On the Streaming Complexity of Computing Local Clustering Coefficients[*]

Konstantin Kutzkov
IT University of Copenhagen
Copenhagen, Denmark
konk@itu.dk

Rasmus Pagh
IT University of Copenhagen
Copenhagen, Denmark
pagh@itu.dk

ABSTRACT

Due to a large number of applications, the problem of estimating the number of triangles in graphs revealed as a stream of edges, and the closely related problem of estimating the graph's clustering coefficient, have received considerable attention in the last decade. Both efficient algorithms and impossibility results have shed light on the computational complexity of the problem. Motivated by applications in Web mining, Becchetti et al. presented new algorithms for the estimation of the *local* number of triangles, i.e., the number of triangles incident to *individual* vertices. The algorithms are shown, both theoretically and experimentally, to efficiently handle the problem. However, at least two passes over the data are needed and thus the algorithms are not suitable for real streaming scenarios.

In the present work, we consider the problem of estimating the clustering coefficient of individual vertices in a graph over n vertices revealed as a stream of m edges. As a first result we show that any one pass randomized streaming algorithm that can distinguish a graph with no triangles from a graph having a vertex of degree d with clustering coefficient $> 1/2$ must use $\Omega(m/d)$ bits of space in expectation.

Our second result is a new randomized one pass algorithm estimating the local clustering coefficient of each vertex with degree at least d. The space requirement of our algorithm is within a logarithmic factor of the lower bound, thus our approach is close to optimal. We also extend the algorithm to local triangle counting and report experimental results on its performance on real-life graphs.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

[*]Work supported by the Danish National Research Council under the Sapere Aude program.

Keywords

Graph algorithms, Streaming, Local clustering coefficient, Triangle counting

1. INTRODUCTION

The last decades have witnessed a rapid growth of available data. Many of the best known algorithms assume random access to the input, and this turns out to be an infeasible requirement for many real-life problems. This motivates the *(semi-) streaming* model of computation where only one or a small number of sequential scans of the input are allowed. The model has become a ubiquitous computational paradigm and considerable progress has been made towards a better understanding of the complexity of many basic algorithmic problems.

Graphs are a ubiquitous representation of relationships between objects in real-life problems. Many applications require knowledge of certain structural properties of a given graph which will allow a better analysis of the original problem. The clustering coefficient of a graph, introduced in [32], gives the probability that two of the neighbors of a vertex chosen at random are connected by an edge. It provides important information about the structure of the graph and the existence of local communities. The computation of the clustering coefficient of a graph, and the closely related problem of computing the total number of triangles in a graph revealed as a stream of edges, have received considerable attention. Becchetti et al. [6] showed that for certain applications in Web mining the computation of the clustering coefficient of individual vertices is required, and presented a semi-streaming algorithm for the problem. In this paper we study whether the problem admits an efficient solution in a real streaming scenario when only one pass over the input graph is allowed.

2. PRELIMINARIES

Notation.

Let $G = (V, E)$ be a simple undirected graph without loops. We denote the number of vertices V by n and the number of edges E by m. An edge between the vertices u and v is written as (u, v). We assume a total order on vertices in V and since our graph is undirected and loopless we will consider only edges (u, v) with $u < v$. $N_G(v) = \{u \in V : (u, v) \in E\}$ is the *neighborhood* of v. The *degree* of a vertex v is $d_v = |N_G(v)|$, i.e., the number of edges adjacent to v. An r-*clique* K_r in G is a subgraph of G on r vertices such that for any vertex pair u, v in K_r there exists an edge $(u, v) \in E$. A *triangle* is a 3-clique. $T_v = \frac{1}{2}|\{(u, w) \in E : (u, v) \in E, (v, w) \in E\}|$ is the number of triangles containing v. A *2-path centered at* v is a triple of vertices (u, v, w) such that there exist edges $(u, v), (v, w)$ and u and w are called the *endpoints* of the 2-path (u, v, w). A triangle on the vertices u, v, w is denoted by $\langle u, v, w\rangle$. The *clustering coefficient* of a vertex v of degree at least two, denoted as C_v, is defined as the ratio of triangles containing v to the number of 2-paths centered at v. More formally, we define

$$C_v = \frac{2T_v}{d_v(d_v - 1)}$$

The *global* clustering coefficient of G is

$$G_G = \frac{1}{n} \sum_{v \in V : d(v) \geq 2} C_v.$$

Note that we define the local clustering coefficient only for vertices of degree at least two. In the literature the clustering coefficient of vertices of degree less than two traditionally has been set to either 0 or 1, or declared to be meaningless. As discussed in several works, see e.g. [18, 26], the different definitions have resulted in different results. Therefore, in order to avoid confusions, we consider only graphs with vertices of degree at least two. This is also justified by the real-life graphs we consider in our experiments, since they do not contain vertices of degree less than two.

Probability theory.

We assume that the reader is familiar with basic definitions from probability theory. In the analysis of our algorithms we will use these inequalities:

- *Markov's inequality* Let X be a random variable. Then for every $\lambda > 1$:

$$\Pr[X \geq \lambda E[X]] \leq \frac{1}{\lambda} \tag{1}$$

- *Chebyshev's inequality.* Let X be a random variable and $s > 0$. Then

$$\Pr[|X - E[X]| \geq s] \leq \frac{V[X]}{s^2} \tag{2}$$

- *Chernoff's inequality.* We will use the following form of the inequality:

  Let $X_1, \dots, X_\ell$ be ℓ independent identically distributed Bernoulli random variables and $E[X_i] = \mu$. Then for any $\varepsilon > 0$ we have

$$\Pr[|\frac{1}{\ell} \sum_{i=1}^{\ell} X_i - \mu| > \varepsilon\mu] \leq 2e^{-\varepsilon^2 \mu\ell/2} \tag{3}$$

A family $\mathcal{F}$ of functions from V to a finite set S is *k-wise independent* if for a function $f : V \to S$ chosen uniformly at random from $\mathcal{F}$ it holds

$$\Pr[f(v_1) = c_1 \wedge f(v_2) = c_2 \wedge \cdots \wedge f(v_k) = c_k] = \frac{1}{s^k}$$

for $s = |S|$, distinct $v_i \in V$ and any $c_i \in S$ and $k \in \mathbb{N}$.

A family $\mathcal{H}$ of functions from V to a finite totally ordered set S is called (ε, k)-*min-wise independent* if for any $X \subseteq V$ and $Y \subseteq X$, $|Y| = k$, for a function h chosen uniformly at random from $\mathcal{H}$ it holds

$$\Pr[\max_{y \in Y} h(y) < \min_{z \in X \setminus Y} h(z)] = (1 \pm \varepsilon)\frac{1}{\binom{|X|}{k}}$$

We will refer to a function chosen uniformly at random from a k-wise independent family as a k-wise independent function.

