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

Mihail N. Kolountzakis[1], Gary L. Miller[2], Richard Peng[2], Charalampos E. Tsourakakis[3]

Department of Mathematics, University of Crete, Greece
kolount@math.uoc.gr
School of Computer Science, Carnegie Mellon University, USA
glmiller@cs.cmu.edu,yangp@cs.cmu.edu
Department of Mathematical Sciences, Carnegie Mellon University, USA
ctsourak@math.cmu.edu

Abstract. The number of triangles is a computationally expensive graph statistic which is 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 which run fast, use small amount of space, provide accurate estimates of the number of triangles and preferably are parallelizable.
In this paper we present an efficient triangle counting approximation algorithm which can be adapted to the semistreaming model [23]. The key idea of our algorithm is to combine the sampling algorithm of [51,52] and the partitioning of the set of vertices into a high degree and a low degree subset respectively as in [5], treating each set appropriately. From a mathematical perspective, we show a simplified proof of [52] which uses the powerful Kim-Vu concentration inequality [31] based on the Hajnal-Szemerédi theorem [25]. Furthermore, we improve bounds of existing triple sampling techniques based on a theorem of Ahlswede and Katona [3]. We obtain a running time $O\left(m + \frac{m^{3/2}\log n}{t\epsilon^2}\right)$ 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 with space usage $O\left(m^{1/2}\log n + \frac{m^{3/2}\log n}{t\epsilon^2}\right)$ and a constant number of passes (three) over the graph stream. We apply our methods in 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.
Note: This is the extended version of our proceedings paper [32].

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 [16]. The problem of counting subgraphs is one of the typical graph mining tasks that has attracted a lot of attention. The most basic, non-trivial subgraph, is the triangle. Many social networks are abundant in triangles, since typically friends of friends tend to become friends themselves [56]. This phenomenon is observed in other types of networks as well (biological, online networks etc.) and is one of the main reasons which gave rise to the definitions of the transitivity ratio and the clustering coefficients of a graph in complex network analysis [40]. In Section 2.2 we provide an extensive list of applications in which triangles are involved. Given the importance of triangle counting and the scale of several real world networks which reach 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 which provides a $(1 \pm \epsilon)$ approximation to the number of triangles in the graph and runs in $O\left(m + \frac{m^{3/2}\log n}{t\epsilon^2}\right)$ 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 by Tsourakakis et al. in [51,52] with the partitioning idea of Alon, Yuster and Zwick [5] in order 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 $O\left(m^{1/2}\log n + \frac{m^{3/2}\log n}{t\epsilon^2}\right)$ space. We apply our methods in various

networks with several millions of edges and we obtain excellent results both with respect to 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 as $1, 2, .., n$ and edge set E. Let m, t be the number of edges and triangles in G respectively. $deg(u)$ stands for the 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 an edge is contained in, 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) [56,60] and according to the latter friends of friends tend to become friends themselves [56]. 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 [11] 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 involving triangles and ranging from social networks which are of main interest to our work to computer aided design applications.

Clustering Coefficients and Transitivity of a Graph Watts and Strogatz [58] in their influential paper proposed a simple model which explains several contradicting 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 of the graph, see Equation 1. The definition of the transitivity $T(G)$ of a graph G, introduced by Newman et al. [41], 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 [41] erroneously claim that $C(G)$ is the same as $T(G)$, see also [47]. The exact definitions of the aforementioned quantities follow.

Definition 1 (Clustering Coefficient). *A vertex $v \in V(G)$ with degree $deg(v)$ has clustering coefficient $C(v)$ equal to the fraction of edges among its neighbors to the maximum number of triangles it could participate:*

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

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

Definition 2 (Transitivity). *The transitivity ratio $T(G)$ of a graph G is defined as $T(G) = \frac{3 \times t}{\sum_{v \in V(G)} \binom{deg(v)}{2}}$.*

Uncovering Hidden Thematic Structures Eckmann and Moses [21] 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 meta-information. This idea has found more applications, e.g., in bioinformatics [45].

Exponential Random Graph Model Frank and Strauss [22] 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 [57] proposed the exponential random graph (ERG) model which generalized the Markov graphs [44]. Triangles are frequently used as one of the sufficient statistics of the ERG model and counting them is necessary for parameter estimation, e.g., using MCMC procedures [10].

Spam Detection Becchetti et al. [8] 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 stack exchange 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 was studied in [59]. The number of triangles that a user participates was shown to play a critical role in answering these questions. For further applications in assesing the role behavior of users see [8].

