

Wedge Sampling for Computing Clustering Coefficients and Triangle Counts on Large Graphs ^{*}

C. Seshadhri[†] Ali Pinar[‡] Tamara G. Kolda[§]

Abstract

Graphs are used to model interactions in a variety of contexts, and there is a growing need to quickly assess the structure of such graphs. Some of the most useful graph metrics are based on *triangles*, such as those measuring social cohesion. Algorithms to compute them can be extremely expensive, even for moderately-sized graphs with only millions of edges. Previous work has considered node and edge sampling; in contrast, we consider *wedge sampling*, which provides faster and more accurate approximations than competing techniques. Additionally, wedge sampling enables estimation local clustering coefficients, degree-wise clustering coefficients, uniform triangle sampling, and directed triangle counts. Our methods come with provable and practical probabilistic error estimates for all computations. We provide extensive results that show our methods are both more accurate and faster than state-of-the-art alternatives.

1 Introduction

Graphs are used to model infrastructure networks, the World Wide Web, computer traffic, molecular interactions, ecological systems, epidemics, citations, and social interactions, among others. Despite the differences in the motivating applications, some topological structures have emerged to be important across all these domains. In particular, the *triangle* is a manifestation of homophily (people become friends with those similar to themselves) and transitivity (friends of friends become friends). The triangle structure of a graph is commonly used in the social sciences for positing various theses on behavior [11, 23, 7, 14]. Triangles have also been used in graph mining applications such as spam detection and finding common topics on the WWW [13, 4]. A new generative model, Blocked Two-Level Erdős-Rényi (BTER) [26], can capture triangle behavior in real-world graphs, but necessarily requires the degree-wise clustering coefficients as input. Relationships among degrees of triangle vertices can also be used as a descriptor of the underlying graph [12].

In this paper, we study the idea of *wedge sampling*, i.e., choosing random wedges (from a uniform distribution over all wedges) to compute various triadic measures on large-scale graphs. We provide precise probabilistic error bounds where the accuracy depends on the number of random samples. The term *wedge* refers to a path of length 2; in Fig. 1, ②–①–③ is a wedge *centered* at node 1. A *triangle* is a cycle of length 3; in Fig. 1, ③–④–⑤ is a triangle. We say a wedge is *closed* if its vertices form a triangle. Observe that each triangle consists of three closed wedges.

^{*}This manuscript is an extended version of [28]. This work was funded by the DARPA GRAPHS program and by the DOE ASCR Complex Distributed Interconnected Systems (CDIS) program, and Sandia’s Laboratory Directed Research & Development (LDRD) program. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.

[†]Sandia National Laboratories, CA, scomand@sandia.gov

[‡]Sandia National Laboratories, CA, apinar@sandia.gov

[§]Sandia National Laboratories, CA, tgkolda@sandia.gov

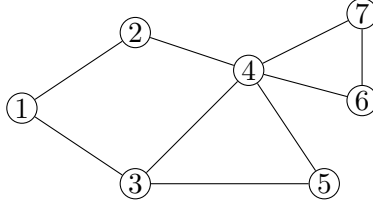


Figure 1: Example graph with $n = 7$ vertices, $m = 8$ edges, $W = 18$ wedges, and $T = 2$ triangles.

1.1 Triadic measures on graphs Let n , m , W , and T denote the number of nodes, edges, wedges, and triangles, respectively, in a graph G . A standard algorithmic task is to count (or approximate) T . There are many other “triadic” measures associated with graphs as well. The following are classic quantities of interest, and are defined on *undirected graphs*. (Formal definitions are given in [Tab. 1](#).)

- **Transitivity or global clustering coefficient** [34]: This is $\kappa = 3T/W$, and is the fraction of wedges that are closed (i.e., participate in a triangle). Intuitively, it measures how often friends of friends are also friends, and it is the most common triadic measure.
- **Local clustering coefficient** [35]: The clustering coefficient of vertex v (denoted by C_v) is the fraction of wedges centered at v that are closed. The average of C_v over all vertices v is the *local clustering coefficient*, denoted by C .
- **Degree-wise clustering coefficients**: We let C_d denote the average clustering coefficient of degree- d vertices. In addition to degree distribution, many graphs are characterized by plotting the degree-wise clustering coefficients, i.e., C_d , versus d .

