Internet Mathematics Vol. 8, Nos. 1–2: 161–185

Efficient Triangle Counting in Large Graphs via Degree-Based Vertex Partitioning

Mihail N. Kolountzakis, Gary L. Miller, Richard Peng, and Charalampos E. Tsourakakis

Abstract. The number of triangles is a computationally expensive graph statistic frequently used in complex network analysis (e.g., transitivity ratio), in various random graph models (e.g., exponential random graph model), and in important real-world applications such as spam detection, uncovering the hidden thematic structures in the Web, and link recommendation. Counting triangles in graphs with millions and billions of edges requires algorithms that run fast, use little space, provide accurate estimates of the number of triangles, and preferably are parallelizable. In this paper we present an efficient triangle-counting approximation algorithm that can be adapted to the semistreaming model [Feigenbaum et al. 05]. Its key idea is to combine the sampling algorithm of [Tsourakakis et al. 09, Tsourakakis et al. 11] and the partitioning of the set of vertices into high- and low-degree subsets as in [Alon et al. 97], treating each set appropriately. From a mathematical perspective, we present a simplified proof of [Tsourakakis et al. 11] that uses the powerful Kim–Vu concentration inequality [Kim and Vu 00] based on the Hajnal–Szemerédi theorem [Hajnal and Szemerédi 70]. Furthermore, we improve bounds of existing triple-sampling techniques based on a theorem of [Ahlswede and Katona 78]. We obtain a running time $O(m + \frac{m^{3/2}}{t\epsilon^2}\log n)$ and an $(1 \pm \epsilon)$ approximation, where n is the number of vertices, m is the number of edges, and Δ is the maximum number of triangles in which any single edge is contained. Furthermore, we show how this algorithm can be adapted to the semistreaming model

ISSN: 1542-7951 print

with space usage $O(m^{1/2} \log n + \frac{m^{3/2} \log n}{t\epsilon^2})$ and a constant number of passes (three) over the graph stream. We apply our methods to various networks with several millions of edges and we obtain excellent results, outperforming existing triangle-counting methods. Finally, we propose a random-projection-based method for triangle counting and provide a sufficient condition to obtain an estimate with low variance.

1. Introduction

Graphs are ubiquitous: the Internet, the World Wide Web (WWW), social networks, protein interaction networks, and many other complicated structures are modeled as graphs [Chung Graham and Lu 06]. The problem of counting subgraphs is one of the typical graph-mining tasks that has attracted considerable attention. The most basic nontrivial subgraph is the triangle. Many social networks are abundant in triangles, since typically friends of friends tend to become friends themselves [Wasserman and Faust 94]. This phenomenon is observed in other types of networks as well (biological, online networks, etc.) and is one of the main reasons for the establishment of the definitions of the transitivity ratio and the clustering coefficients of a graph in complex network analysis [Newman 03]. In Section 2.2 we provide an extensive list of applications in which triangles are involved. Given the importance of triangle counting and the fact that several real-world networks have reached the planetary scale (e.g., Facebook, LinkedIn), fast and practical algorithms with strong theoretical guarantees are desired.

In this paper, we propose a new triangle-counting method that provides a $(1 \pm \epsilon)$ approximation to the number of triangles in the graph and runs in $O(m + \frac{m^{3/2} \log n}{t\epsilon^2})$ time, where n is the number of vertices, m is the number of edges, and Δ is the maximum number of triangles in which any single edge is contained. The key idea of the method is to combine the sampling scheme introduced in [Tsourakakis et al. 09, Tsourakakis et al. 11] with the partitioning idea of [Alon et al. 97] to obtain a more efficient sampling scheme. Furthermore, we show that this method can be adapted to the semistreaming model with a constant number of passes and $O(m^{1/2} \log n + \frac{m^{3/2} \log n}{t\epsilon^2})$ space. We apply our methods to various networks with several millions of edges, and we obtain excellent results with respect to both the accuracy and the running time. Furthermore, we optimize the cache properties of the code in order to obtain a significant additional speedup. Finally, we propose a random-projection-based method for triangle counting and provide a sufficient condition to obtain an estimate with low variance. Even if such a method is unlikely to be practical, it raises some interesting theoretical issues.

The paper is organized as follows: Section 2 presents briefly the existing work and the theoretical background; Section 3 presents our proposed method; and Section 4 presents the experimental results on several large graphs. In Section 5 we provide a sufficient condition for obtaining a concentrated estimate of the number of triangles using random projections. Finally, in Section 6 we conclude and provide possible research directions.

2. Preliminaries

In this section, we first introduce in Section 2.1 the notation used in the paper, and then in Section 2.2, we present an extensive list of applications involving triangles. In Section 2.3 we present the existing work on the triangle-counting problem. Finally, in Section 2.4 we briefly present theorems and lemmas necessary for our methods.

2.1. Notation

For the rest of the paper we use the following notation: $G([n], E)$ stands for an undirected simple graph with n vertices labeled $1, 2, \ldots, n$ and edge set E. Let m, t be the numbers of edges and triangles in G respectively; $\deg(u)$ stands for the degree of vertex u. For an edge $e \in E(G)$, we define $\Delta(e)$ to be the number of triangles containing edge e, and Δ to be the maximum number of triangles in which an edge is contained, i.e., $\Delta = \max_{e \in E(G)} \Delta(e)$. Finally, let $p \in (0, 1)$ be 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) [Wasserman and Faust 94, Wimmer and Lewis 10], and according to the latter, friends of friends tend to become friends themselves [Wasserman and Faust 94]. These facts have several implications, which we present in the following together with other applications of triangle counting. For example, recently, [Bonato et al. 09] proposed the iterated local transitivity model, which has several properties matching empirical properties of "real-world" networks such as skewed degree distribution and communities. In the following, we provide an extensive list of applications involving triangles ranging from social networks, which are of main interest to our work, to computer-aided design applications.

2.2.1. Clustering Coefficients and Transitivity of a Graph. In the influential paper [Watts and Strogatz 98], the authors proposed a simple model that explains several contradictory properties in social networks such as the abundance of 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 that of a graph; see equation (2.1). The definition of the transitivity $T(G)$ of a graph G, introduced in [Newman et al. 02], is closely related to the clustering coefficient and quantifies the probability that two neighbors of any vertex are connected. It is worth pointing out that the authors of [Newman et al. 02] erroneously claim that $C(G)$ is the same as $T(G)$; see also [Schank and Wagner 05b]. The exact definitions of the aforementioned quantities follow.