We will say that an algorithm returns an (ε, δ)-*approximation* of some quantity q, if it returns a value $\tilde{q}$, such that $(1 - \varepsilon)q \leq \tilde{q} \leq (1 + \varepsilon)q$, with probability at least $1 - \delta$ for any $0 < \varepsilon, \delta < 1$.

The streaming model.

We assume that the input graph is read as a stream of edges. In the literature, two models have been considered. In the *incidence list* stream model for each vertex $u \in V$ the edges $(u, v_1), \dots, (u, v_{d_u})$, i.e., all edges adjacent to u, are revealed in succession. In an *adjacency stream* model edges arrive in arbitrary order. Clearly, the incidence list model provides more information and algorithms analyzed in this model are usually better than algorithms assuming the adjacency stream model. Our results apply to the adjacency stream model.

3. PREVIOUS WORK

Counting the number of triangles in a graph and computing its clustering coefficient are widely studied problems, with applications ranging from computational biology to mining social networks. It is known that unlike random Erdös-Rényi graphs [14], real-life graphs are characterized by high clustering coefficients, see for instance [2], and the clustering coefficient is an important metric for mining knowledge about the original data. For example, Coleman [11] and Portes [25] use the clustering coefficient when analyzing human behaviour in social networks, and Becchetti et al. [6] used it for the detection of spamming activity in large-scale Web graphs. The best known exact algorithm [3] uses as a subroutine matrix multiplication and runs in time $O(n^\omega)$, where $\omega = 2.3727$ is the best known exponent for rectangular matrix multiplication [30]. The algorithm also counts exactly the number of triangles per vertex. Note however that this algorithm is mainly of theoretical importance since it requires random access to the graph, and thus the graph has to fit in memory. Also, currently known asymptotically fast matrix multiplication algorithms do not admit an efficient implementation for realistic input sizes.

Several authors presented sampling based algorithms for the estimation of the *global* number of triangles in a (semi)-streaming setting [5, 9, 12, 16, 21, 24, 28, 29]. Building upon results from linear algebra, researchers proposed techniques for approximate triangle counting not relying on sampling [4, 27]. The closely related problem of estimating the global clustering coefficient was considered in [10, 26]. The algorithm by Schank and Wagner [26] is based on sampling and requires two passes over the input graph. The algorithm was improved to work in a single pass by Buriol et al. [10] at the price of a slightly increased time and space complexity. Motivated by the problem of detection of emerging web communities by analyzing the Web graph [22], Buriol et al. [10] studied the problem of counting bipartite cliques of small size. The more general problem of estimating the number of graph minors of fixed size was studied by Bordino et al [7].

Despite of the fact that researchers have considered applications involving counting the number of triangles per vertex [13, 23], only recently the problem of estimating the *local* number of triangles was rigorously studied by Becchetti et al. [6]. The authors propose two algorithms working in a *semi-streaming* fashion such that the input graph is stored on an external device and a few sequential scans over it are allowed. The algorithms are based on min-wise permutation hashing [8]. Both algorithms use $O(n)$ main memory. However, the first algorithm also requires sequential writing to a persistent storage device and uses $O(m)$ external memory. The second algorithm works only in main memory but one cannot theoretically prove that the algorithm achieves an arbitrarily good approximation of the number of triangles at individual vertices. It is easy to adjust the algorithms to work in two passes but there is no straightforward extension of the approach running in a single pass over the input, thus it is not suitable for real streaming applications.

Our contribution.

Algorithms working in only one pass over the input are important since they can be used in real streaming scenarios and thus have a wider range of applications. We study the complexity of estimating the local clustering coefficient by randomized algorithms in one pass over the input from two perspectives:

1. *Lower bound.* We show that any one pass randomized algorithm detecting with error probability at most $1/3$ if there exists a vertex of degree at least $2d$ with clustering coefficient at least $1/2$ needs $\Omega(m/d)$ bits. This holds even if we guarantee that if there exists any triangle, there exists a vertex with clustering coefficient at least $1/2$.

2. *Upper bound.* We design a randomized one pass algorithm deciding with constant error probability whether a given vertex of degree d or more has a clustering coefficient above a given constant threshold α using $O(\frac{m}{d})$ words. More precisely, we achieve an (ε, δ)-approximation of the clustering coefficient α of all vertices of degree at least d in time $O(\frac{m}{\alpha\varepsilon^2} \log \frac{1}{\varepsilon} \log \frac{n}{\delta})$ and space $O((\frac{m}{d} + \log \frac{1}{\varepsilon}) \frac{1}{\alpha\varepsilon^2} \log \frac{n}{\delta})$ for any $\delta, \varepsilon, \alpha > 0$. We extend our algorithm to also estimate the number of triangles per high-degree vertex. At the end we provide an experimental evaluation of the algorithm.

In case d is no larger than the average degree, the space usage is $\Omega(n)$ bits so we are in the semi-streaming domain in terms of space usage. Our algorithm has space usage $o(n)$ whenever the fraction of vertices with degree d or more is sufficiently small. This is the case for many real-life graphs.

Note that the lower bound is given in bits while the upper bound is terms of the number of sampled edges, i.e. machine words of $O(\log n)$ bits each. Thus, for a fixed vertex the lower and upper bound match up to a logarithmic factor.

4. LOWER BOUND

We begin with a negative result that will show the limitations of any randomized one pass algorithm detecting vertices with high clustering coefficient. Lower bounds on the complexity of global triangle counting have been shown in several works [5, 16, 33].

THEOREM 1. *Let $G = (V, E)$ be a simple undirected graph without loops over m edges. Any randomized streaming algorithm, performing only one pass over the edges of G and being able to distinguish between a graph where all vertices of degree $2d = o(\sqrt{m})$ have a clustering coefficient 0, and a graph where there is a vertex of degree $2d$ with clustering coefficient at least $1/2$, with error probability $1/3$, must use $\Omega(m/d)$ bits in expectation.*

PROOF. Let DETECTCLUSTEREDVERTICES be a randomized streaming algorithm detecting vertices with high clustering coefficient in G. Assume without loss of generality that $m/(2d)$ is integer. We show that if DETECTCLUSTEREDVERTICES uses s bits of memory, this would imply a protocol for the SET DISJOINTNESS problem on strings $x, y \in \{0,1\}^{m/(2d)}$ using s bits of communication. This problems has answer 1 if there exists an index i such that $x_i = y_i = 1$, and answer 0 otherwise. By existing lower bounds on the randomized communication complexity of SET DISJOINTNESS [19] this implies that $s = \Omega(m/d)$ bits are needed for any constant error probability smaller than 1.

