T) \leq \max\left\{1, p^2\Delta\right\},\ \mathbb{E}_{\geq 0}(T) \leq \max\left\{1, p^2\Delta, p^3 t\right\}.$$

• Case 1 ($p^2\Delta < 1$):
We get $\mathbb{E}_{\geq 1}(T) \leq 1$, and from (6), $\mathbb{E}\left[T\right] \geq \mathbb{E}_{\geq 1}(T)$.

• Case 2 ($p^2\Delta \geq 1$):
We get $\mathbb{E}_{\geq 1}(T) \leq p^2\Delta$ and, from (5), $\mathbb{E}\left[T\right] \geq \mathbb{E}_{\geq 1}(T)$.

We get, for some constant $c_3 > 0$, from Theorem 1:

$$\mathbf{Pr}\left[|T - \mathbb{E}\left[T\right]| \geq c_3\lambda^3(\mathbb{E}\left[T\right]\mathbb{E}_{\geq 1}(T))^{1/2}\right] \leq e^{-\lambda + 2\log n}. \tag{9}$$

Notice that since in both cases we have $\mathbb{E}\left[T\right] \geq \mathbb{E}_{\geq 1}(T)$.

We now select λ so that the lower bound inside the probability on the left-hand side of (9) becomes $\epsilon\mathbb{E}\left[T\right]$. In Case 1 we pick

$$\lambda = \frac{\epsilon^{1/3}}{c_3^{1/3}}(p^3 t)^{1/6}$$

while in Case 2

$$\lambda = \frac{\epsilon^{1/3}}{c_3^{1/3}}\left(\frac{pt}{\Delta}\right)^{1/6}$$

to get

$$\mathbf{Pr}\left[|T - \mathbb{E}\left[T\right]| \geq \epsilon\mathbb{E}\left[T\right]\right] \leq \exp(-\lambda + 2\log n) \tag{10}$$

Since $\lambda \geq (K+2)\log n$ follows from our assumptions (5) and (6) if n is sufficiently large, we get $\mathbf{Pr}\left[|T - \mathbb{E}\left[T\right]| \geq \epsilon\mathbb{E}\left[T\right]\right] \leq n^{-K}$, in both cases. □

Complexity Analysis The expected running time of edge sampling is sublinear, i.e., $O(pm)$, see Claim 1. The complexity of the counting step depends on which algorithm we use to count triangles[2]. For instance, if we use [2] as our triangle counting algorithm, the expected running time of Triangle Sparsifier is $O(pm + (pm)^{\frac{2\omega}{\omega+1}})$, where ω currently is 2.371 [18]. If we use the node-iterator (or any other standard listing triangle algorithm) the expected running time is $O(pm + p^2\sum_i d_i^2)$.

Claim 1 (Sparsification in sublinear expected time) *The edge sampling can run in $O(pm)$ expected time.*

[2]We assume for fairness that we use the same algorithm in both the original graph G and the sparsified graph G' to count triangles.

Proof: We do not "toss a p-coin" m times in order to construct E'. This would be very wasteful if p is small. Instead we construct the random set E' with the following procedure which produces the right distribution. Observe that the number X of unsuccessful events, i.e., edges which are not selected in our sample, until a successful one follows a geometric distribution. Specifically, $\mathbf{Pr}\left[X = x\right] = (1-p)^{x-1}p$. To sample from this distribution it suffices to generate a uniformly distributed variable U in $[0, 1]$ and set $X \leftarrow \left\lceil \frac{\ln U}{1-p} \right\rceil$. Clearly the probability that $X = x$ is equal to $\mathbf{Pr}\left[(1-p)^{x-1} > U \geq (1-p)^{x}\right] = (1-p)^{x-1} - (1-p)^{x} = (1-p)^{x-1}p$ as required. This provides a practical and efficient way to pick the subset E' of edges in subliner expected time $O(pm)$. For more details see [33]. $\square$

Expected Speedup: The expected speedup with respect to the triangle counting task depends on the triangle counting subroutine that we use. If we use [2] as our subrouting which is the fastest known algorithm the expected speedup is $p^{-\frac{2\omega}{\omega+1}}$, i.e., currently $p^{-1.41}$ where ω currently is 2.371 [18]. As already outlined, in practice $p^{-\frac{2\omega}{\omega+1}}$, i.e., currently $p^{-1.41}$, and p^{-2} respectively.