Definition 2.1. (Clustering coefficient.) A vertex $v \in V(G)$ with degree $\deg(v)$ has clustering coefficient $C(v)$ equal to the quotient of edges among its neighbors and the maximum number of triangles in which it could participate:

$$C(v) = \frac{\Delta(v)}{\binom{\deg(v)}{2}}. \tag{2.1}$$

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

Definition 2.2. (Transitivity.) The transitivity ratio $T(G)$ of a graph G is defined as

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

2.2.2. Uncovering Hidden Thematic Structures. In [Eckmann and Moses 02], the authors 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 curvature indicate a common topic, allowing the authors to extract useful metainformation. This idea has found a number of applications, such as in bioinformatics [Rougemont and Hingamp 03].

2.2.3. Exponential Random-Graph Model. In [Frank and Strauss 86], the authors 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. The authors of [Wasserman and Pattison 96] proposed the exponential random graph (ERG) model, which generalized the Markov graphs [Robins et al. 07]. Triangles are frequently used as one of the sufficient statistics of the ERG model, and counting them is necessary for parameter estimation, for example, using MCMC procedures [Bhamidi et al. 08].

2.2.4. Spam Detection. In [Becchetti et al. 08], the authors show that the distribution of triangles among spam hosts and nonspam hosts can be used as a feature for classifying a given host as spam or nonspam. The same result holds also for web pages, i.e., the spam and nonspam triangle distributions differ from each other at a detectable level using standard statistical tests.

2.2.5. Content Quality and Role Behavior Identification. Today, there exist many online forums in which acknowledged scientists participate (e.g., MathOverflow, CStheory stack exchange) to discuss problems of their fields. This yields significant information for researchers. Several interesting questions arise, such as which participants comment on each other's contributions. This question among several others was studied in [Wesler et al. 07]. The number of triangles in which 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 [Becchetti et al. 08].

2.2.6. Structural Balance and Status Theory. The notion of *balance theory* appeared first in the seminal work [Heider 46]. It is based on the concepts "the friend of my friend is my friend," "the enemy of my friend is my enemy," and the like [Wasserman and Faust 94]. 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. The concept of *status theory* is based on interpreting a positive edge (u, v) as u having lower status than v, while the negative edge (u, v) means that u regards v as having a lower status than itself. Recently, the authors of [Leskovec et al. 10] performed experiments to study which of the two aforementioned theories applies better to online social networks and to predict the signs of incoming links. Their algorithms require counts of signed triangles in the graph.

2.2.7. Microscopic Evolution of Networks. In [Leskovec et al. 08], the authors 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.

2.2.8. Community Detection. Counting triangles is also used in community-detection algorithms. Specifically, in [Berry et al. 11], the authors use triangle counting to deduce the edge support measure in their community-detection algorithm.

2.2.9. Motif Detection. Triangles are abundant not only in social networks but also in biological networks [Li et al. 04, Yook et al. 04]. This fact can be used to correlate the topological and functional properties of protein interaction networks [Yook et al. 04].

2.2.10. CAD Applications. In [Fudos and Hoffman 97], the authors introduced a graph-constructive approach to solving systems of geometric constraints, a problem that 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, the exact and the approximate. It is worth noting that for most of the applications described in Section 2.2, the exact number of triangles is not crucial. Hence approximate counting algorithms that are faster 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 by Alon, Yuster, Zwick (AYZ) [Alon et al. 97]; it runs in time $O(m^{\frac{2\omega}{\omega+1}})$, where currently the fast matrix multiplication exponent ω is 2.371 [Coppersmith and Winograd 87]. Thus, the 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 restrictive memory requirements. Even for medium-sized networks, matrix-multiplication-based algorithms are not applicable.

The AYZ algorithm introduces the concept of partitioning the vertices into high- and low-degree vertices using as threshold the value $m^{\frac{\omega-1}{\omega+1}}$ and treating them appropriately by matrix multiplication and exact combinatorial counting respectively. We use this partitioning idea with an appropriately defined threshold for our purpose in Section 3 to obtain state-of-the-art results on approximate triangle counting.

In 1978, Itai and Rodeh presented an algorithm that finds a triangle in any graph in $O(m^{3/2})$ time [Itai and Rodeh 77]. 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 $O(m^{3/2})$ time in the worst case [Chiba and Nishizeki 85]. For special types of graphs, more efficient triangle-counting algorithms exist. For instance, in planar graphs, triangles can be found in $O(n)$ time [Chiba and Nishizeki 85, Itai and Rodeh 77, Papadimitriou and Yannakakis 81].

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 the number of triangles in which it is involved, 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 [Latapy 08, Schank and Wagner 05a].

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 [Bar-Yosseff et al. 10]. Two main representations of a graph as a stream are the edge stream and the incidence/adjacency 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 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 also be applied in the setting in which the graph fits in memory. Monte Carlo sampling techniques have been proposed to give a fast estimate of the number of triangles. According to such an approach, also known as naive sampling [Schank and Wagner 05b], we choose three nodes at random repeatedly and check whether they form a triangle. If one makes

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

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', which is equal to the

fraction of selected triples that form triangles multiplied by $\binom{n}{3}$, 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 [Bar-Yosseff et al. 10], 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 (AMS) algorithms [Alon et al. 96] to proceed. The key is that the triangle computation reduces to estimating the zeroth, first, and second frequency moments, which can be done efficiently. It is worth noting that of independent interest is their space lower bound of $\Omega(n^2)$ for approximating the number of triangles in a graph given in the adjacency stream representation. 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 naive sampling.

The AMS algorithms are also used by [Jowhari and Ghodsi 05], 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. Along the same lines, the authors of [Buriol et al. 06] proposed two space-bounded sampling algorithms to estimate the number of triangles. Again, the underlying sampling procedures are simple. For instance, for the case of the edge stream representation, they sample randomly an edge and a node in the stream and check whether 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 draws that have to be made in order to get concentration (these draws are done in parallel) is of order

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

Even if the term T_0 is missing compared to 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.

In [Tsourakakis et al. 09], the DOULION algorithm tosses a coin independently for each edge with probability p to keep the edge and probability $q = 1-p$ to discard it. Then it counts the number of triangles t' in the sparsified graph and outputs as an estimate for the true number of triangles the value t'/p^3. It was shown later in [Tsourakakis et al. 11] using a powerful theorem from [Kim and Vu 00] that under mild conditions on the triangle density, the method results in a strongly concentrated estimate around the number of triangles t. Recently,

Pagh and Tsourakakis proposed a more efficient sampling scheme that colors the vertices of G using N colors uniformly at random and counts monochromatic triangles; see [Pagh and Tsourakakis 12].

Another line of work is based on linear-algebraic arguments. Specifically, in the case of "power-law" networks, it was shown in [Tsourakakis 08] that the spectral counting of triangles can be efficient due to their special spectral properties. Then [Chung Graham et al. 03] and [Tsourakakis 11] extended this idea using the randomized algorithm of [Drineas et al. 99] by proposing a simple biased node sampling.