Denote by x_i the ith bit in the string x. We will reduce the SET DISJOINTNESS problem to detecting vertices with clustering coefficient more than $1/2$ in a graph revealed as a stream of edges in the adjacency stream model. Let $W^x = \{w_1^x, \ldots, w_d^x\}$, $W^y = \{w_1^y, \ldots, w_d^y\}$, $U^x = \{u_1^x, \ldots, u_d^x\}$, $U^y = \{u_1^y, \ldots, u_d^y\}$ be four sets of d vertices each. The reader is referred to Figure 1 for a graphical description of our construction. We read $x \in \{0,1\}^{m/(2d)}$ bit by bit and create a vertex v_i for the ith index in x. If $x_i = 0$, then we feed DETECTCLUSTEREDVERTICES with the d edges (v_i, u_j^x), $1 \le j \le d$. If $x_i = 1$, then we feed DETECTCLUSTEREDVERTICES with the d edges (v_i, w_j^x), $1 \le j \le d$. We proceed with the bits in y in the same way but we create edges (v_i, u_j^y) and (v_i, w_j^y).

After processing x and y we add d^2 edges (w_i^x, w_j^y), $1 \le i, j \le d$, for $w_i^x \in W^x, w_j^y \in W^y$. Now it is easy to see that the graph contains a triangle if and only if there exists an i such that $x_i = y_i = 1$, and in this case the clustering coefficient of v_i is $d^2/\binom{2d}{2} > 1/2$, otherwise all vertices have clustering coefficient 0. Also, each vertex v_i has degree exactly $2d$, the total number of vertices is $m/(2d) + 4d$ and for $d = o(\sqrt{m})$ the number of edges is $m + d^2 = O(m)$. Thus, we have reduced the SET DISJOINTNESS problem for input strings $x, y \in \{0,1\}^{m/(2d)}$ to the problem of distinguishing between vertices of degree d with clustering coefficient 0 or more than $1/2$ in streamed graphs over $O(m)$ edges. We conclude that DETECTCLUSTEREDVERTICES needs $\Omega(m/d)$ bits for graphs over m edges. $\square$

One can extend the above proof such that the vertices v_i can also have odd degree in a trivial way. For a better readability we choose not to present this extension here.

The above lower bound supports the intuition that the space requirements of any randomized one pass algorithm for estimating the clustering coefficient of individual vertices, should not depend only on the value of the coefficient but also on the vertex degree. In order to estimate the clustering coefficient of a given vertex v we need some kind of sampling of 2-paths centered at v for which we can check whether they are a part of a triangle. But to obtain a good estimate of low degree vertices, without any prior knowledge about the degree distributions in the graph, we need a low sampling probability, which in turn results in many sampled edges.

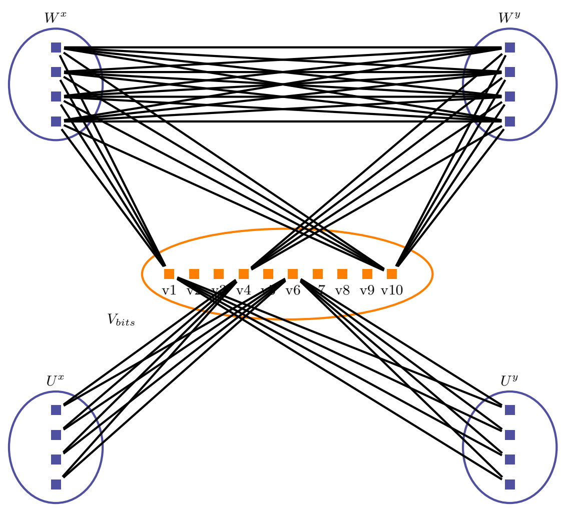

Figure 1: An example of a hard instance for our lower bound construction. The edges going out from v_1 denote that $x_1 = 1$ and $y_1 = 0$. Similarly, we see that $x_4 = 0$ and $y_4 = 1$, while $x_6 = y_6 = 0$ and $x_{10} = y_{10} = 1$. Only v_{10} among the considered vertices is part of a triangle and has a clustering coefficient larger than 1/2.

5. ONE PASS ALGORITHM

In this section we present a randomized one pass algorithm estimating the clustering coefficient of vertices of degree above some threshold d. In Section 5.1 we first give some intuition what are the difficulties for the estimation of the clustering coefficient in only one pass and then describe the main idea behind the new algorithm. A thorough description and theoretical analysis of our algorithm is given is Section 5.2. In Section 5.3 we show how to extend the algorithm to local triangle counting for high-degree vertices. In Section 5.4 we conclude with an experimental evaluation of the performance of the algorithm on real-life graphs.

5.1 Intuition

A straightforward idea for the estimation of the clustering coefficient in only one pass of high degree vertices is to sample independently each incoming edge with a certain probability p. Then in the sparsified graph we compute the clustering coefficient for all vertices. This sparsification approach is essentially the Doulion algorithm [28, 29], which was shown to accurately estimate the number of triangles in a graph given as a stream of edges. Let us consider a vertex v contained in a triangle $\langle u, v, w\rangle$. The probability that we will sample the 2-path (u, v, w) is p^2 while the probability to sample the triangle $\langle u, v, w\rangle$ is p^3. Thus, we have to multiply by $1/p$ the clustering coefficient we have computed for each vertex in the sparsified graph in order to obtain in expectation the true clustering coefficient. But this means that the estimates obtained from the sparsified graph need to be very accurate in order to obtain a concentration around the expected value with high probability, which in turn implies a large value for the sampling probability p resulting in large space consumption.

A simple algorithm can work as follows: for each vertex v we sample independently a number of 2-paths (u, v, w) and for each of them check whether the edge (u, w) exists in the graph. The approach can be implemented in three passes over the edges. In the first pass we find a set of candidate vertices of degree at least d. This can be done by a frequent items mining algorithms, e.g. [20], in space $O(m/d)$ such that each vertex of degree d or more is guaranteed to be among the candidates. In a second pass we determine the set of vertices of degree d or more and the exact degree of each of them. Then for each vertex v of degree d or more we sample a number of 2-paths (u, v, w) and check whether the edge (u, w) appears in the stream, this can be done for example using reservoir sampling [31] as shown in [9]. It is easy to show that the sketched algorithm will compute an (ε, δ)-approximation of a vertex of degree at least d and clustering coefficient α in space $O(\frac{m}{d\varepsilon^2\alpha}\log\frac{1}{\delta})$

In the next section we show how to combine the above two approaches to work for high degree vertices in only one pass over the input graph achieving essentially the same space complexity. We will use the idea of *monochromatic sampling* [24]. Basically, we randomly color the vertices and then take an edge in the sample iff its endpoints have the same color. Clearly, if two of the edges of a triangle are sampled, the third edge must also be sampled. Thus, for a sampled 2-path (u, v, w), after processing the stream of edges we can check whether there exists a triangle $\langle u, v, w\rangle$. The sampling probability for a triangle $\langle u, v, w\rangle$ is the same as for the 2-path (u, v, w), namely p^2.