Discussion: This theorem states the important result that the estimator of the number of triangles is concentrated around its expected value, which is equal to the actual number of triangles t in the graph under mild conditions on the triangle density of the graph. The mildness comes from condition (5): picking $p = 1$, given that our graph is not triangle-free, i.e., $\Delta \geq 1$, gives that the number of triangles t in the graph has to satisfy $t \geq \Delta \log^{6+\gamma} n$. This is a mild condition on t since $\Delta \leq n$ and thus it suffices that $t \geq n \log^{6+\gamma} n$ (after all, we can always add two dummy connected nodes that connect to every other node, as in Figure 1(a), even if in empirically Δ is smaller than n). The critical quantity besides the number of triangles t, is Δ. Intuitively, if the sparsification procedure throws away the common edge of many triangles, the triangles in the resulting graph may differ significantly from the original. A significant problem is the choice of p for the sparsification. Conditions (5) and (6) tell us how small we can afford to choose p, but the quantities involved, namely t and Δ, are unknown. We discuss a practical algorithm using a doubling procedure in Section 4.4. Furthermore, our method justifies significant speedups. For a graph G with $t \geq n^{3/2+\epsilon}$ and $\Delta \sim n$, we get $p = n^{-1/2}$ implying a linear expected speedup if we use a practical exact counting method as the node iterator. Finally, it is worth pointing out that *Triangle Sparsifier* essentially outputs a sparse graph $H(V, E', w)$ with $w = 1/p$ for all edges $e \in E'$ which approximates $G(V, E)$ with respect to the count of triangles (a triangle formed by the edges (e_1, e_2, e_3) in a weighted graph counts for $w(e_1)w(e_2)w(e_3)$ unweighted triangles). As we shall see in Section 6 *Triangle Sparsifier* is not recommended for weighted graphs.

Description	Availability
SNAP	`http://snap.stanford.edu/`
UF Sparse Matrix Collection	`http://www.cise.ufl.edu/research/sparse`
Max Planck [38]	`http://socialnetworks.mpi-sws.org/`

Table 1: Dataset sources.

4 Experimental Results

In this Section we present our experimental findings. Specifically, in Section 4.1 we describe the datasets we used, in Section 4.2 we give details with respect to the experimental setup and in Section 4.3 the experimental results.

4.1 Datasets

The graphs we used with the exceptions of Livejournal-links and Flickr are available on the Web. Table 1 summarizes the data resources. We preprocessed the graphs by first making them undirected and removing all self-loops. Furthermore, a common phenomenon was to have multiple edges in the edge file, i.e., a file whose each line corresponds to an edge, despite the fact that the graphs were claimed to be simple. Those multiple edges were removed. Table 2 summarizes the datasets we used after the preprocessing.

Name (Abbr.)	Nodes	Edges	Triangle Count
$\odot$ AS-Skitter (AS)	1,696,415	11,095,298	28,769,868
$\star$Flickr (FL)	1,861,232	15,555,040	548,658,705
$\star$Livejournal-links (LJ)	5,284,457	48,709,772	310,876,909
$\star$Orkut-links (OR)	3,072,626	116,586,585	621,963,073
$\star$Soc-LiveJournal (SL)	4,847,571	42,851,237	285,730,264
$\star$Youtube (YOU)	1,157,822	2,990,442	4,945,382
$\diamond$Web-EDU (WE)	9,845,725	46,236,104	254,718,147
$\diamond$Web-Google (WG)	875,713	3,852,985	11,385,529
$\diamond$Wikipedia 2005/11 (W0511)	1,634,989	18,540,589	44,667,095
$\diamond$Wikipedia 2006/9 (W0609)	2,983,494	35,048,115	84,018,183
$\diamond$Wikipedia 2006/11 (W0611)	3,148,440	37,043,456	88,823,817
$\diamond$Wikipedia 2007/2 (W0702)	3,566,907	42,375,911	102,434,918

Table 2: Datasets used in our experiments. Abbreviations are included. Symbol $\odot$ stands for Autonomous Systems graphs, $\star$ for online social networks and $\diamond$ for Web graphs. Notice that the networks with the highest triangle counts are online social networks (Flickr, Livejournal, Orkut), verifying the folklore that online social networks are abundant in triangles.