In [Fig. 1](#), the transitivity is $\kappa = 3T/W = 3 \cdot 2/18 = 1/3$. The clustering coefficients of the vertices are 0, 0, $1/3$, $1/5$, 1, 1, and 1 in order, and the local clustering coefficient is 0.5048. In general, the most direct method to compute these metrics is an exact computation that finds all triangles. For degree-wise clustering coefficients, no other method was previously known.

Triangles and transitivity in directed graphs have also been the subject of recent work (see e.g. [27] and references therein). In a directed graph, edges are ordered pairs of vertices of the form (i, j) , indicating a link from node i to node j . When edges (i, j) and (j, i) exist, we say there is a *reciprocal edge* between i and j . If just one edge exists, then we say it is a *one-way edge*. Considering all possible combinations of reciprocal and one-way edges leads to 6 different wedges and 7 different triangles.

1.2 Our contributions In this paper, we discuss the simple yet powerful technique of *wedge sampling* for counting triangles. Wedge sampling is really an algorithmic template, since various algorithms can be obtained from the same basic idea. It can be used to estimate all the different triadic measures detailed above. The method also provides precise probabilistic error bounds where the accuracy depends on the number of random samples. The mathematical analysis of this method is a direct consequence of standard Chernoff-Hoeffding bounds. If we want an estimate for κ with error $\varepsilon \leq 0.1$ with probability 99.9% (say), we need only 380 wedge samples. This estimate is *independent of the size of the graph*, though the preprocessing required by our method is linear in the number of edges (to obtain the degree distribution).

We discovered older independent work by Schank and Wagner that proposes the same wedge sampling idea for estimating the global and local clustering coefficients [24]. Our work extends that in several directions, including several other uses for wedge sampling (such as directed triangle

counting, random triangle sampling, degree-wise clustering coefficients) and much more extensive numerical results. This manuscript is an extended version of [28]; here we give more detailed experimental results and also show how wedge sampling applies to computing directed triangle counts. A detailed list of contributions follows.

- **Computing the transitivity (global clustering coefficient) and local clustering coefficient:** We show how to use wedge sampling to approximate the global and local clustering coefficients: κ and C . We compare these methods to the state of the art, showing significant improvement for large-scale graphs.

- **Computing degree-wise clustering coefficients:** The idea of wedge sampling can be extended for more detailed measurements of graphs. We show how to calculate the degree-wise clustering coefficients, $\{C_d\}$ for $d = 1, \dots, d_{\max}$. The only other competing method that can compute $\{C_d\}$ is an exhaustive enumeration. We compare with the basic fast enumeration algorithm given by [8, 25] (which has been studied and reinvented by [10, 29]).

- **Computing triangles per degree:** Wedge sampling can also be employed to sample random triangles, including the application of estimating the number of triangles containing one (or more) vertices of degree d , denoted $\{T_d\}$ for $d = 1, \dots, d_{\max}$.

- **Counting directed triangles:** Few methods have been considered for counting triangles in directed graphs. It is especially complicated because there are 7 types of triangles to consider. Once again, the versatility of wedge sampling is used to develop a method for counting all types of directed triangles.

1.3 Related Work There has been significant work on enumeration of all triangles [8, 25, 20, 5, 9]. Recent work by Cohen [10] and Suri and Vassilvitskii [29] give MapReduce implementations of these algorithms. Arifuzzaman et al. [1] give a massively parallel algorithm for computing clustering coefficients. Enumeration algorithms however, can be very expensive, since graphs even of moderate size (millions of vertices) can have an extremely large number of triangles (see, e.g., Tab. 2). Eigenvalue/trace based methods have been used by Tsourakakis [32] and Avron [2] to compute estimates of the total and per-degree number of triangles. However, computing eigenvalues (even just a few of them) is a compute-intensive task and quickly becomes intractable on large graphs. In our experiment, even computing the largest eigenvalue was multiple orders of magnitude slower than full enumeration.