5.2 The algorithm

A high-level pseudocode description of our algorithm is given in Figure 2. The main method is EstimateClusteringCoefficients. It reads the input graph as a stream of edges in arbitrary order. First we run K independent copies of SparsifyGraph in parallel. Assume we have defined a k-wise independent function $f : V \to [0, 1]^1$, for a k that will be specified later. Let C be a natural number denoting the number of colors. Then we sample each incoming edge (u, v) iff $\lfloor Cf(u)\rfloor = \lfloor Cf(v)\rfloor$, where $\lfloor r\rfloor$ denotes that the real number r is mapped to the biggest integer smaller than (or equal to) r, i.e., we map to one of C colors. At the end we obtain a sparsified graph where all edge endpoints have the same color. Obviously, if $C = d$, we expect sparsified graphs with m/d edges. In order to control the space usage of the algorithm, we check after each edge has been sampled whether not more than t edges are sampled, for a user-defined t. If this is the case, then the algorithm breaks its execution and returns an empty graph.

Clearly, if we have sampled a 2-path (u, v, w), then, if existent, we must have also sampled the edge (u, w). In CheckTwoPaths for each vertex with at least two sampled neighbors we choose the two of these neighbors for which f gives the smallest value and check whether the considered 2-path is part of a triangle. We will show that the expected value is an almost unbiased estimator of the clustering coefficient of vertex v such that the bias depends on how independent is the coloring function. Assume we return $R \leq K$ nonempty graphs. At the end, for each vertex v with at least $R/2$ sampled 2-paths (u, v, w) we output as an estimate of the clustering coefficient α_v the ratio of triangles to 2-paths centered at v obtained from the samples.

[1]In fact, f maps V to a discrete set uniformly distributed on the unit interval.

THEOREM 2. *Let $G = (V, E)$ be a graph over n variables revealed as a stream of m edges. Let further $K = \frac{4}{\alpha\varepsilon^2}\log\frac{n}{\delta}$, $C = d/4$ colors, $f_i : V \to [0, 1]$, $1 \le i \le K$, be k-wise independent, such that $k = O(\log\frac{1}{\varepsilon})$, and $t = 9m/d$. Then* ESTIMATECLUSTERINGCOEFFICIENTS *returns an (ε, δ)-approximation of the clustering coefficient for all vertices of degree at least d and clustering coefficient at least α for any $\varepsilon, \alpha, \delta > 0$. The expected running time of the algorithm is $O(\frac{m}{\alpha\varepsilon^2}\log\frac{1}{\varepsilon}\log\frac{n}{\delta})$ and the worst case space complexity is $O((\frac{m}{d} + \log\frac{1}{\varepsilon})\frac{1}{\alpha\varepsilon^2}\log\frac{n}{\delta})$.*

PROOF. Let us assume that we run K parallel instances of SPARSIFYGRAPH. In the following we will obtain a bound on K that will guarantee the claimed bounds. We will estimate the probability for each of three "bad" events that lead to an incorrect estimate.

Assume first that for a given vertex v we have sampled $\ell \le K$ 2-paths (u_i, v, w_i), each in one of the copies of the algorithm run in parallel. For a given non-empty sparsified graph G_S^i, $1 \le i \le \ell$, introduce an indicator random variable $X_i, 1 \le i \le \ell$, such that $X_i = 1$ iff $(u_i, w_i) \in G_S^i$, i.e., if u, v, w form a triangle. We first show that for a random enough coloring function f_i, $E[X_i]$ is an almost unbiased estimator of α. Let v's clustering coefficient be α_v and let us define a function $h : N(v) \to [0, 1]$ as $h(u) = (f(u) - f(v)) \bmod 1$ for $u \in N(v)$. It is easy to see that the two vertices u, w yielding the smallest values $h(u)$ and $h(w)$ are those yielding the smallest values $f(u)$ and $f(w)$ larger than $f(v)$. Thus, under the assumption that $|N_v(G_S)| \ge 2$, choosing the two neighbors of v in G_S with the smallest values given by f corresponds to choosing the two vertices in $N_v(G)$ with the smallest values given by h. If f is k-wise independent under the assumptions that $f(v)$ evaluates to a certain value, we have that h is $(k-1)$-wise independent. For $k = O(\log\frac{1}{\varepsilon})$, h is then $(\varepsilon, 2)$-min-wise independent, see [15] for the state-of-the-art result. Thus, the chosen pair of vertices is sampled almost uniformly at random among all pairs of v's neighbors and we have $(1-\varepsilon)\alpha_v \le E[X_i] \le (1+\varepsilon)\alpha_v$.

We return $\tilde{\alpha}_v = \frac{1}{\ell}\sum_{i=1}^{\ell} X_i$ as an estimation of α_v. The functions f_i are independent and thus the colorings are also independent. Consequently the indicator random variables X_i are independent, thus by applying Chernoff's inequality and using $\varepsilon < 1$ we bound the probability that $\tilde{\alpha}_v$ is not an $(1 \pm 3\varepsilon)$-approximation is upper bounded by $2^{-\varepsilon^2\alpha_v\ell/2}$. Since $\alpha \le \alpha_v$, we need

$$\ell = \frac{\log 3}{\varepsilon^2\alpha}\log\frac{n}{\delta}$$

such that the error probability for a single vertex is bounded by $\frac{\delta}{3n}$. Thus, the error probability of the estimate for *any* vertex is $\delta/3$.