In [Becchetti et al. 08], the semistreaming 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 [Broder et al. 98] are applicable to the problem. More recently, Avron proposed a new approximate triangle-counting method based on a randomized algorithm for trace estimation [Avron 10].

2.4. Theoretical Preliminaries

2.4.1. Concentration of Measure. In Section 3 we make extensive use of the following version of the Chernoff bound [Chernoff 81].

Theorem 2.3. *Let $X_1, X_2, \ldots, X_k$ be independently distributed $\{0,1\}$ variables with $E[X_i] = p$. Then for any $\epsilon > 0$, we have*

$$\Pr\left[\left|\frac{1}{k}\sum_{i=1}^{k} X_i - p\right| > \epsilon p\right] \leq 2e^{-\epsilon^2 pk/2}.$$

2.4.2. Random Projections. A random projection $x \to Rx$ from $\mathbb{R}^d$ to $\mathbb{R}^k$ approximately preserves all Euclidean distances. One version of the Johnson–Lindenstrauss lemma [Johnson and Lindenstrauss 84] is the following:

Lemma 2.4. (Johnson and Lindenstrauss.) *Suppose $x_1, \ldots, x_n \in \mathbb{R}^d$ and $\epsilon > 0$ and take $k = C\epsilon^{-2}\log n$. Define the random matrix $R \in \mathbb{R}^{k\times d}$ by taking all $R_{i,j} \sim N(0,1)$ (standard Gaussian) and independent. Then with probability bounded below by a constant, the points $y_j = Rx_j \in \mathbb{R}^k$ satisfy*

$$(1-\epsilon)|x_i - x_j| \leq |y_i - y_j| \leq (1+\epsilon)|x_i - x_j|$$

for $i, j = 1, 2, \ldots, n$, where $|\cdot|$ represents the Euclidean norm.

2.4.3. Extremal Graph Theory. Hajnal and Szemerédi proved in 1970 the following conjecture of Paul Erdős [Hajnal and Szemerédi 70]:

Theorem 2.5. Hajnal–Szemerédi theorem *Every graph with n vertices and maximum vertex degree at most k is $(k+1)$-colorable with all color classes of size $\lfloor \frac{n}{k+1} \rfloor$ or $\lceil \frac{n}{k+1} \rceil$.*

Ahlswede and Katona consider the following problem: which graph with a given number of vertices n and a given number of edges m maximizes the number of edges in its line graph $L(G)$? The problem is equivalent to maximizing the sum of squares of the degrees of the vertices under the constraint that their sum equal twice the number of edges. The following theorem was given in [Ahlswede and Katona 78] and answers this question.

Lemma 2.6. (Ahlswede–Katona theorem.) *The maximum value of the sum of the squares of all vertex degrees $\sum_{v \in V(G)} \deg(v)^2$ over the set of all graphs with n vertices and m edges occurs at one or both of two special types of graphs: the quasistar graph and the quasicomplete graph.*

For further progress on other questions related to the above optimization problem such as when the optimum occurs at both graphs, see [Abrego et al. 09].

3. Proposed Method

Our algorithm combines two approaches that have been taken on triangle counting: sparsify the graph by keeping a random subset of the edges [Tsourakakis et al. 09, Tsourakakis et al. 11] followed by a triple sampling using the idea of vertex partitioning due to [Alon et al. 97]. In the following, we shall assume that the input is in the form of an edge file, i.e., a file each line of which contains an edge. Notice that given this representation, computing the degrees takes linear time.

3.1. Edge Sparsification

The following method was introduced in [Tsourakakis et al. 09] and has been shown to perform very well in practice: Keep each edge with probability p independently. Then for each triangle, the probability of it being kept is p^3. So the expected number of triangles left is $p^3 t$. This is an inexpensive way to reduce the size of the graph, since it can be done in one pass over the edge list using $O(mp)$ random variables (more details can be found in Section 4.2.1 and in [Knuth 97]).

In a later analysis [Tsourakakis et al. 11], it was shown that from the number of triangles in the sampled graph we can obtain a concentrated estimate around the actual triangle count as long as $p^3 \geq \tilde{\Omega}(\frac{\Delta}{t})$.[1] Here, we prove a similar bound using more elementary techniques. Suppose we have a set of k triangles such that no two share an edge. For each such triangle we define a random variable X_i that is 1 if the triangle is kept by the sampling, and 0 otherwise.

Then since the triangles do not have any edges in common, the X_i's are independent and take the value 0 with probability $1 - p^3$, and the value 1 with probability p^3. So by the Chernoff bound,

$$\Pr\left[\left|\frac{1}{k}\sum_{i=1}^{k} X_i - p^3\right| > \epsilon p^3\right] \leq 2e^{-\epsilon^2 p^3 k/2}.$$

So when $p^3 k\epsilon^2 \geq 4d \log n$, where d is a positive constant, the probability of sparsification returning an ϵ approximation is at least $1 - n^{-d}$. This is equivalent to $p^3 k \geq 4d \log n/\epsilon^2$, which suggests that in order to sample with small p and hence discard many edges, we need k to be large.

To show that such a large set of independent triangles exists, we invoke the Hajnal–Szemerédi theorem, Theorem 2.5, on an auxiliary graph H, which we construct as follows. For each triangle i $(i = 1, \ldots, t)$ in G we create a vertex v_i in H. We connect two vertices v_i, v_j in H if and only if they represent triangles i, j respectively that share an edge in G. Notice that the maximum degree in the auxiliary graph H is $O(\Delta)$. Hence, we obtain the following corollary.

Corollary 3.1. *Given t triangles such that no edge belongs to more than Δ triangles, we can partition the triangles into sets $S_1, \ldots, S_l$ such that $|S_i| > \Omega(t/\Delta)$ and l is bounded by $O(\Delta)$.*

Combining Corollary 3.1 and the Chernoff bound allows us to prove the next theorem.