4.2 Experimental Setup

The experiments were performed on a single machine, with Intel Xeon CPU at 2.83 GHz, 6144KB cache size and and 50GB of main memory. The algorithm was implemented in C++, and compiled using gcc version 4.1.2 and the -O3 optimization flag. Time was measured by taking the user time given by the linux time command. IO times are included in that time since the amount of memory operations performed in setting up the graph is non-trivial. However, we use a modified IO routine that's much faster than the standard C/C++ scanf. Furthermore, as we mentioned in Section 3 picking a random subset of expected size $p|S|$ from a set S can be done in expected sublinear time [33]. A simple way to do this in practice is to generate the differences between indices of entries retained. This allows us to sample in a sequential way and also results in better cache performance. As a competitor we use an implementation of ours of the 1 pass algorithm of [15, §2.2].

4.3 Experimental Results

Table 2 shows the count of triangles for each graph used in our experiments. Notice that Orkut, Flickr and Livejournal graphs have ∼622M, 550M and 311M triangles respectively. This confirms the folklore that online social networks are abundant in triangles. Table 3 shows the results we obtain for $p = 0.1$ over 5 trials. All running times are reported in seconds. The first column shows the running time for the exact counting algorithm over 5 runs. Standard deviations are neglibible for the exact algorithm and therefore are not reported. The second and third column show the error and running time averaged over 5 runs for each dataset (two decimal digits of accuracy). Standard deviations are also included (three decimal digits of accuracy). The last column shows the running time averaged over 5 runs for the 1-pass algorithm as stated in [15, §2.2] and the standard deviations. For each dataset the number of samples needed by the 1-pass algorithm was set to a value that achieves *at most* as good accuracy as the ones achieved by our counting method. Specifically, for any dataset, if $\alpha, \beta(\%)$ are the errors obtained by our algorithm and the Buriol et al. algorithm, we "tune" the number of samples in the latter algorithm in such way that $\alpha \leq \beta \leq \alpha + 1\%$. Even by favoring in this way the 1-pass algorithm of Buriol et al. [15], one can see that the running times achieved by our method are consistently better. However, it is important to outline once again that our method and other triangle counting methods can be combined. For example, in [31] it was shown that Triangle Sparsifiers and other sampling methods can be combined to obtain a superior performance both in practice and theoretically by improving the sampling scheme of Buriol et al. [15]. This was achieved by distinguish vertices into two subsets according to their degree and using two sampling schemes, one for each subset [31]. We also tried other competitors, but our running times outperform them significantly. For example, even the exact counting method outperforms other approximate counting methods. As we show in Section 4.4 smaller values of p values work as well and these can be found by a simple doubling-like procedure.

	Results			
	Exact	Triangle Sparsifier		Buriol et al. [15]
	Avg. time	Avg. err.% (std)	Avg. time (std)	Avg. time (std)
AS	4.45	2.60 (0.022)	0.79 (0.023)	2.72 (0.128)
FL	41.98	0.11 (0.003)	0.96 (0.014)	3.40 (0.175)
LJ	50.83	0.34 (0.001)	2.85 (0.054)	12.40 (0.250)
OR	202.01	0.60 (0.004)	5.60 (0.159)	11.71 (0.300)
SL	38.27	8.27 (0.006)	2.50 (0.032)	8.92 (0.115)
YOU	1.35	1.50 (0.050)	0.30 (0.002)	10.91 (0.130)
WE	8.50	0.70 (0.005)	2.79 (0.090)	6.56 (0.025)
WG	1.60	1.58 (0.011)	0.40 (0.004)	1.85 (0.047)
W0511	32.47	1.53 (0.010)	1.19 (0.020)	3.71 (0.038)
W0609	86.62	0.40 (0.055)	2.07 (0.014)	8.10 (0.040)
W0611	96.11	0.62 (0.008)	2.16 (0.042)	7.90 (0.090)
W0702	122.34	0.80 (0.015)	2.48 (0.012)	11.00 (0.205)