Second, let us assume that the K parallel instances return R non-empty graphs, i.e., in R cases we have not sampled more than the allowed t edges. We estimate how many independent colorings, each for $C = d/4$ colors, are needed such that we obtain at least $\ell = \frac{\log 3}{\varepsilon^2\alpha}\log\frac{n}{\delta}$ samples for a vertex of degree at least d. Consider a given vertex v with $d_v \ge d$. We have $\frac{d}{4}$ colors and thus a sampling probability of $\frac{4}{d}$. We introduce an indicator random variable $X_1, \dots, X_{d_v}$ for each neighbor u_i of v, denoting whether $f(v) = f(u_i)$. Let $X = \sum_{i=1}^{d_v} X_i$. We want to bound $\Pr[X \le 1]$. Clearly, we have $E[X] = \frac{4d_v}{d} \ge 4$. The X_i are $\{0, 1\}$-valued, and w.l.o.g. we can assume that f is 3-wise independent, therefore it is easy to see that $V[X] \le E[X]$. Thus, we can set $s = E[X] - 1$ in Chebyshev's inequality and obtain

$$\Pr[|X - E[X]| \ge E[X] - 1] \le \frac{V[X]}{(E[X]-1)^2}$$

$$\le \frac{E[X]}{(E[X]-1)^2} \le \frac{4}{9}$$

for $d_v \ge d$. Now, we introduce an indicator random variable Y_i^v for vertex v of degree at least d, denoting whether enough edges have been sampled in the ith sparsified graph, $1 \le i \le S$. Clearly, $E[Y_i^v] \ge 5/9$. Since the colorings are independent, we can apply again Chernoff's inequality and bound the probability for not enough sampled edges. Therefore, if we have

$$R = 2\frac{\log 3}{\varepsilon^2\alpha}\log\frac{n}{\delta}$$

the probability that in $\ell = \frac{\log 3}{\varepsilon^2\alpha}\log\frac{n}{\delta}$ graphs for any vertex of degree at least d we have less than two sampled neighbors, is upper bounded by $\delta/3$.

Finally, we have to consider the possibility that too many edges have been sampled and the algorithm returns an empty graph. By simply applying Markov's inequality for a sampling probability $\frac{4}{d}$ we get that with probability 4/9 more than $\frac{9m}{d}$ edges are sampled. We apply again Chernoff's inequality. Running

$$K = 4\frac{\log 3}{\varepsilon^2\alpha}\log\frac{n}{\delta}$$

copies in parallel, we bound the probability that too many edges are sampled in more than $R = 2\frac{\log 3}{\varepsilon^2\alpha}\log\frac{n}{\delta}$ of the copies, to $\delta/3$.

Summing up, the probability that either too many edges are sampled, or not enough neighbors for a vertex of degree d are sampled, or the algorithm does not return an $(1 \pm \varepsilon)$-approximation of the clustering coefficient for vertices of degree at least d, is $3(\delta/3) = \delta$ for $0 < \delta < 1$.

A $O(\log\frac{1}{\varepsilon})$-wise independent function can be defined in $O(\log\frac{1}{\varepsilon})$ machine words and evaluated in time $O(\log\frac{1}{\varepsilon})$. Note that for a approximation parameter ε that can be described in constant number of machine words, $\log\frac{1}{\varepsilon}$ is constant. Storing the adjacency lists of the vertices in the sparsified graphs in hashtables, we can achieve expected constant update time per incoming edge. The sampling of 2-paths and checking whether they are contained in a triangle can thus be done in expected linear time in the size of the sparsified graph. The size of the hash table H is clearly bounded by the number of sampled edges in the R non-empty graphs and the expected time for update and look-up is constant. The time and space complexity of the algorithm follow then immediately from the above discussion. $\square$

The size of the input graph.

In the above analysis we assume that the number of edges and vertices are known in advance. If this is not the case, one can start with a conservative choice for the number of colors and, if too many edges have been sampled, scale the sampling probability, i.e., increase the number of colors. When presenting the implementation of our algorithm we discuss more details about this, but we omit the rigorous description of this extension.

```
SameColor
Input: a $k$-wise independent function $f : V \to [0,1]$, number of colors $C$, vertices $u$ and $v$
1: if $\lfloor Cf(u) \rfloor = \lfloor Cf(v) \rfloor$ then
2:     return true
3: else
4:     return false
```

```
SparsifyGraph
Input: stream of edges $E$, number of colors $C$, $k$-wise independent function $f : V \to [0,1]$, threshold $t$
1: $G_S = \emptyset$
2: for each edge $(u,v) \in E$ do
3:     if SameColor$(f, C, u, v)$ then
4:         $G_S = G_S \cup (u,v)$.
5:         if $|G_S| > t$ then
6:             return the empty graph.
7: Return $G_S$.
```

```
CheckTwoPaths
Input: a sparsified graph $G_S$
1: for each vertex $v \in G_S$ such that $|N_{G_S}(v)| \geq 2$ do
2:     choose the two neighbors $u, w$, $u < w$, from $N_{G_S}(v)$ with the smallest $f(u)$ and $f(w)$
3:     sampled$(v)$ = true
4:     if $(u,w) \in G_S$ then
5:         $X_\triangle(v) = 1$ //there is a triangle
6:     else
7:         $X_\triangle(v) = 0$ //no triangle
8: Return a set of pairs $(v, X_\triangle(v))$ such that $v \in V$ and sampled$(v)$
```

```
EstimateClusteringCoefficients
Input: a stream of edges $E$, a family of $K$ $k$-wise functions $f_i : V \to [0,1]$, number of colors $C$, Hashtable $H\langle$V, $\langle$Int, Int$\rangle\rangle$, int $K$, threshold $t$
1: Run in parallel $K$ copies of SparsifyGraph$(E, C, f_i, t)$
2: for each of $R$ returned non-empty sparsified graph $G_S^i$ do
3:     $V_P$ = CheckTwoPaths$(G_S^i)$
4:     for each $(v, X_\triangle(v)) \in V_P$ do
5:         if $v \notin H$ then
6:             $p_v = t_v = 0$
7:         else
8:             get $(v, (p_v, t_v))$ from $H$
9:         $p_v$++
10:        if $X_\triangle(v) == 1$ then
11:            $t_v$++
12:        put $(v, (p_v, t_v))$ in $H$
13: for each $(v, (p_v, t_v)) \in H$ with $p_v \geq R/2$ do
14:     Return $(v, t_v/p_v)$.
```

Figure 2: A high-level pseudocode description of the algorithm.

5.3 Local triangle counting

It is easy to extend the algorithm to count the local number of triangles for vertices of sufficiently high degree. All we need is to also estimate the number of 2-paths per high-degree vertex. Indeed, we already have the necessary components for the analysis. In the following theorem we show how to estimate the degree of a vertex for which we have sampled at least $R/2$ times at least two neighbors and that for high degree vertices we can obtain a high quality estimate of the degree. We will use the same notation as in the proof of Theorem 2.