Theorem 3.2. *If $p^3 \in \Omega(\frac{\Delta \log n}{\epsilon^2 t})$, then with probability $1 - n^{-2}$, the sampled graph has a triangle count that ϵ-approximates t.*

Proof. Consider the partition of triangles given by Corollary 3.1 and let $d = 5$. By choice of p we get that the probability that the triangle count in each set is preserved within a factor of $\epsilon/2$ is at least $1 - n^{-d}$. Since there are at most n^3 such sets, an application of the union bounds gives that their total is approximated

[1] We use the tilde notation to hide polylogarithmic factors polylog(n).

within a factor of $\epsilon/2$ with probability at least $1 - n^{3-d}$. This gives that the triangle count is approximated within a factor of ϵ with probability at least $1 - n^{3-d}$. Substituting $d = 5$ completes the proof. □

3.2. Triple Sampling

Since each triangle corresponds to a triple of vertices, we can construct a set of triples U that includes all triangles. From this list, we can then sample triples uniformly at random. Let these samples be numbered from 1 to s. Also, for the ith triple sampled, let X_i be 1 if it is a triangle and 0 otherwise. Since we pick triples randomly from U, and t of them are triangles, we have $E(X_i) = t/|U|$, and the X_i are independent. So by the Chernoff bound we obtain

$$\Pr\left[\left|\frac{1}{s}\sum_{i=1}^{s} X_i - \frac{t}{|U|}\right| > \epsilon\frac{t}{|U|}\right] \leq 2e^{-\epsilon^2 ts/(2|U|)}.$$

If $s = \Omega(\frac{|U|\log n}{t\epsilon^2})$, then we have that $|U|\sum_{i=1}^{s}\frac{X_i}{s}$ approximates t within a factor of ϵ with probability at least $1 - n^{-d}$ for any d of our choice. Since $|U| \leq n^3$, this immediately gives an algorithm with runtime $O(n^3 \log n/(t\epsilon^2))$ that approximates t within a factor of ϵ. Slightly more careful bookkeeping can also give tighter bounds on $|U|$ in sparse graphs.

A simple but crucial observation that allows us to decide whether we will sample a triple of vertices or an edge and a vertex is the following. Consider any triple (u, v, w) containing vertex u. Since $uv, uw \in E$, we have that the number of such triples involving u is at most $\deg(u)^2$. From an edge–vertex sampling point of view, since $vw \in E$, another bound on the number of such triples is m. When $\deg(u) > m^{1/2}$, the second bound is tighter, and the first is tighter in the other case.

These two cases naturally suggest that low-degree vertices with degree at most $m^{1/2}$ be treated separately from high-degree vertices with degree greater than $m^{1/2}$. For the number of triangles around low-degree vertices, the value of $\sum_u \deg(u)^2$ is maximized when all edges are concentrated in as few vertices as possible [Ahlswede and Katona 78]. Since the maximum degree of such a vertex is $m^{1/2}$, the number of such triangles is bounded above by $m^{1/2} \cdot (m^{1/2})^2 = m^{3/2}$. Also, since the sum of all degrees is $2m$, there can be at most $2m^{1/2}$ high-degree vertices, which means that the total number of triangles incident to these high-degree vertices is at most $2m^{1/2} \cdot m = 2m^{3/2}$. Combining these bounds gives that $|U|$ can be bounded above by $3m^{3/2}$. Note that this bound is asymptotically tight when G is a complete graph ($n = m^{1/2}$). However, in practice, the second bound can be further reduced by summing over the degrees of all v adjacent to u, becoming $\sum_{uv\in E} \deg(v)$. As a result, an algorithm that implicitly constructs U by

picking the better one of these two cases by examining the degrees of all neighbors will achieve $|U| \leq O(m^{3/2})$. This improved bound on U gives an algorithm that ϵ-approximates the number of triangles in time

$$O\left(m + \frac{m^{3/2} \log n}{t\epsilon^2}\right).$$

As our experimental data in Section 4.1 indicate, the value of t is usually $\Omega(m)$ in practice. In such cases, the second term in the above calculation becomes negligible compared to the first one. In fact, in most of our data, sampling just the first type of triples (that is, pretending that all vertices are of low degree) brings the second term below the first.

3.3. Hybrid Algorithm

Edge sparsification with a probability of p allows us to work on only $O(mp)$ edges. Therefore, the total runtime of the triple sampling algorithm after sparsification with probability p becomes

$$O\left(mp + \frac{\log n(mp)^{3/2}}{\epsilon^2 tp^3}\right) = O\left(mp + \frac{\log n \cdot m^{3/2}}{\epsilon^2 tp^{3/2}}\right).$$

As stated above, since the first term in most practical cases is much larger, we can set the value of p to balance these two terms out:

$$pm = \frac{m^{3/2} \log n}{p^{3/2} t\epsilon^2} \Rightarrow p^{5/2} t\epsilon^2 = m^{1/2} \log n \Rightarrow p = \left(\frac{m^{1/2} \log n}{t\epsilon^2}\right)^{2/5}.$$

The actual value of p chosen would also depend heavily on the constants in front of both terms, since sampling is likely much less expensive due to factors such as cache effect and memory efficiency. Nevertheless, our experimental results in Section 4 seem to indicate that this type of hybrid algorithm can perform better in certain situations.

3.4. Sampling in the Semistreaming Model

The previous analysis of triangle counting in [Alon et al. 97] was done in the streaming model, in which the assumption was constant available space. We show that our sampling algorithm can be done in a slightly weaker model with space usage equaling

$$O\left(m^{1/2} \log n + \frac{m^{3/2} \log n}{t\epsilon^2}\right).$$

We assume that the edges adjacent to each vertex are given in order [Feigenbaum et al. 05]. We first need to identify high-degree vertices, specifically those with

degree greater than $m^{1/2}$. This can be done by sampling $O(m^{1/2} \log n)$ edges and recording the vertices that are endpoints of one of those edges.

Lemma 3.3. *Suppose $dm^{1/2} \log n$ samples were taken. Then the probability of all vertices with degree at least $m^{1/2}$ being chosen is at least $1 - n^{-d+1}$.*

Proof. Consider some vertex v with degree at least $m^{1/2}$. The probability of it being picked in each iteration is at least $m^{1/2}/m = m^{-1/2}$. As a result, the probability of it not being picked in $dm^{1/2} \log n$ iterations is

$$(1 - m^{-1/2})^{dm^{1/2} \log n} = \left[(1 - m^{1/2})^{m^{1/2}}\right]^{d \log n} \leq \left(\frac{1}{e}\right)^{d \log n} = n^{-d}.$$

Since there are at most n vertices, applying the union bound gives that all vertices with degree at least $m^{1/2}$ are sampled with probability at least $1 - n^{-d+1}$. □

Our proposed method comprises the following three steps/passes over the stream:

1. Identifying high-degree vertices requires one pass of the graph. Also, note that the number of potential candidates can be reduced to $m^{1/2}$ using another pass over the edge list.
2. For all the low-degree vertices, we can read their $O(m^{1/2})$ neighbors and sample from them. For the high-degree vertices, we do the following: for each edge, obtain a random variable y from a binomial distribution equal to the number of edge/vertex pairs in which this edge is involved. Then pick y vertices from the list of high-degree vertices randomly. These two sampling procedures can be done together in another pass over the data.
3. Finally, we need to check whether each edge in the sampled triples belongs to the edge list. We can store all such queries in a hash table, since there are at most $O(\frac{m^{3/2} \log n}{t\epsilon^2})$ edges sampled with high probability. Then going through the graph edges in a single pass and looking them up in the table yields the desired answer.