Table 3: Results of experiments averaged over 5 trials using $p = 0.1$. All running times are reported in seconds. The first column shows the running time for the exact counting algorithm averaged over 5 runs. The second and third column show the error and running time averaged over 5 runs for each dataset (two decimal digits of accuracy). Standard deviations are also included (three decimal digits of accuracy). The last column shows the running time averaged over 5 runs for the 1-pass algorithm as stated in [15, §2.2] and the corresponding standard deviations. The number of samples for each dataset was set to a value that achieves *at most* as good accuracy as the ones achieved by our counting method. See Section 4.3 for all the details.

4.4 The "Doubling" Algorithm

As we saw in Section 3, setting optimally the parameter p requires knowledge of the quantity we want to estimate, i.e., the number of triangles. To overcome this problem we observe that when we have concentration, the squared coefficient of variation $\frac{\mathrm{Var}[T]}{\mathbb{E}[T]^2}$ is "small". Furthermore, by the Bienayme-Chebyshev inequality and by the median boosting trick [27] it suffices to sample $\{T_1, \ldots, T_s\}$ where $s = O(\frac{\mathrm{Var}[T]}{\mathbb{E}[T]^2} \frac{1}{\epsilon^2} \ln \frac{1}{\delta})$ in order to obtain a $(1 \pm \epsilon)$ approximation $\mathbb{E}[T] = t$ with probability at least 1-δ. Hence, one can set a desired value for the number of samples s and of the failure probability δ and calculate the expected error $\epsilon = O(\sqrt{\frac{\mathrm{Var}[T]}{\mathbb{E}[T]^2} \frac{1}{s} \ln \frac{1}{\delta}})$. If this value is significantly larger than the desired error threshold then one increases p and repeats the same procedure until the stopping criterion is satisfied. One way one can change p is to use the multiplicative rule $p \leftarrow cp$, where $c > 1$ is a constant. For example, if $c = 2$ then we have a doubling procedure. Notice that we've placed the word doubling in the title of this section in quotes in order to emphasize that one may use any $c > 1$ to change p from one round to the next.

For how many rounds can this procedure run? Let's consider the realistic scenario where one wishes to be optimistic and picks as an initial guess for p a value $p_0 = n^{-\alpha}$ where α is a positive constant, e.g., $\alpha = 1/2$. Let $p*$ be the minimum value over all possible p with the property that for $p*$ we obtain a concentrated estimate of the true number of triangles. Clearly, $p* \leq 1$ and hence the number of rounds performed by our procedure is less that r where $p_0 c^r = 1$. Hence, for any constant $c > 1$ we obtain that the number of rounds performed by our algorithm is $O(\log n)$. Furtermore, note that the running time of the doubling procedure is dominated by the last iteration. To see why, consider for simplicity the scenario where $r + 1$ rounds are needed to deduce concentration, $c = \sqrt{2}$ and the use of the node-iterator algorithm to count triangles in the triangle sparsifier. Then, the total running time shall be $p_0^2 \sum_{v \in V(G)} deg_v^2 \left(1{+}2{+}\ldots{+}2^{r-1}{+}2^r \right)$. Finally, observe that $1{+}2{+}...{+}2^{r-1} = O(2^r)$. In practice, this procedure works even for small values of s. An instance of this procedure with $s = 2$, $\delta = 1/100$ and error threshold equal to 3% is shown in Table 4.

p	$\{T_1, T_2\}$	$\sqrt{\frac{\mathrm{Var}[T]}{\mathbb{E}[T]^2} \frac{1}{s} \ln \frac{1}{\delta}}$	err(%)
0.01	$\{42,398,007 \& 50,920,488\}$	0.1960	4.46
0.02	$\{42,540,941 \& 43,773,753\}$	0.0307	3.38
0.04	$\{44,573,294 \& 43,398,549\}$	0.0287	1.52

Table 4: Doubling procedure for the Wikipedia 2005 graph with 44,667,095 triangles.

5 Theoretical Ramifications

In Section 5.1 we investigate the performance of the Benczúr-Karger cut sparsifier and the Spielman-Srivastava spectral sparsifier with respect to triangle counting.