THEOREM 3. *Let $G = (V, E)$ be a graph over n variables revealed as a stream of m edges. There exists a randomized one pass algorithm achieving an (ε, δ)-approximation of the number of triangles centered at each vertex of degree at least d and a clustering coefficient at least α in expected time $O(\frac{m}{\alpha\varepsilon^2}\log\frac{1}{\varepsilon}\log\frac{n}{\delta})$ and worst case space complexity $O((\frac{m}{d} + \log\frac{1}{\varepsilon})\frac{1}{\alpha\varepsilon^2}\log\frac{n}{\delta})$ for any $\varepsilon, \alpha, \delta > 0$.*

PROOF. Consider a vertex v of degree $d_v \geq d$. We first show how to obtain an approximation of d_v. Assume we extend EstimateClusteringCoefficients in the following way. For each vertex with at least one sampled neighbor we record the number of sampled neighbors in each sparsified graph.

Assume we return $S = \frac{3}{\varepsilon'^2}\log\frac{n}{\delta}$ non-empty sparsified graphs for some $\varepsilon' > 0$ that will be specified later. Let us logically divide them into $\log\frac{n}{\delta}$ groups of $\frac{3}{\varepsilon'^2}$ graphs. In each group for each sparsified graph we introduce an indicator random variable X_i, $1 \leq i \leq d_v$, for each of the d_v neighbors of v. Let $X = \sum_{i=1}^{\frac{3d_v}{\varepsilon'^2}} X_i$, i.e., X is the sum of the X_i in all groups. Clearly, $E[X] = \frac{3d_v}{d\varepsilon'^2}$. Also, we can assume that the coloring function is 3-wise independent and the $\frac{3}{\varepsilon'^2}$ colorings are independent, thus we have $V[X] \leq E[X]$. We apply Chebyshev's inequality with $s = \varepsilon' E[X]$:

$$\Pr[|X - E[X]| \geq \varepsilon' E[X]] \leq \frac{V[X]}{\varepsilon'^2 E[X]^2} \leq \frac{1}{\varepsilon'^2 E[X]} = \frac{d}{3d_v} \leq \frac{1}{3}$$

In each of the $\log\frac{n}{\delta}$ groups we then estimate d_v as $\frac{d\varepsilon'^2 X}{3}$, which is with probability at least 2/3 $(1 \pm \varepsilon')$-approximation of d_v. Now we return the median of the estimates, $\tilde{d_v}$, in the $\log\frac{n}{\delta}$ groups. The colorings in the groups are independent and a standard application of Chernoff's inequality yields that the median will not be an $(1 \pm \varepsilon')$-approximation of d_v with error probability δ/n. Summing up, the error probability for any vertex is bounded by δ.

An estimate of the number of triangles at v is now $\binom{\tilde{d_v}}{2}\tilde{\alpha}_v$, where $\tilde{\alpha}_v$ is an $(1 \pm \varepsilon')$-approximation of the clustering coefficient α_v. With some algebra one can obtain that this yields an $(1 \pm 7\varepsilon')$-approximation of the number of triangles at v.

The bounds on the running time and the space complexity of the algorithm easily follow from Theorem 2 and the above discussion. □

For the case of directed graphs the algorithm can be extended to distinguish between the four possible kinds of triangles [6] a high-degree vertex can be involved in. The only difference is that we will count only the kind of triangles we are interested in. We defer the description of the extension to the full version of the paper.

5.4 Experiments

We implemented our algorithm in Java and ran experiments on a Windows machine with an Intel Core i5 with 2.66 GHz clocked processor with 3 MB Cache. The available RAM memory was 4 GB. Due to the relatively small amount of memory, we ran the copies of SparsifyGraph sequentially, thus we will not report results on the running time. It is clear that this modification does not affect in any way the estimates obtained by our algorithm. For our coloring function we created random numbers $r(v)$, $v \in V$, in the interval $[0, 1)$ by reading for each vertex 64 random bits from the Marsaglia Random Number CDROM[2]. Then for a given number of colors C two vertices u, v have the same color iff $\lfloor C \cdot r(u) \rfloor = \lfloor C \cdot r(v) \rfloor$. When the graph size is not known in advance, one can dynamically adjust the sampling probability as follows. Once we have that too many edges have been sampled, we double the number of colors, $C = 2C$, and throw away edges that are not any more monochromatic.

We performed experiments on several graphs and chose to present results for two representative graphs, which we think illustrate best the advantages and limitations of the algorithm. Also, we present only experiments on the estimation of the local clustering coefficient, since the estimates on the local number of triangles yield identical observations.

The first graph, Web-BerkStan, is taken from the SNAP library [3]. It is a directed Web graph in which an edge (u, v) shows that there is a link from a page u in the domain berkeley.edu to a page v in the domain stanford.edu. We made the graph undirected such that there is an edge (u, v) for pages u and v connected by a link in either direction and removed loops. The resulting graph consists of 680,485 vertices and 15,190,579 edges.

Following [10] we created a graph MovieActors1000 from the Internet Movie Database[4]. From a set of 1,000 movies, with at least 10 actors starring, we created an undirected graph such that an edge (u, v) records that the actors u in v star together in at least one movie. The resulting graph has 19,037 vertices and 6,501,181 edges.

Clearly, the first graph Web-BerkStan is very sparse, average degree of 22.323, while MovieAvtors1000 is a relatively dense graph with an average degree of 341.502. Also, the degree distribution in Web-BerkStan is quite skewed, the largest degree being 84,290 while the largest degree in MovieActors1000 is only 2,542.

The number of edges in the two graphs is of the same order, thus we compared the results for a fixed sparsification rate of 0.001 and a varying number of parallel copies of SparsifyGraph. For the two graphs we computed exactly the clustering coefficient for vertices of degree at least 1,000. Note that we chose a smaller sampling probability than the one given by Theorem 2 such that estimates for vertices of degree at least 1,000 will be reported with high probability. The reason is that allows a clearer overview of (dis)advantages of the algorithm for the two considered graphs. In Web-BerkStan there are 568 vertices of degree at least 1,000, while in MovieActors1000 there are 1,161 such vertices. The average clustering coefficient for the considered high-degree vertices for Web-BerkStan is 0.030141, while for MovieActors1000 it is 0.466.