Description	Availability
Stanford Large Network Dataset collection	http://snap.stanford.edu/
UF Sparse Matrix Collection	http://www.cise.ufl.edu/research/sparse
Max Planck	http://socialnetworks.mpi-sws.org/

Table 1. Sources of data sets.

4. Experiments

4.1. Data

The graphs used in our experiments are shown in Table 2. Multiple edges and self-loops (if any) were removed. All graphs with the exceptions of Livejournal-links and Flickr are available on the Web. Table 1 summarizes the resources.

4.2. Experimental Setup and Implementation Details

The experiments were performed on a single machine, with Intel Xeon CPU at 2.83 GHz, 6144 KB cache size, and 50 GB of main memory. The graphs are from real-world web graphs; some details regarding them are in Table 2. 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 this time, since the number of memory operations performed in setting up the graph is not negligible. However, we use a modified IO routine that is much faster than the standard C/C++ `scanf`.

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

Table 2. Data sets used in our experiments. Orkut-links is from [Mislove et al. 07]; Youtube is from [Mislove et al. 07].

A major optimization that we used was to sort the edges in the graph and store the input file in the format of a sequence of neighbor lists per vertex. Each neighbor list begins with the size of the list, followed by the neighbors. This is similar to how software programs such as MATLAB store sparse matrices. The preprocessing time to change the data into this format is not included. It can significantly improve the cache property of the graph stored, and hence the overall performance.

Some implementation details are based on this graph-storage format. Specifically, since each triple that we check by definition has two edges already in the graph, it suffices to check/query whether the third edge is present in the graph. In order to do this efficiently, rather than querying the existence of an edge upon sampling each triple, we store the entire set of the queries and answer them in one pass through the graph. Finally, in the next section we discuss the details behind efficient binomial sampling. Specifically, picking a random subset of expected size $p|S|$ from a set S can be done in expected sublinear time [Knuth 97].

4.2.1. Binomial Sampling in Expected Sublinear Time. Most of our algorithms have the following routine in their core: Given a list of values, keep each of them with probability p and discard with probability $1 - p$. If the list has length n, this can clearly be done using n random variables. Since generating random variables can be expensive, it is preferable to use $O(np)$ random variables in expectation if possible. One possibility is to pick $O(np)$ random elements, but this would likely involve random accesses in the list or maintaining a list of the indices picked in sorted order. A simple way that we use in our code to perform this sampling is to generate the differences between indices of entries retained [Knuth 97]. This variable clearly belongs to an exponential distribution, and if x is a uniform random number in $(0, 1)$, we take $\lceil \log_{(1-p)} x \rceil$ as the value of the random variable; see [Knuth 97]. The primary advantage of doing so is that sampling can be done while the data are accessed in a sequential fashion, which results in much better cache performance.

4.3. Results

The six variants of the code involved in the experiment are first separated by whether the graph was first sparsified by keeping each edge with probability $p = 0.1$. In either case, an exact algorithm based on hybrid sampling with performance bounded by $O(m^{3/2})$ was run. Then two triple-based sampling algorithms are also considered. They differ in whether an attempt is made to distinguish between low- and high-degree vertices, so the simple version essentially samples all V-shaped triples off each vertex. Notice that the first column of Table 3 is all

	No Sparsification					
Graph	Exact		Simple		Hybrid	
	err(%)	time	err(%)	time	err(%)	time
AS-Skitter	0.000	4.452	1.308	0.746	0.128	1.204
Flickr	0.000	41.981	0.166	1.049	0.128	2.016
Livejournal-links	0.000	50.828	0.309	2.998	0.116	9.375
Orkut-links	0.000	202.012	0.564	6.208	0.286	21.328
Soc-LiveJournal	0.000	38.271	0.285	2.619	0.108	7.451
Web-EDU	0.000	8.502	0.157	2.631	0.047	3.300
Web-Google	0.000	1.599	0.286	0.379	0.045	0.740
Wiki-2005	0.000	32.472	0.976	1.197	0.318	3.613
Wiki-2006/9	0.000	86.623	0.886	2.250	0.361	7.483
Wiki-2006/11	0.000	96.114	1.915	2.362	0.530	7.972
Wiki-2007	0.000	122.395	0.943	2.728	0.178	9.268
Youtube	0.000	1.347	1.114	0.333	0.127	0.500

Table 3. Results of experiments averaged over five trials using only triple sampling.

zeros since when we use the exact counting algorithm on the original graph we compute the true count of triangles. Errors are measured by the absolute value of the difference between the true count of triangles and our estimate, normalized (or divided) by the true count of estimates. The results on error and running time are averaged over five runs, and the results are shown in Tables 3 and 4.

4.4. Remarks

From Table 2 it is evident that social networks are abundant in triangles. For example, the Flickr graph with only ∼1.9 M vertices has ∼550 M triangles and the Orkut graph with ∼3 M vertices has ∼620 M triangles. Furthermore, from Tables 3 and 4 it is clear that none of the variants clearly outperforms the others on all the data. The gain/loss from sparsification is likely due to the fixed sampling rate. Adapting a doubling procedure for the sampling rate as in [Tsourakakis et al. 11] is likely to mitigate this discrepancy. The difference between simple and hybrid sampling is due to the fact that handling the second case of triples has a much worse cache access pattern, since it examines vertices that are two hops away. There are alternative implementations for handling this situation, which would be interesting to explore for future implementations. A fixed sparsification rate of $p = 10\%$ was used mostly to simplify the setups of the experiments. In practice, the preferred option is to vary p to look for a rate such that the result stabilizes [Tsourakakis et al. 11].

	Sparsified ($p = 0.1$)					
Graph	Exact		Simple		Hybrid	
	err(%)	time	err(%)	time	err(%)	time
AS-Skitter	2.188	0.641	3.208	0.651	1.388	0.877
Flickr	0.530	1.389	0.746	0.860	0.818	1.033
Livejournal-links	0.242	3.900	0.628	2.518	1.011	3.475
Orkut-links	0.172	9.881	1.980	5.322	0.761	7.227
Soc-LiveJournal	0.681	3.493	0.830	2.222	0.462	2.962
Web-EDU	0.571	2.864	0.771	2.354	0.383	2.732
Web-Google	1.112	0.251	1.262	0.371	0.264	0.265
Wiki-2005	1.249	1.529	7.498	1.025	0.695	1.313
Wiki-2006/9	0.402	3.431	6.209	1.843	2.091	2.598
Wiki-2006/11	0.634	3.578	4.050	1.947	0.950	2.778
Wiki-2007	0.819	4.407	3.099	2.224	1.448	3.196
Youtube	1.358	0.210	5.511	0.302	1.836	0.268

Table 4. Results of experiments averaged over five trials using sparsification and triple sampling.