5.1 Cut and Spectral Sparsifiers

Consider the graph G shown in Figure 1. The strong edge connectivity of any edge in the graph is 2 and therefore the Benczúr-Karger algorithm does not distinguish the importance of the edge $e = (1, 2)$ with respect to triangle counting. The Spielman-Srivastava sparsifier with probability $1 - o(1)$ throws away the critical edge $e = (1, 2)$ as the number of vertices n tends to infinity. To prove this claim, we need use Foster's theorem, see Section 2.4.3.

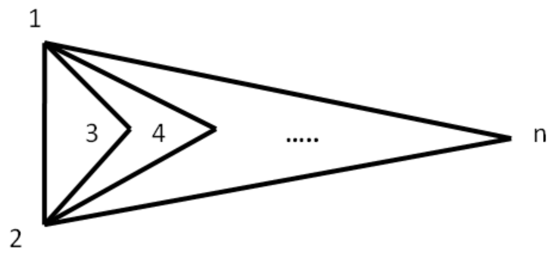

Figure 1: Graph with linear number $O(n)$ of triangles.

Claim 2 *The effective resistance $R(1,2)$ of the edge $(1,2)$ tends to 0 as n grows to infinity, i.e., $R(1,2) = o(1)$. Furthermore, all other edges have constant effective resistance.*

Proof:

Using the in-series and in-parallel network simplification rules [12], the effective conductance of the edge $(1,2)$ is $1 + \sum_{i=1}^{n-2} \frac{1}{2} = \frac{n}{2}$. Hence, the effective resistance of the edge $e = (1,2)$ is $2/n$, which also proves the first part of our claim. By Foster's theorem, the sum of the effective resistances of the edges of G is $n-1$. Due to symmetry, we obtain that $R_{(1,3)} = R_{(2,3)} = R_{(1,4)} = R_{(2,4)} = \ldots = R_{(1,n)} = R_{(2,n)} = R_n$. Therefore we obtain $\frac{2}{n} + 2(n-2)R_n = n-1 \rightarrow R_n = \frac{n^2-n-2}{2n^2-4n}$. Asymptotically as $n \rightarrow +\infty$, $R_n \rightarrow \frac{1}{2}$. □

Clearly, the Spielman-Srivastava sparsifier fails to capture the importance of the edge $(1,2)$ with respect to triangle counting. Finding an easy-to-compute quantity which allows a sparsification that preserves triangles more efficiently is an interesting problem. It is worth outlining that our analysis does not exclude effective resistances which can be computed very efficiently [32], but the use of them as is typically done in the context of spectral sparsifiers.

6 Conclusions

In this paper, we introduce the notion of a *triangle sparsifier*, i.e., a sparse graph which preserve the total number of triangles with respect to the input graph G. This notion naturally introduces a new randomized algorithm for counting triangles in a graph G, a problem with various real-world applications as outlined in Section 1.

Using a theorem on the concentration of multivariate polynomials due to Kim and Vu [29] we proved that under mild conditions on G and p our estimate T is concentrated around t with high probability, thus making our algorithm a reliable choice for counting triangles in large social networks with millions and billions of edges. By choosing a straightforward triangle counting algorithm as

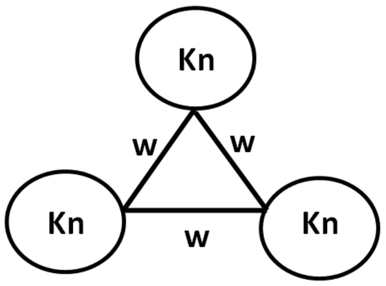

Figure 2: Weighted Graph

the node or the edge-iterator we obtain quadratic speedups $\frac{1}{p^2}$. We can justify significant speedups: for example if we know $t \geq n^{3/2+\epsilon}$ and assume $\Delta \sim n$, we get $p = n^{-1/2}$ resulting in a $O(n)$ speedup. As it was shown in [31], Triangle Sparsifiers can also be used in combination with approximate triangle counting methods to yield better practical and theoretical performance.