Most relevant to our work are sampling mechanisms. Tsourakakis et al. [30] started the use of sparsification methods, the most important of which is Doulion [33]. This method sparsifies the graph by keeping each edge with probability p ; counts the triangles in the sparsified graph; and multiplies this count by p^{-3} to predict the number of triangles in the original graph. Various theoretical analyses of this algorithm (and its variants) have been proposed [19, 31, 22]. One of the main benefits of Doulion is that it reduces large graphs to smaller ones that can be loaded into memory. However, the Doulion estimate can suffer from high variance [36]. Alternative sampling mechanisms have been proposed for streaming and semi-streaming algorithms [3, 17, 4, 6]. Yet, all these fast sampling methods only estimate the number of triangles and give no information about other triadic measures.

In subsequent work by the authors of this paper, a Hadoop implementation of these techniques is given in [18], and a streaming version of the wedge sampling method is presented in [16].

Table 1: Graph notation and clustering coefficients for undirected graphs

n	number of vertices	n_d	number of vertices of degree d
m	number of edges	d_v	degree of vertex v
V_d	set of degree- d vertices	W	total number of wedges
W_v	number of wedges centered at vertex v	T	total number of triangles
T_v	number of triangles incident to vertex v	T_d	number of triangles incident to degree- d vertices
<hr/>			
$\kappa = 3T/W$	transitivity		
$C_v = T_v/W_v$	clustering coefficient of vertex v		
$C = n^{-1} \sum_v C_v$	local clustering coefficient		
$C_d = n_d^{-1} \sum_{v \in V_d} C_v$	degree-wise clustering coefficient		

2 Overview of wedge sampling

We present the general method of wedge sampling for estimating clustering coefficients. In later sections, we instantiate this for different algorithms. Before we begin, we summarize the notation presented in §1 in Tab. 1.

We say a wedge is *closed* if it is part of a triangle; otherwise, the wedge is *open*. Thus, in Fig. 1, ⑤-④-⑥ is an open wedge, while ③-④-⑤ is a closed wedge. The middle vertex of a wedge is called its *center*, i.e., wedges ⑤-④-⑥ and ③-④-⑤ are centered at ④.

Wedge sampling is best understood in terms of the following thought experiment. Fix some distribution over wedges and let w be a random wedge. Let X be the indicator random variable that is 1 if w is closed and 0 otherwise. Denote $\mu = \mathbf{E}[X]$.

Suppose we wish to estimate μ . We simply generate k independent random wedges w_1, w_2, \dots, w_k , with associated random variables X_1, X_2, \dots, X_k . Define $\bar{X} = \frac{1}{k} \sum_{i \leq k} X_i$ as our estimate. The Chernoff-Hoeffding bounds give guarantees on \bar{X} , as follows.

THEOREM 2.1. (HOEFFDING [15]) *Let X_1, X_2, \dots, X_k be independent random variables with $0 \leq X_i \leq 1$ for all $i = 1, \dots, k$. Define $\bar{X} = \frac{1}{k} \sum_{i=1}^k X_i$. Let $\mu = \mathbf{E}[\bar{X}]$. Then for $\varepsilon \in (0, 1)$, we have*

$$\Pr \{ |\bar{X} - \mu| \geq \varepsilon \} \leq 2 \exp(-2k\varepsilon^2).$$

Hence, if we set $k = \lceil 0.5\varepsilon^{-2} \ln(2/\delta) \rceil$, then $\Pr[|\bar{X} - \mu| > \varepsilon] < \delta$. In other words, for k samples, with confidence at least $1 - \delta$, the error in our estimate is at most ε .

Fig. 2 shows the number of samples needed as the error rate varies. We show three curves for different confidence levels. Increasing the confidence has minimal impact on the number of samples. The number of samples is fairly low for error rates of 0.1 or 0.01, but it increases with the inverse square of ε .