When compared with previous results on this problem, the error rates and running times of our results are all significantly lower. In fact, on the wiki graphs, our exact counting algorithms have about the same order of speed as other approximate triangle-counting implementations. This is also why we did not include any competitors in the exposition of the results, since our implementation is a highly optimized C/C++ implementation with an emphasis on performance for huge graphs.

As we discussed in Section 2, there exists considerable interest in counting triangles in signed graphs. It is clear that our method applies to this setting as well, by considering individually each possible configuration of a signed triangle. However, we do not include any of our experimental findings here due to the small size of the signed networks available from the Stanford Network Analysis library (SNAP).

5. Theoretical Ramifications

In Section 5.1 we discuss random projections and triangles, motivated by the simple observation that the inner product of two rows of the adjacency matrix corresponding to two connected vertices forming edge e gives the count of triangles $\Delta(e)$.

5.1. Random Projections and Triangles

Consider any two vertices $i, j \in V$ that are connected, i.e., $(i, j) \in E$. Observe that the inner product of the ith and jth columns of the adjacency matrix of graph G gives the number of triangles in which edge (i, j) participates. Viewing the adjacency matrix as a collection of n points in $\mathbb{R}^n$, a natural question to ask is whether we can use results from the theory of random projections [Johnson and Lindenstrauss 84] to reduce the dimensionality of the points while preserving the inner products that contribute to the count of triangles. In [Magen and Zouzias 08], the authors have considered a similar problem, namely random projections that preserve approximately the volume for all subsets of at most k points.

According to Lemma 2.4, the projection $x \to Rx$ from $\mathbb{R}^d$ to $\mathbb{R}^k$ approximately preserves all Euclidean distances. However, it does not preserve all pairwise inner products. This can easily be seen by considering the set of points $e_1, \ldots, e_n \in \mathbb{R}^n = \mathbb{R}^d$, where $e_1 = (1, 0, \ldots, 0)$, etc. Indeed, all inner products of the above set are zero, which cannot happen for the points Re_j, since they belong to a lower-dimensional space, and they cannot all be orthogonal. For the triangle-counting problem we do not need to approximate *all* inner products. Suppose $A \in \{0, 1\}^n$ is the adjacency matrix of a simple undirected graph G with vertex set $V(G) = \{1, 2, \ldots, n\}$, and write A_i for the ith column of A. The quantity in which we are interested is the number of triangles in G (actually six times the number of triangles)

$$t = \sum_{u,v,w \in V(G)} A_{uv} A_{vw} A_{wu}.$$

If we apply a random projection of the above kind to the columns of A,

$$A_i \to RA_i,$$

and write

$$X = \sum_{u,v,w \in V(G)} (RA)_{uv} (RA)_{vw} (RA)_{wu},$$

then it is easy to see that $\mathbb{E}[X] = 0$, since X is a linear combination of triple products $R_{ij} R_{kl} R_{rs}$ of entries of the random matrix R and all such products have expected value 0, no matter what the indices. So we cannot expect this kind of random projection to work.

Therefore, we consider the following approach, which still has limitations, as we will show in the following. Let

$$t = \sum_{u \sim v} A_u^\top A_v, \quad \text{where } u \sim v \text{ means } A_{uv} = 1,$$

and look at the quantity

$$Y = \sum_{u\sim v}(RA_u)^\top(RA_v) = \sum_{l=1}^{k}\sum_{i,j=1}^{n}\left(\sum_{u\sim v}A_{iu}A_{jv}\right)R_{li}R_{lj}$$
$$= \sum_{l=1}^{k}\sum_{i,j=1}^{n}\#\{i-*-*-j\}R_{li}R_{lj}.$$

This is a quadratic form in the Gaussian $N(0,1)$ variables R_{ij}. By simple calculation for the mean value and diagonalization for the variance we see that if the X_j are independent $N(0,1)$ variables and

$$Z = X^\top BX,$$

where $X = (X_1,\ldots,X_n)^\top$ and $B \in \mathbb{R}^{n\times n}$ is *symmetric*, then

$$\mathbb{E}[Z] = \operatorname{Tr} B, \quad \mathbb{V}\mathrm{ar}[Z] = \operatorname{Tr} B^2 = \sum_{i,j=1}^{n}(B_{ij})^2.$$

Hence $\mathbb{E}[Y] = \sum_{l=1}^{k}\sum_{i=1}^{n}\#\{i-*-*-i\} = k\cdot t$, so the mean value is the quantity we want (multiplied by k). For this to be useful, we should have some concentration for Y near $\mathbb{E}[Y]$. We do not need exponential tails because we have only one quantity to control. In particular, a statement of the type

$$\Pr|Y - \mathbb{E}[Y]| > \epsilon\mathbb{E}[Y] < 1 - c_\epsilon,$$

where $c_\epsilon > 0$, would be enough.

The simplest way to check this is by computing the standard deviation of Y. By Chebyshev's inequality it suffices that the standard deviation be much smaller than $\mathbb{E}[Y]$. According to the formula above for the variance of a quadratic form, we get

$$\mathbb{V}\mathrm{ar}[Y] = \sum_{l=1}^{k}\sum_{i,j=1}^{n}\#\{i-*-*-i\}^2 = C\cdot k\cdot\#\{x-*-*-*-*-*-*-x\}$$
$$= C\cdot k\cdot(\text{number of circuits of length 6 in } G).$$

Therefore, to have concentration it is sufficient that

$$\mathbb{V}\mathrm{ar}[Y] = o(k\cdot(\mathbb{E}[Y])^2). \tag{5.1}$$

Observe that (5.1) is a sufficient—and not necessary—condition. Furthermore,(5.1) is certainly not always true, since there are graphs with many 6-circuits and no triangles at all (the circuits *may* repeat vertices or edges).

6. Conclusions and Future Work

In this work, we have extended previous work [Tsourakakis et al. 09, Tsourakakis et al. 11] by introducing a powerful idea from [Alon et al. 97]. Specifically, we have proposed a Monte Carlo algorithm that approximates the true number of triangles within ϵ and runs in $O(m + \frac{m^{3/2} \log n}{t\epsilon^2})$ time. Our method can be extended to the semistreaming model using three passes and a memory overhead of $O(m^{1/2} \log n + \frac{m^{3/2} \log n}{t\epsilon^2})$.