We believe that a tighter theoretical analysis is possible. Specifically we conjecture that if $t \geq n \log(n)$ we can obtain concentration using our method. Also, our sampling scheme can be adapted to weighted graphs: multiply the weight of sampled edge by $\frac{1}{p}$ and count weighted triangles. However this can be problematic as the graph shown in Figure 2 indicates. Specifically, for a sufficiently large w, throwing out any weighted edge results in an arbitrarily bad estimate of the count of triangles. Finding a better sampling scheme for weighted graphs is left as a problem for future work. Finally, finding an easy-to-compute quantity which gives us an optimal way to sparsify the graph with respect to triangles is an interesting research problem.

References

[1] Alon, N., Yossi, M., Szegedy, M.: *The space complexity of approximating the frequency moments.* ACM Symposium on Theory of Computing (1996)

[2] Alon, N., Yuster, R., Zwick, U.: *Finding and Counting Given Length Cycles.* Algorithmica, Volume 17, Number 3, 209–223 (1997)

[3] Alon, N., Kaufman, T., Krilevich, M., Ron, D.: *Testing triangle-freeness in general graphs* Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm (SODA), pp. 279-288 (2006)

[4] Avron, H.: *Counting triangles in large graphs using randomized matrix trace estimation.* Workshop on Large-scale Data Mining: Theory and Applications (2010)

[5] Bar-Yehuda, R. and Even, S.: *On approximating a vertex cover for planar graphs.* Proceedings of the fourteenth annual ACM symposium on Theory of computing, pp. 303-309 (1982)

[6] Bar-Yosseff, Z., Kumar, R., Sivakumar, D.: *Reductions in streaming algorithms, with an application to counting triangles in graphs.* ACM-SIAM Symposium on Discrete Algorithms (2010)

[7] Becchetti, L., Boldi, P., Castillo, C., Gionis, A.: *Efficient Semi-Streaming Algorithms for Local Triangle Counting in Massive Graphs.* ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2008)

[8] Benczúr, A., Karger, D.: *Approximating -t Minimum Cuts in Otilde;(^{2}) Time* ACM Symposium on Theory of Computing, pp. 47-55 (1998)

[9] Benczúr, A., Karger, D.: *Randomized Approximation Schemes for Cuts and Flows in Capacitated Graphs.* Arxiv Preprint, available at `http://arxiv.org/abs/cs.DS/0207078` (2002)

[10] Berry, J.W., Hendrickson, B., LaViolette, R., Phillips, C.A.: *Tolerating the Community Detection Resolution Limit with Edge Weighting.* Arxiv Preprint, available at `http://arxiv.org/abs/0903.1072`

[11] Bhamidi, S., Bresler, G., Sly, A.: *Mixing Time of Exponential Random Graphs.*, Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 803-812, (2008)

[12] Bollobás, B.: *Modern Graph Theory* Springer Verlag (1998)

[13] Bonato, A., Hadi, N, Horn, P., Pralat, P., Wang, C.: *Models of Online Social Networks.* Internet Mathematics, Vol. 6(3), 285-313, (2009)

[14] Broder, A.Z., Charikar, M., Frieze, A., Mitzenmacher, M.: *Min-wise independent permutations.* ACM Symposium on Theory of Computing (1998)

[15] Buriol, L., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., Sohler, C.: *Counting Triangles in Data Streams.* Symposium on Principles of Database Systems (2006)

[16] Chiba, N., Nishizeki, T.: *Arboricity and Subgraph Listing Algorithms.* SIAM Journal on Computing, Vol. 14(1), pp. 210-223 (1985)

[17] Chung Graham, F., Lu, L., Vu, V.: *The Spectra of Random Graphs with Given Expected Degrees.* Internet Mathematics, Vol.1(3) (2003)

[18] Coppersmith D., Winograd S.: *Matrix multiplication via arithmetic progressions.* ACM Symposium on Theory of Computing (1987)

[19] Drineas, P., Frieze, A., Kannan, R., Vempala, S., Vinay, V.: *Clustering in Large Graphs and Matrices* ACM Symposium on Discrete Algorithms (1999)

[20] Eckmann, J.-P., Moses, E.: *Curvature of co-links uncovers hidden thematic layers in the World Wide Web.* Proceedings of the National Academy of Sciences (2002)