[2]http://www.stat.fsu.edu/pub/diehard/cdrom/
[3]http://snap.stanford.edu/data/
[4]http://www.imdb.com/

Let H be the set of vertices of degree at least 1,000 for which estimates were reported and let $h = |H|$. Let further C_v be the exact clustering coefficients for vertices $v \in H$ and $\tilde{C}_v$ be the corresponding approximation we obtain. Following [6] we evaluated our algorithm with respect to the following measures:

1. Average relative error.
$$\sum_{v \in H} \frac{|C_v - \tilde{C}_v|}{C_v}$$
2. Pearson correlation coefficient.
$$r = \frac{h \sum\limits_{v \in H} C_v \tilde{C}_v - \sum\limits_{v \in H} C_v \sum\limits_{v \in H} \tilde{C}_v}{\sqrt{(h \sum\limits_{v \in H} C_v^2 - (\sum\limits_{v \in H} C_v)^2)(h \sum\limits_{v \in H} \tilde{C}_v^2 - (\sum\limits_{v \in H} \tilde{C}_v)^2)}}$$
3. Spearman's rank correlation.
Let *Exact* and *Approx* be two sets containing the exact and approximated clustering coefficients. Sorting *Exact* and *Approx* in decreasing order, for each vertex $v \in H$ we define $\text{dist}(v) = r_{Exact}(C_v) - r_{Approx}(\tilde{C}_v)$, where $r_S(x)$ is the rank of the element $x \in S$ in the sorted sequence S. Then the Spearman's rank correlation is defined as $\rho = 1 - \frac{6 \sum_{v \in H} \text{dist}(v)^2}{h(h^2 - 1)}$

For a number of parallel copies of the algorithm, varying between 50 and 400, we evaluated the quality of the achieved estimates. Note that this means that the sparsification factor is between 5% and 40%. We report estimates on the clustering coefficient of vertices $v \in V$ for which in at least half of the copies a 2-path centered at v was recorded.

Our first observation is that the larger skew in degree distributions in Web-BerkStan results in a higher recall in reported results for vertices of degree 1,000 or more. For Web-BerkStan for all number of parallel copies estimates for more than 80% of the high degree vertices were reported while for MovieActors1000 the recall never exceeded 50%.

Not surprisingly, the larger the number of samples the better approximation is achieved. Figures 3 and 4 show the achieved approximation of clustering coefficient of reported high-degree vertices for the two graphs for a sparsification ratio of 20%. The approximation is tighter for the MovieActors1000 graph while for Web-BerkStan it is more dispersed but this is due to the larger clustering coefficients in the former graph. This is also confirmed by the plot in Figure 5 where we see that the average relative error is much smaller for MovieActors1000. However, the correlation of the estimates of the clustering coefficient for high degree vertices and its true value is comparable for the two graphs as can be seen in Figures 6 and 7.

The above evaluation, as well as other experiments we performed but not report here, lead to the following observations:

- For graphs with a very skewed degree distribution we can report good estimates only for a fraction of the vertices if we want to achieve notable space savings. This is in accordance with the result in Theorem 1 and we believe that this is the best one can hope for. On the other hand the high degree vertices are correctly identified.

- For graphs with a lighter skew in the degree distribution however, our algorithm is able to yield good results for a reasonably big proportion of the vertices. A drawback is that it becomes more difficult to detect all high degree vertices.
- Even in cases when the estimates do not approximate very good the exact values, as in Web-BerkStan, there is a clear correlation between estimates and exact values.

It is worth noting that we did not perform experiments on the large Web-scale graphs considered in [6]. For the full version of the paper we are planning to present a more thorough experimental evaluation on datasets from different domains.

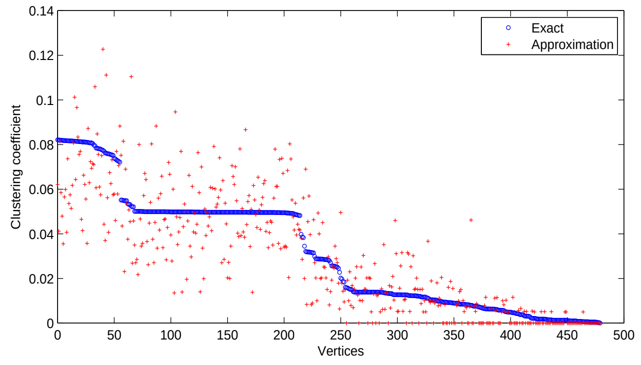

Figure 3: Exact and approximated clustering coefficients of high degree vertices for Web-BerkStan.

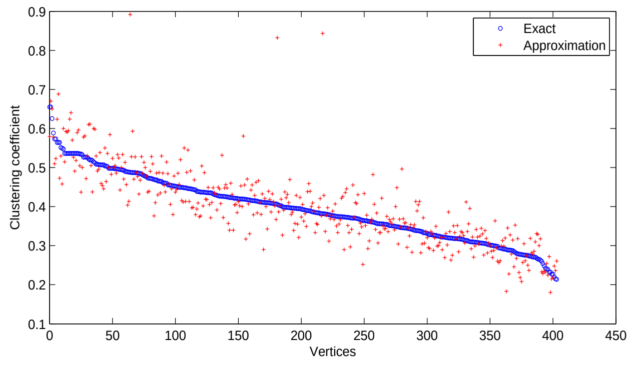

Figure 4: Exact and approximated clustering coefficients of high degree vertices for MovieActors1000.

6. FUTURE DIRECTIONS

We have presented results on the streaming complexity of randomized algorithms estimating the local clustering coefficient. The upper and lower bound on the space usage almost match. We believe that our algorithm is almost optimal in terms of space requirements and an open question is to obtain a tight bound on the space complexity of the problem. Also, it would be interesting whether a similar lower bound is possible for the incidence list stream model.

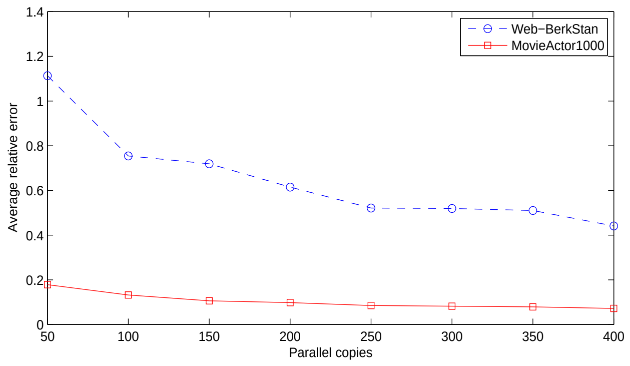

Figure 5: Average relative error for a varying number of parallel copies for the two graphs.

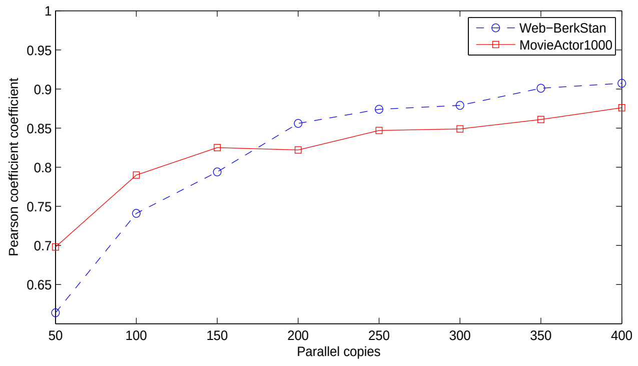

Figure 6: Pearson correlation coefficient for a varying number of parallel copies for the two graphs.