Structural Balance and Status Theory *Balance theory* appeared first in Heider's seminal work [26] and is based on the concept "the friend of my friend is my friend", "the enemy of my friend is my enemy" etc. [56]. 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 than 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 study which of the two aforementioned theories applies better 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 [9].

Motif Detection Triangles are abundant not only in social networks but in biological networks [37,61]. This fact can be used to correlate the topological and functional properties of protein interaction networks [61].

CAD applications Fudos and Hoffman [24] 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, 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 which are faster and output a high quality estimate are desirable for the practical applications in which we are interested in this work.

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 [5] 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's 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. Itai and Rodeh in 1978 showed an algorithm which finds a triangle in any graph in $O(m^{\frac{3}{2}})$ [27]. 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 [15]. For special types of graphs more efficient triangle counting algorithms exist. For instance in planar graphs, triangles can be found in $O(n)$ time [15,27,43].

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 it's 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 [34,46].

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 [7]. 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 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 [47], we choose three nodes at random repeatedly 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' which is equal to the fraction of selected triples that form triangles times $\binom{n}{3}$, 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 [7] 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 [4] (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. 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 the naive sampling. The AMS algorithms are also used by [29], 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, [13] proposed two space-bounded sampling algorithms to estimate the number of triangles. Again, the underlying sampling procedures are simple. E.g., for the case of the edge stream representation, they sample randomly an edge and a node in the stream and check if they form a triangle. Their algorithms are the state-of-the-art algorithms to 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 done in order to get concentration (these draws are done in parallel), is of the order

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

Even if the term T_0 is missing compared to the naive sampling, the graph has still to be fairly dense with respect to the number of triangles in order to get an ϵ approximation with high probability. In [51] 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. Interestingly, the same sampling scheme works for randomized matrix and tensor decompositions [2,49]. It was shown later by Tsourakakis, Kolountzakis and Miller [52] using a powerful theorem due to Kim and Vu [31] 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 which colors the vertices of G using N colors uniformly at random and counts monochromatic triangles, see [42].

Another line of work is based on linear algebraic arguments. Specifically, in the case of "power-law" networks Tsourakakis showed in [48] that the spectral counting of triangles can be efficient due to their special spectral properties [17] and [50] extended this idea using the randomized algorithm of [20] by proposing a simple biased node sampling. In [8] 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 [12] are applicable to the problem. More recently, Avron proposed a new approximate triangle counting method based on a randomized algorithm for trace estimation [6].

2.4 Theoretical Preliminaries

Concentration of Measure In Section 3 we make extensive use of the following version of the Chernoff bound [14].

Theorem 1. *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*

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

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 [28] is the following:

Lemma 1 (Johnson 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.

Extremal Graph Theory Hajnal and Szemerédi [25] proved in 1970 the following conjecture of Paul Erdös:

Theorem 2 (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 equals twice the number of the edges. The following theorem was given in [3] and answers this question.

Lemma 2 (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 quasi-star graph or the quasi-complete graph.*

For further progress on other questions related to the above optimization problem such as when does the optimum occur at both graphs, see the work of Abrego, Fernández-Merchant, Neubauer and Watkins [1].

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 [51,52] followed by a triple sampling using the idea of vertex partitioning due to Alon, Yuster and Zwick [5]. In the following, we shall assume that the input is in the form of an edge file, i.e., a file whose each line contains an edge. Notice that given this representation, computing the degrees takes linear time.

3.1 Edge Sparsification

The following method was introduced in [51] and was 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 as 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 and [33]).

In a later analysis [52], 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 show 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 which is 1 if the triangle is kept by the sampling and 0 otherwise. Then as the triangles do not have any edges in common, the X_is are independent and take value 0 with probability $1 - p^3$ and 1 with probability p^3. So by Chernoff bound

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

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 like k to be large. To show that such a large set of independent triangles exist, we invoke the Hajnal-Szemerédi Theorem 2 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 which 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 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 1 and the Chernoff bound allows us to prove the next theorem.