[21] Frank, O., Strauss, D.: *Markov Graphs.* Journal of the American Statistical Association, Vol. 81(395), pp. 832-842 (1986)

[22] Foster, R.M.: *The Average Impedance of an Electrical Network.* Contributions to Applied Mechanics (Reissner Anniversary Volume), pp. 333-340 (1949)

[23] Fudos, I., Hoffman, C.: *A Graph-Constructive Approach to Solving Systems of Geometric Constraints* ACM Trans. Graph., Vol. 16, pp. 179-216 (1997)

[24] Heider, F.: *Attitudes and Cognitive Organization* Journal of Psychology, Vol. 21, 107-112 (1946)

[25] Imrich, W., Klavzar, S., Mulder, H.M.: *Median Graphs and Triangle-Free Graphs* SIAM Journal on Discrete Mathematics,**12**(1) pp. 111-118 (1999)

[26] Itai, A., Rodeh, M.: *Finding a minimum circuit in a graph.* ACM Symposium on Theory of Computing (1977)

[27] Jerrum, M., Valiant, L., Vazirani, V.: *Random generation of cominatorial structures from a uniform distribution* Theoretical Computer Science, Vol. 43, pp.169-188 (1986)

[28] Jowhari, H., Ghodsi, M.: *New Streaming Algorithms for Counting Triangles in Graphs.* Computing and Combinatorics (2005)

[29] Kim, J.H., Vu, V.H: *Concentration of multivariate polynomials and its applications.* Combinatorica, Vol. 20(3), pp. 417-434 (2000)

[30] Kim, J. H., Vu, V. H.: *Divide and conquer martingales and the number of triangles in a random graph.* Journal of Random Structures and Algorithms, 24(2), pp. 166-174 (2004)

[31] Kolountzakis, M.N., Miller, G.L., Peng, R., Tsourakakis, C.E.: *Efficient Triangle Counting in Large Graphs via Degree-based Vertex Partitioning.* 7th Workshop on Algorithms and Models for the Web Graph, (2010)

[32] Koutis, I., Miller, G.L., Peng, R.: *Solving SDD linear systems in time* $\tilde{O}(m \log n \log 1/\epsilon)$. Arxiv Preprint, available at `http://arxiv.org/abs/1102.4842`

[33] Knuth, D.: *Seminumerical Algorithms* Addison-Wesley Professional; 3 edition (1997)

[34] Latapy, M.: *Main-memory triangle computations for very large (sparse (power-law)) graphs.* Theoretical Computer Science, 407, pp. 458-473 (2008)

[35] Leskovec, J., Huttenlocher, D., Kleinberg, J.: *Predicting positive and negative links in online social networks.* International conference on World wide web, pp. 641-650 (2010)

[36] Leskovec, J., Backstrom, L., Kumar, R., Tomkins, A.: *Microscopic evolution of social networks.* ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 462-470 (2008)

[37] Li, S., Armstrong, C.M., Bertin, N., Ge. H., Miltein,. S., Boxem, M., Vidalain, P.O, Han, J.D., Chesneau, A., Hao, T., Goldberg, D., Li, N., Martinez, M., Rual, J.F., Lamesch, P., Xu, Lai, Tewari, M., Wong, S., Zhang, L., Berriz, G., Jacotot, L., Vaglio, P., Reboul, J., Hirozane-Kishikawa, T., Li, Q., Gabel, H., Elewa, A., Baumgartner, B., Rose, D., Yu, H., Bosak, S., Sequerra, R., Fraser, A., Mango, S., Saxton, W., Strome, S., Van Den Heuvel, S., Piano, F., Vandenhaute, J., Sardet, C., Gerstein, M., Doucette-Stamm, L., Gunsalus, K., Harper, J., Cusick, M., Roth, F. Hill, D., Vidal, M.: *A map of the interactome network of the metazoan C. elegans.* Science, Vol. 303, pp/ 540-543 (2004)

[38] Mislove, A., Massimiliano, M., Gummadi, K., Druschel, P., Bhattacharjee, B.: *Measurement and Analysis of Online Social Networks.* Internet Measurement Conference (2007)

[39] Newman, M. E. J. and Watts, D. J. and Strogatz, S.: *Random graph models of social networks.* Proceedings of the National Academy of Sciences of the United States of America, Vol. 99, pp. 2566-2572 (2002)

[40] Pagh, C., Tsourakakis, C.E.: *Colorful Triangle Counting and a MapReduce Implementation.* Arxiv Preprint, available at `http://arxiv.org/abs/1103.6073`

[41] Papadimitriou, C., Yannakakis, M.: *The clique problem for planar graphs.* Information Processing Letters, 13, 131–133 (1981).