We expect that certain heuristics can improve the running time performance of our algorithm. For example, for many vertices we sample more than two neighbors and we can check for more 2-paths whether they are involved in a triangle. However, one should not consider 2-paths sharing an edge since this will increase the variance of the estimates and the bounds in Theorem 2 will not hold any more.

Also, the advantages of monochromatic sampling compared to naïve sampling of edges become more pronounced if one wants to count the number of k-cliques or more generally dense subgraphs of fixed size, both locally and globally, for $k > 3$. For example, inspired by concrete applications in Web mining, Bordino et al. [7] present an extension of the approach by Buriol et al. [9] to counting graph minors of fixed size. Many of the graph minors are dense, thus it is interesting to study whether using monochromatic sampling one can design better algorithms.

An interesting direction is to study whether one can combine our approach with ideas from L_p sampling in order to obtain an algorithm for the more general problem of processing dynamic graph streams, where edge deletions are also allowed, see e.g. [1, 17].

Acknowledgements.

We thank Paolo Boldi for useful discussions in the early stage of this work.

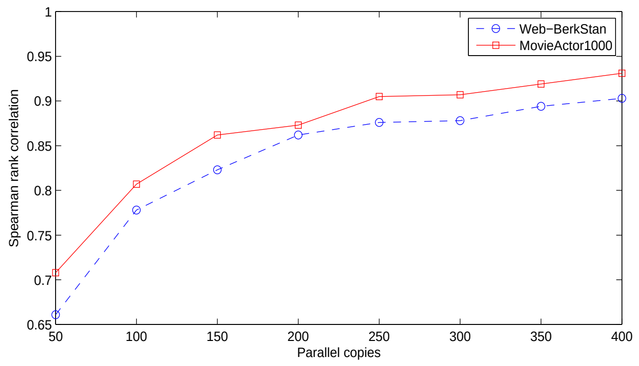

Figure 7: Spearman rank correlation for a varying number of parallel copies for the two graphs.

7. REFERENCES

[1] K. J. Ahn, S. Guha, A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. *PODS 2012*: 5–14

[2] R. Albert, A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys. 74*, 47–97 (2002)

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

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

[5] Z. Bar-Yossef, R. Kumar, D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. *SODA 2002*: 623–632

[6] L. Becchetti, P. Boldi, C. Castillo, A. Gionis. Efficient algorithms for large-scale local triangle counting. *TKDD 4(3)*: (2010)

[7] I. Bordino, D. Donato, A. Gionis, S. Leonardi. Mining Large Networks with Subgraph Counting. *ICDM 2008*: 737–742

[8] A. Z. Broder, M. Charikar, A. M. Frieze, M. Mitzenmacher. Min-Wise Independent Permutations. *STOC 1998*: 327–336

[9] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, C. Sohler. Counting triangles in data streams. *PODS 2006*: 253–262

[10] L. S. Buriol, G. Frahling, S. Leonardi, C. Sohler. Estimating Clustering Indexes in Data Streams. *ESA 2007*: 618–632

[11] J. S. Coleman. Social capital in the creation of human capital. *American Journal of Sociology, 94*: 95–120, 1988

[12] D. Coppersmith, R. Kumar. An improved data stream algorithm for frequency moments. *SODA 2004*: 151–156

[13] J.-P. Eckmann and E. Moses. Curvature of co-links uncovers hidden thematic layers in the world wide web. *PNAS, 99(9)*:5825–5829, 2002.

[14] P. Erdös, and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci. 5* 17–61, 1960

[15] G. Feigenblat, E. Porat, A. Shiftan. Exponential time improvement for min-wise based algorithms. *Inf. Comput. 209(4)*: 737–747 (2011)

[16] H. Jowhari, M. Ghodsi. New Streaming Algorithms for Counting Triangles in Graphs. *COCOON 2005*: 710–716

[17] H. Jowhari, M. Saglam, G. Tardos. Tight bounds for L_p samplers, finding duplicates in streams, and related problems. *PODS 2011*: 49–58

[18] M. Kaiser. Mean clustering coefficients: the role of isolated nodes and leafs on clustering measures for small-world networks. *New J. Phys. 10*, 2008.

[19] B. Kalyanasundaram, G. Schnitger. The Probabilistic Communication Complexity of Set Intersection. *SIAM J. Discrete Math. 5(4)*: 545–557 (1992)

[20] R. M. Karp, S. Shenker, C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst. 28*: 51–55 (2003)

[21] M. N. Kolountzakis, G. L. Miller, R. Peng, C. E. Tsourakakis. Efficient Triangle Counting in Large Graphs via Degree-based Vertex Partitioning. *Internet Mathematics*, to appear

[22] R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins. Trawling the Web for Emerging Cyber-Communities. *Computer Networks* 31(11-16): 1481–1493 (1999)

[23] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003

[24] R. Pagh, C. E. Tsourakakis. Colorful triangle counting and a MapReduce implementation. *Inf. Process. Lett. 112(7)*: 277–281 (2012)

[25] A. Portes. Social capital: Its origins and applications in modern sociology. *Annual Review of Sociology, 24(1)*:1–24, 1998.

[26] T. Schank, D. Wagner. Approximating Clustering Coefficient and Transitivity. *J. Graph Algorithms Appl. 9(2)*: 265–275 (2005)

[27] C. E. Tsourakakis. Fast Counting of Triangles in Large Real Networks without Counting: Algorithms and Laws. *ICDM 2008*: 608–617

[28] C. E. Tsourakakis, U. Kang, G. L. Miller, C. Faloutsos. DOULION: counting triangles in massive graphs with a coin. *KDD 2009*: 837–846

[29] C. E. Tsourakakis, M. N. Kolountzakis, G. L. Miller. Triangle Sparsifiers. *J. of Graph Algorithms and Appl. 15(6)*: 703–726 (2011)

[30] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. *STOC 2012*, 887–898

[31] J. S. Vitter. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.* 11(1): 37–57 (1985)

[32] D. J. Watts, S. H. Strogatz. Collective dynamics of "small world" networks. *Nature, 393:* 440–442, 1998.

[33] S. Zhang. Streaming Algorithms Measured in Terms of the Computed Quantity. *COCOON 2007*: 338–348