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

Theorem 3. *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 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 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 include 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 i^{th} 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 $E(X_i) = \frac{t}{|U|}$ and X_is are independent. So by Chernoff bound we obtain:

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

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. As $|U| \leq n^3$, 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 which allows us to decide whether we will sample a triple of vertices or an edge and a vertex is the following. Consider any triple containing vertex u, (u, v, w). Since $uv, uw \in E$, we have the number of such triples involving u is at most $deg(u)^2$. From an edge-vertex sampling point of view, as $vw \in E$, 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 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 [3]. Since the maximum degree of such a vertex is $m^{1/2}$, the number of such triangles is upper bounded by $m^{1/2} \cdot (m^{1/2})^2 = m^{3/2}$. Also, as the sum of all degrees is $2m$, there can be at most $2m^{1/2}$ high degree vertices, which means 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 give that $|U|$ can be upper bounded 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 degree 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 among 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, just sampling the first type of triples (aka. pretending 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 only work on $O(mp)$ edges, therefore the total runtime of the triple sampling algorithm after sparsification with probability p becomes:

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

As stated above, since the first term in most practical cases are 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 picked would also depend heavily on constants in front of both terms, as sampling is likely much less expensive due to factors such as cache effect and memory efficiency. Nevertheless, our experimental results in section 4 does seem to indicate that this type of hybrid algorithms can perform better in certain situations.

3.4 Sampling in the Semi-Streaming Model

The previous analysis of triangle counting by Alon, Yuster and Zwick was done in the streaming model [5], where 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 the edges adjacent to each vertex are given in order [23]. We first need to identify high degree vertices, specifically the ones with degree higher 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. *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 picked in $dm^{1/2}\log n$ iterations is:

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

As there are at most n vertices, applying 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 is comprised of 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/vertices pairs that this edge is involved in. 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 belong to the edge list. We can store all such queries into a hash table as there are at most $O(\frac{m^{3/2}\log n}{t\epsilon^2})$ edges sampled w.h.p. Then going through the graph edges in a single pass and looking them up in table yields the desired answer.

4 Experiments

4.1 Data

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

	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. Dataset sources.

Name	Nodes	Edges	Triangle Count	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[39]	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[39]	1,157,822	2,990,442	4,945,382	Person to Person

Table 2. Datasets used in our experiments.

4.2 Experimental Setup and Implementation Details

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 graphs are from real world web-graphs, some details regarding them are in Table 1 and 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 that time since the amount of memory operations performed in setting up the graph is non-negligible. However, we use a modified IO routine that's much faster than the standard C/C++ scanf.

A major optimization that we used was to sort the edges in the graph and store the input file in the format as 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 like Matlab stores 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 2 edges already in the graph, it suffices to check/query whether the 3rd 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 [33].

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. As generating random variables can be expensive, it's preferrable 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 [33]. This variable clearly belongs to an exponential distribution, and if x is a uniform random number in $(0,1)$, taking $\lceil \log_{(1-p)} x \rceil$ as the value of the random variable, see [33]. The primary advantage of doing so is that sampling can be done while accessing the data in a sequential fashion, which results in much better cache performances.

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 to distinguish between low and high degree vertices, so the simple version is essentially sampling all 'V' shaped triples off each vertex. Note that no sparsification and exact also generates the exact number of triangles. Errors are measured by the absolute value of the difference between the value produced and the exact number of triangles divided by the exact number. The results on error and running time are averaged over five runs. The results are shown in Tables 3, 4.

	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 5 Trials using only triple sampling.

4.4 Remarks

From Table 2 it is evident that social networks are abundant in triangles. For example, the Flickr graph with only ∼1.9M vertices has ∼550M triangles and the Orkut graph with ∼3M vertices has ∼620M triangles. Furthermore, from Table 3 and Table 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 [52] is likely to mitigate this discrepancy. The difference between simple and hybrid sampling are due to the fact that handling the second case of triples has a much worse cache access pattern as it examines vertices that are two hops away. There are alternative implementations of how to handle this situation, which would be interesting for future implementations. A fixed sparsification rate of $p = 10\%$ was used mostly to simplify the setups of the experiments. In practice varying p to look for a rate where the result stabilize is the preferred option [52].

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 with other appoximate 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 mentioned earlier in Section 2.2 there exists a lot of interest in the sociology as already mentioned in Section 2 in signed networks. 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 to us via the Stanford Network Analysis library (SNAP).

	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 5 trials using sparsification and triple sampling.

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 $i, j \in V$ which are connected, i.e., $(i, j) \in E$. Observe that the inner product of the i-th and j-th column of the adjacency matrix of graph G gives the number of triangles that edge (i, j) participates in. 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 [28] to reduce the dimensionality of the points while preserving the inner products which contribute to the count of triangles. Magen and Zouzias [38] have considered a similar problem, namely random projections which preserve approximately the volume for all subsets of at most k points.

According to Lemma 1 projecton $x \to Rx$ 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 as 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 i-the column of A. The quantity we are interested in 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

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

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 that 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

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

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

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

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

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

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 following type

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

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

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

Therefore, to have concentration it is sufficient that

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

Observe that (2) is a sufficient -and not necessary- condition. Furthermore,(2) is certainly not always true as there are graphs with many 6-circuits and no triangles at all (the circuits *may* repeat vertices or edges).