[42] Robins, G., Pattison, P., Kalish, Y., Lusher, D.: *An introduction to exponential random graph (p*) models for social networks.* Social Networks Journal, Special Section: Advances in Exponential Random Graph (p*) Models, Vol. 29, pp. 173-191 (2007)

[43] Rougemont, J. and Hingamp, P: *DNA microarray data and contextual analysis of correlation graphs.* BMC Bioinformatics, Vol. 4 (2003)

[44] Schank, T., Wagner, D.: *Finding, Counting and Listing all Triangles in Large Graphs, An Experimental Study.* Workshop on Experimental and Efficient Algorithms (2005)

[45] Schank, T., Wagner, D.: *Approximating Clustering Coefficient and Transitivity.* Journal of Graph Algorithms and Applications, Vol. 9, pp. 265-275 (2005)

[46] Spielman, D., Srivastava, N.: *Graph Sparsification by Effective Resistances.* Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC), pp. 563-568 (2008)

[47] Spielman, D., Teng, S.H.: *Spectral Sparsification of Graphs.* Arxiv Preprint, available at `http://arxiv.org/abs/0808.4134`

[48] Spinrad, J.P.: *Efficient graph representations* Fields Institute Monographs, 19, 2003

[49] Tao, T., Vu, V.H.: *Additive Combinatorics.* Cambridge Studies in Advanced Mathematics (2006)

[50] Tsourakakis, C.E.: *Fast Counting of Triangles in Large Real Networks, without counting: Algorithms and Laws.* International Conference on Data Mining (2008)

[51] Tsourakakis, C.E.: *Counting Triangles Using Projections.* Knowledge and Information Systems (2011)

[52] Tsourakakis, C.E., Kang, U, Miller, G.L., Faloutsos, C.: *Doulion: Counting Triangles in Massive Graphs with a Coin.* ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2009)

[53] Tsourakakis, C.E., Drineas, P., Michelakis, E., Koutis, I., Faloutsos, C.: *Spectral Counting of Triangles via Element-Wise Sparsification and Triangle-Based Link Recommendation.* Advances in Social Networks Analysis and Mining (2010)

[54] V.H. Vu, *On the concentration of multivariate polynomials with small expectation.* Random Structures and Algorithms, Vol. 16, pp. 344-363 (2000)

[55] Wasserman, S., Faust, K.: *Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences).* Cambridge University Press (1994)

[56] Wasserman, S., Pattison, P.: *Logit models and logistic regressions for social networks: I. An introduction to Markov graphs and p*.* Psychometrika, Vol. 61, pp. 401-425 (1996)

[57] Watts, D., Strogatz, S.: *Collective dynamics of small-world networks.* Nature, Vol. 393, pp. 440-442 (1998)

[58] Wesler, H., Gleave, E., Fisher, D., Smith, M.: *Visualizing the Signatures of Social Roles in Online Discussion Groups.* The Journal of Social Structure, Vol. 8, (2007)

[59] Wimmer, A., Lewis, K.: *Beyond and Below Racial Homophily: ERG Models of a Friendship Network Documented on Facebook.* American Journal of Sociology, Vol. 2, pp. 583–642 (2010)

[60] Vassilevska Williams, V., Williams, R.: *Subcubic Equivalences between Path, Matrix and Triangle Problems* Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, pp. 645-654 (2010)

[61] Wimmer, A., Lewis, K.: *Beyond and Below Racial Homophily: ERG Models of a Friendship Network Documented on Facebook.* American Journal of Sociology, Vol. 2, pp. 583–642 (2010)

[62] Yook, S., Oltvai, Z., Barabasi, A.L.: *Functional and topological characterization of protein interaction networks.* Proteomics, Vol. 4, pp. 928-942 (2004)