In practice, our methods obtain excellent running times. The accuracy is also satisfactory, especially for the type of applications we are concerned with. Finally, we propose a random-projection-based method for triangle counting and provide a sufficient condition to obtain an estimate with low variance. A natural question is the following: can we provide some reasonable condition on G that would guarantee (5.1)? Furthermore, our proposed methods are easily parallelizable, and developing such an implementation in the MapReduce framework—see [Jeffrey and Ghemawat 04] and [Kang et al. 09, Kang et al. 10]—is a natural practical direction. Finally, coming up with an easy-to-compute quantity that would allow us to sparsify more efficiently is an interesting question.

Acknowledgments. The authors gratefully acknowledge support from the University of Crete, Grant No. 2569, and the National Science Foundation, Grants Nos. IIS-0705359, CCF-0635257, and CCF-1013110. This is the extended version of our proceedings paper [Kolountakis et al. 10].

References

[Abrego et al. 09] B. Abrego, S. Fernandez-Merchant, M Neubauer, and W. Watkins. "Sum of Squares of Degrees in a Graph." *Journal of Inequalities in Pure and Applied Mathematics* 10 (2009), ART 64.

[Ahlswede and Katona 78] R. Ahlswede and G. O. H. Katona. "Graphs with Maximal Number of Adjacent Pairs of Edges." *Acta Mathematica Academiae Scientarium Hungaricae* 32 (1978), 97–120.

[Alon et al. 96] N. Alon, M. Yossi, and M. Szegedy. "The Space Complexity of Approximating the Frequency Moments." In *ACM Symposium on Theory of Computing*, 1996.

[Alon et al. 97] N. Alon, R. Yuster, and U. Zwick. "Finding and Counting Given Length Cycles." *Algorithmica* 17:3 (1997), 209–223.

[Avron 10] H. Avron. "Counting Triangles in Large Graphs Using Randomized Matrix Trace Estimation." In *Workshop on Large-Scale Data Mining: Theory and Applications*, 2010.

[Bar-Yosseff et al. 10] Z. Bar-Yosseff, R. Kumar, and D. Sivakumar. "Reductions in Streaming Algorithms, with an Application to Counting Triangles in Graphs." In *ACM–SIAM Symposium on Discrete Algorithms*, 2010.

[Becchetti et al. 08] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. "Efficient Semi-streaming Algorithms for Local Triangle Counting in Massive Graphs." In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2008.

[Berry et al. 11] J. W. Berry, B. Hendrickson, R. LaViolette, and C. A. Phillips. "Tolerating the Community Detection Resolution Limit with Edge Weighting." *Physical Review E* 83:5 (2011), 056119-1–056119-9.

[Bhamidi et al. 08] S. Bhamidi, G. Bresler, and A. Sly. "Mixing Time of Exponential Random Graphs." In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pp. 803–812, 2008.

[Bonato et al. 09] A. Bonato, N. Hadi, P. Horn, P. Pralat, and C. Wang. "Models of Online Social Networks." *Internet Mathematics* 6:3 (2009), 285–313.

[Broder et al. 98] A. Z. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. "Min-wise Independent Permutations." In *ACM Symposium on Theory of Computing*, 1998.

[Buriol et al. 06] L. Buriol, G. Frahling, S. Leonardi, A, Marchetti-Spaccamela, and C. Sohler. "Counting Triangles in Data Streams." In *Symposium on Principles of Database Systems*, 2006.

[Chernoff 81] H. Chernoff. "A Note on an Inequality Involving the Normal Distribution." *Annals of Probability* 9:3 (1981), 533–535.

[Chiba and Nishizeki 85] N. Chiba and T. Nishizeki. "Arboricity and Subgraph Listing Algorithms." *SIAM Journal on Computing* 14:1 (1985), 210–223.

[Chung Graham and Lu 06] F. Chung Graham and L. Lu. "Complex Graphs and Networks." Providence: American Mathematical Society, 2006.

[Chung Graham et al. 03] F. Chung Graham, L. Lu, and V. Vu. "The Spectra of Random Graphs with Given Expected Degrees." In *Proceedings of the National Academy of Sciences of the United States of America* 100:11 (2003), 6313–6318.

[Coppersmith and Winograd 87] D. Coppersmith and S. Winograd. "Matrix multiplication via arithmetic progressions." In *ACM Symposium on Theory of Computing*, 1987.

[Drineas et al. 99] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. "Clustering in Large Graphs and Matrices." In *ACM Symposium on Discrete Algorithms*, 1999.

[Eckmann and Moses 02] J.-P. Eckmann and E. Moses. "Curvature of Co-links Uncovers Hidden Thematic Layers in the World Wide Web." *Proceedings of the National Academy of Sciences* 99:9 (2002), 5825–5829.

[Frank and Strauss 86] O. Frank and D. Strauss. "Markov Graphs." *Journal of the American Statistical Association* 81:395 (1986), 832–842.

[Feigenbaum et al. 05] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. "On Graph Problems in a Semi-streaming Model." *Journal of Theoretical Computer Science* 348:2 (2005), 207–216.

[Fudos and Hoffman 97] I. Fudos and C. Hoffman. "A Graph-Constructive Approach to Solving Systems of Geometric Constraints." *ACM Trans. Graph.* 16 (1997), 179–216.

[Hajnal and Szemerédi 70] A. Hajnal and E. Szemerédi. "Proof of a Conjecture of Erdős." *Combinatorial Theory and Its Applications* 2 (1970), 601–623.

[Heider 46] F. Heider. "Attitudes and Cognitive Organization." *Journal of Psychology* 21 (1946), 107–112.

[Itai and Rodeh 77] A. Itai and M. Rodeh. "Finding a Minimum Circuit in a Graph." In *ACM Symposium on Theory of Computing*, 1977.

[Jeffrey and Ghemawat 04] D. Jeffrey and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters." *Operating Systems Design and Implementation* 51:3 (2004), 107–113.

[Johnson and Lindenstrauss 84] S. Johnson and J. Lindenstrauss. "Extensions of Lipschitz Mappings into a Hilbert Space." *Contemporary Mathematics* 26 (1984), 189–206.

[Jowhari and Ghodsi 05] H. Jowhari and M. Ghodsi. "New Streaming Algorithms for Counting Triangles in Graphs." In *Computing and Combinatorics*, 710–716, 2005.

[Kang et al. 09] U. Kang, C. Tsourakakis, and C. Faloutsos. "PEGASUS: A Peta-scale Graph Mining System." In *International Conference on Data Mining*, 2009.

[Kang et al. 10] U. Kang, C. Tsourakakis, A. P. Appel, C. Faloutsos, and J. Leskovec. "Radius Plots for Mining Tera-byte Scale Graphs: Algorithms, Patterns, and Observations." *SIAM Data Mining* 548–558, 2010.