6 Conclusions & Future Work

In this work, we extend previous work [51,52] by introducing the powerful idea of Alon, Yuster and Zwick [5]. Specifically, we propose a Monte Carlo algorithm which approximates the true number of triangles within ϵ and runs in $O\left(m + \frac{m^{3/2} \log n}{t\epsilon^2}\right)$ time. Our method can be extended to the semi-streaming model using three passes and a memory overhead of $O\left(m^{1/2} \log n + \frac{m^{3/2} \log n}{t\epsilon^2}\right)$.

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 (2)? Furthermore, our proposed methods are easily parallelizable and developing such an implementation in the MAPREDUCE framework, see [19] and [**?**,54], is a natural practical direction. Finally, coming up with an easy-to-compute quantity which would allow us to sparsify more efficiently is an interesting question.

7 Acknowledgments

The authors gratefully acknowledge support from the University of Crete, Grant No. 2569, and the National Science Foundation, Grants No. IIS-0705359, No. CCF-0635257 and No. CCF-1013110.

References

1. Abrego, B., Fernandez-Merchant, S., Neubauer, M., Watkins, W.: *Sum of squares of degrees in a graph.* Journal of Inequalities in Pure and Applied Mathematics, Vol. 10 (2009)
2. Achlioptas, D., Mcsherry, F.: *Fast computation of low-rank matrix approximations.* Journal of the ACM, Vol. 54 (2007)
3. Ahlswede, R., Katona, G.O.H.: *Graphs with maximal number of adjacent pairs of edges.* Acta Mathematica Academiae Scientarium Hungaricae, Vol. 32, pp. 97-120 (1978)
4. Alon, N., Yossi, M., Szegedy, M.: *The space complexity of approximating the frequency moments.* ACM Symposium on Theory of Computing (1996)
5. Alon, N., Yuster, R., Zwick, U.: *Finding and Counting Given Length Cycles.* Algorithmica, Vol. 17, pp. 209–223 (1997)
6. Avron, H.: *Counting triangles in large graphs using randomized matrix trace estimation.* Workshop on Large-scale Data Mining: Theory and Applications (2010)
7. 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)
8. 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)
9. Berry, J.W., Hendrickson, B., LaViolette, R., Phillips, C.A.: *Tolerating the Community Detection Resolution Limit with Edge Weighting.* Phys. Rev. E, Vol. 83(5)
10. 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)
11. Bonato, A., Hadi, N, Horn, P., Pralat, P., Wang, C.: *Models of Online Social Networks.* Internet Mathematics, Vol. 6(3), 285-313, (2009)
12. Broder, A.Z., Charikar, M., Frieze, A., Mitzenmacher, M.: *Min-wise independent permutations.* ACM Symposium on Theory of Computing (1998)
13. Buriol, L., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., Sohler, C.: *Counting Triangles in Data Streams.* Symposium on Principles of Database Systems (2006)
14. Chernoff, H.: *A Note on an Inequality Involving the Normal Distribution.* Annals of Probability, Vol. 9(3), pp. 533-535 (1981)
15. Chiba, N., Nishizeki, T.: *Arboricity and Subgraph Listing Algorithms.* SIAM Journal on Computing, Vol. 14(1), pp. 210-223 (1985)
16. Chung Graham, F., Lu, L.: *Complex Graphs and Networks.* American Mathematical Society, No. 107 (2006)
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. Dean, J., Ghemawat, S.: *MapReduce: Simplified Data Processing on Large Clusters.* Operating Systems Design and Implementation, Vol. 51(3), pp. 107-113 (2004)
20. Drineas, P., Frieze, A., Kannan, R., Vempala, S., Vinay, V.: *Clustering in Large Graphs and Matrices* ACM Symposium on Discrete Algorithms (1999)
21. 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)
22. Frank, O., Strauss, D.: *Markov Graphs.* Journal of the American Statistical Association, Vol. 81(395), pp. 832-842 (1986)

23. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: *On graph problems in a semi-streaming model.*, Journal of Theoretical Computer Science, Vol. 348(2), pp. 207-216 (2005)
24. Fudos, I., Hoffman, C.: *A Graph-Constructive Approach to Solving Systems of Geometric Constraints* ACM Trans. Graph., Vol. 16, pp. 179-216 (1997)
25. Hajnal, A. and Szemerédi, E: *Proof of a Conjecture of Erds.* Combinatorial Theory and Its Applications, Vol. 2, pp. 601-623 (1970)
26. Heider, F.: *Attitudes and Cognitive Organization* Journal of Psychology, Vol. 21, 107-112 (1946)
27. Itai, A., Rodeh, M.: *Finding a minimum circuit in a graph.* ACM Symposium on Theory of Computing (1977)
28. Johnson, W., Lindenstrauss, J. : *Extensions of Lipschitz mappings into a Hilbert space.* Contemporary Mathematics, vol. 26, pp. 189-206, (1984)
29. Jowhari, H., Ghodsi, M.: *New Streaming Algorithms for Counting Triangles in Graphs.* Computing and Combinatorics (2005)
30. Kang, U, Tsourakakis, C., Faloutsos, C.: *PEGASUS: A Peta-Scale Graph Mining System.* IEEE International Conference on Data Mining (2009)
31. Kim, J.H., Vu, V.H: *Concentration of multivariate polynomials and its applications.* Combinatorica, Vol. 20(3), pp. 417-434 (2000)
32. Kolountakis, M.N., Miller, G.L., Peng, R., Tsourakakis, C.E.: *Efficient Triangle Counting in Large Graphs via Degree-based Vertex Partitioning.* Workshop on Algorithms and Models for the Web Graph (2010)
33. Knuth, D.: *Seminumerical Algorithms.* Addison-Wesley Professional, 3rd edition (1997)
34. Latapy, M.: *Main-memory triangle computations for very large (sparse (power-law)) graphs.* Theoretical Computer Science, Vol. 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. Magen, A., Zouzias, A.: *Near Optimal Dimensionality Reductions That Preserve Volumes.* Randomization and Approximation Techniques in Computer Science, pp. 523-534 (2008)
39. Mislove, A., Massimiliano, M., Gummadi, K., Druschel, P., Bhattacharjee, B.: *Measurement and Analysis of Online Social Networks.* Internet Measurement Conference (2007)
40. Newman, M.: *The structure and function of complex networks.* SIAM Review, Vol. 45(2), pp. 167-256, 2003
41. 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)
42. Pagh, C., Tsourakakis, C.E.: *Colorful Triangle Counting and a MapReduce Implementation.* Information Processing Letters, Vol. 112(7), pp. 227-281 (2012)
43. Papadimitriou, C., Yannakakis, M.: *The clique problem for planar graphs.* Information Processing Letters, Vol. 13, pp. 131-133 (1981)
44. 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)
45. Rougemont, J. and Hingamp, P: *DNA microarray data and contextual analysis of correlation graphs.* BMC Bioinformatics, Vol. 4 (2003)
46. Schank, T., Wagner, D.: *Finding, Counting and Listing all Triangles in Large Graphs, An Experimental Study.* Workshop on Experimental and Efficient Algorithms (2005)
47. Schank, T., Wagner, D.: *Approximating Clustering Coefficient and Transitivity.* Journal of Graph Algorithms and Applications, Vol. 9, pp. 265-275 (2005)
48. Tsourakakis, C.E.: *Fast Counting of Triangles in Large Real Networks, without counting: Algorithms and Laws.* IEEE International Conference on Data Mining (2008)
49. Tsourakakis, C.E.: *MACH: Fast Randomized Tensor Decompositions* SIAM International Conference on Data Mining (2010)
50. Tsourakakis, C.E.: *Counting Triangles Using Projections.* Knowledge and Information Systems (2011)
51. 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)
52. Tsourakakis, C.E., Kolountzakis, M., Miller, G.L.: *Triangle Sparsifiers.* Journal of Graph Algorithms and Applications (JGAA), Vol. 15(6), pp. 703-726 (2011)

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. Tsourakakis, C.E.: *Large Scale Graph Mining with MapReduce: Diameter Estimation and Eccentricity Plots of Massive Graphs with Mining Applications.* Social Network Mining, Analysis and Research Trends: Techniques and Applications edited by Ting, I-H., Hong, T.-P., Wang, L.
55. V.H. Vu, *On the concentration of multivariate polynomials with small expectation.* Random Structures and Algorithms, Vol. 16, pp. 344-363 (2000)
56. Wasserman, S., Faust, K.: *Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences).* Cambridge University Press (1994)
57. 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)
58. Watts, D., Strogatz, S.: *Collective dynamics of small-world networks.* Nature, Vol. 393, pp. 440-442 (1998)
59. 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)
60. 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)
61. Yook, S., Oltvai, Z., Barabasi, A.L.: *Functional and topological characterization of protein interaction networks.* Proteomics, Vol. 4, pp. 928-942 (2004)