[Kim and Vu 00] J. H. Kim and V. H. Vu. "Concentration of Multivariate Polynomials and Its Applications." *Combinatorica* 20:3 (2000), 417–434.

[Kolountakis et al. 10] M. N. Kolountakis, G. L. Miller, R. Peng, and C. E. Tsourakakis. "Efficient Triangle Counting in Large Graphs via Degree-Based Vertex Partitioning." In *Workshop on Algorithms and Models for the Web Graph*, 2010.

[Knuth 97] D. Knuth. *Seminumerical Algorithms*, 3rd edition. Reading, MA: Addison-Wesley Professional, 1997.

[Latapy 08] M. Latapy. "Main-Memory Triangle Computations for Very Large (Sparse (Power-Law)) Graphs." *Theoretical Computer Science* 407 (2008), 458–473.

[Leskovec et al. 10] J. Leskovec, D. Huttenlocher, and J. Kleinberg. "Predicting Positive and Negative Links in Online Social Networks." In *International Conference on World Wide Web*, pp. 641–650, 2010.

[Leskovec et al. 08] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. "Microscopic Evolution of Social Networks." In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 462–470, 2008.

[Li et al. 04] S. Li, C. M. Armstrong, N. Bertin, H. Ge., et al. "A Map of the Interactome Network of the Metazoan *C. elegans*." *Science* 303 (2004), 540–543.

[Magen and Zouzias 08] A. Magen and A. Zouzias. "Near Optimal Dimensionality Reductions That Preserve Volumes." In *Randomization and Approximation Techniques in Computer Science*, 523–534, 2008.

[Mislove et al. 07] A. Mislove, M. Massimiliano, K. Gummadi, P. Druschel, and B. Bhattacharjee. "Measurement and Analysis of Online Social Networks." In *Internet Measurement Conference*, 2007.

[Newman 03] M. Newman. "The Structure and Function of Complex Networks." *SIAM Review* 45:2 (2003), 167–256.

[Newman et al. 02] M. E. J. Newman, D. J. Watts, and S. Strogatz. "Random Graph Models of Social Networks." *Proceedings of the National Academy of Sciences of the United States of America* 99 (2002), 2566–2572.

[Pagh and Tsourakakis 12] C. Pagh and C. E. Tsourakakis. "Colorful Triangle Counting and a MapReduce Implementation." *Information Processing Letters* 112:7 (2012), 227–281.

[Papadimitriou and Yannakakis 81] C. Papadimitriou and M. Yannakakis. "The Clique Problem for Planar Graphs." *Information Processing Letters* 13 (1981), 131–133.

[Robins et al. 07] G Robins, P. Pattison, Y. Kalish, and D. Lusher. "An Introduction to Exponential Random Graph (p*) Models for Social Networks." *Social Networks Journal, Special Section: Advances in Exponential Random Graph (p*) Models* 29 (2007), 173–191.

[Rougemont and Hingamp 03] J. Rougemont and P. Hingamp. "DNA Microarray Data and Contextual Analysis of Correlation Graphs." *BMC Bioinformatics* 4 (2003), 4–15.

[Schank and Wagner 05a] T. Schank and D. Wagner. "Finding, Counting and Listing All Triangles in Large Graphs: An Experimental Study." In *Workshop on Experimental and Efficient Algorithms*, 2005.

[Schank and Wagner 05b] T. Schank and D. Wagner. "Approximating Clustering Coefficient and Transitivity." *Journal of Graph Algorithms and Applications* 9 (2005), 265–275.

[Tsourakakis 08] C. E. Tsourakakis. "Fast Counting of Triangles in Large Real Networks, without Counting: Algorithms and Laws." In *International Conference on Data Mining*, 2008.

[Tsourakakis 11] C. E. Tsourakakis. "Counting Triangles Using Projections." *Knowledge and Information Systems* 26:3 (2011), 501–520.

[Tsourakakis et al. 09] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos. "Doulion: Counting Triangles in Massive Graphs with a Coin." In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2009.

[Tsourakakis et al. 10] C. E. Tsourakakis, P. Drineas, E. Michelakis, I. Koutis, and C. Faloutsos. "Spectral Counting of Triangles via Element-wise Sparsification and Triangle-Based Link Recommendation." *Advances in Social Networks Analysis and Mining* (2010).

[Tsourakakis et al. 11] C. E. Tsourakakis, M. Kolountzakis, and G. L. Miller. "Triangle Sparsifiers." *Journal of Graph Algorithms and Applications* 15:6 (2011), 703–726.

[Vu 00] V. H. Vu. "On the Concentration of Multivariate Polynomials with Small Expectation." *Random Structures and Algorithms* 16 (2000), 344–363.

[Wasserman and Faust 94] S. Wasserman and K. Faust. "Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)." Cambridge UK: Cambridge University Press, 1994.

[Wasserman and Pattison 96] S. Wasserman and P. Pattison. "Logit Models and Logistic Regressions for Social Networks: I. An Introduction to Markov Graphs and p*." *Psychometrika* 61 (1996), 401–425.

[Watts and Strogatz 98] D. Watts and S. Strogatz. "Collective Dynamics of Small-World Networks." *Nature* 393 (1998), 440–442.

[Wesler et al. 07] H. Wesler, E. Gleave, D. Fisher, and M. Smith. "Visualizing the Signatures of Social Roles in Online Discussion Groups." *Journal of Social Structure* 8 (2007).

[Wimmer and Lewis 10] A. Wimmer and K. Lewis. "Beyond and Below Racial Homophily: ERG Models of a Friendship Network Documented on Facebook." *American Journal of Sociology* 2 (2010), 583–642.

[Yook et al. 04] S. Yook, Z. Oltvai, and A. L. Barabasi. "Functional and Topological Characterization of Protein Interaction Networks." *Proteomics* 4 (2004), 928–942.

Mihail N. Kolountzakis, Department of Mathematics, University of Crete, Knossou Ave., 71409 Heraklion, Greece (kolount@math.uoc.gr)

Gary L. Miller, School of Computer Science, Carnegie Mellon University, 5000 Forbes Av., Pittsburgh, PA 15213, USA (glmiller@cs.cmu.edu)

Richard Peng, School of Computer Science, Carnegie Mellon University, 5000 Forbes Av., Pittsburgh, PA 15213, USA (ypeng@cs.cmu.edu)

Charalampos E. Tsourakakis, Department of Mathematical Sciences, Carnegie Mellon University, 5000 Forbes Av., Pittsburgh, PA 15213, USA (ctsourak@math.cmu.edu)

Received April 2, 2011; accepted July 15, 2011.