3 Computing the transitivity and the number of triangles

We use the wedge sampling scheme to estimate the transitivity, κ . Consider the uniform distribution on wedges. We can interpret $\mathbf{E}[X]$ as the probability that a uniform random wedge is closed or, alternately, the fraction of closed wedges.

To generate a uniform random wedge, note that the number of wedges centered at vertex v is $W_v = \binom{d_v}{2}$ and $W = \sum_v W_v$. We set $p_v = W_v/W$ to get a distribution over the vertices. Note that the probability of picking v is proportional to the number of wedges centered at v . A uniform

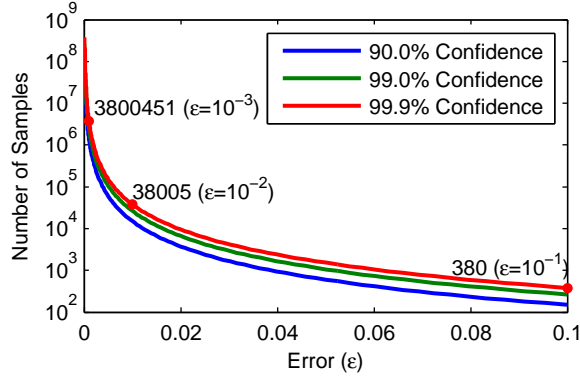


Figure 2: The number of samples needed for different error rates and different levels of confidence. A few data points at 99.9% confidence are highlighted.

random wedge centered at v can be generated by choosing two random neighbors of v (without replacement).

CLAIM 3.1. *Suppose we choose vertex v with probability p_v and take a uniform random pair of neighbors of v . This generates a uniform random wedge.*

Proof. Consider some wedge w centered at vertex v . The probability that v is selected is $p_v = W_v/W$. The random pair has probability of $1/\binom{d_v}{2} = 1/W_v$. Hence, the net probability of w is $1/W$. \square

Alg. 1 shows the randomized algorithm κ -wedge sampler for estimating κ in a graph G . Observe that the first step assumes that the degree of each vertex is already computed. Sampling with replacements means that we sample from the original distribution repeatedly and repeats are allowed. Sampling without replacement means that we cannot pick the same item twice.

Algorithm 1 Computing the transitivity (κ -wedge sampler)

- 1: Select k random vertices (with replacement) according to the probability distribution defined by $\{p_v\}$ where $p_v = W_v/W$.
 - 2: For each selected vertex v , choose two neighbors of v (uniformly at random without replacement) to generate a random wedge. The set of all wedges comprises the sample set.
 - 3: Output the fraction of closed wedges in the sample set as estimate of κ .
-

Combining the bound of Thm. 2.1 with Claim 3.1, we get the following theorem. Note that the number of samples required is *independent of the graph size*, but computing p_v does depend on the number of edges, m .

THEOREM 3.1. *Set $k = \lceil 0.5\epsilon^{-2} \ln(2/\delta) \rceil$. The algorithm κ -wedge sampler outputs an estimate \bar{X} for the transitivity κ such that $|\bar{X} - \kappa| < \epsilon$ with probability greater than $(1 - \delta)$.*

To get an estimate on T , the number of triangles, we output $\bar{X} \cdot W/3$, which is guaranteed to be within $\pm\epsilon W/3$ of T with probability greater than $1 - \delta$.

We implemented our algorithms in C and ran our experiments on a computer equipped with a 2.3GHz Intel core i7 processor with 4 cores and 256KB L2 cache (per core), 8MB L3 cache, and

Table 2: Properties of the graphs and runtimes for enumeration.

Graph Name	Nodes n	Edges m	Wedges W	Triangles T	Transitivity κ	Local CC C	Enum. Time (secs)
amazon0312	401K	2350K	69M	3686K	0.160	0.421	0.261
amazon0505	410K	2439K	73M	3951K	0.162	0.427	0.269
amazon0601	403K	2443K	72M	3987K	0.166	0.430	0.268
as-skitter	1696K	11095K	16022M	28770K	0.005	0.296	90.897
cit-Patents	3775K	16519K	336M	7515K	0.067	0.092	3.764
roadNet-CA	1965K	2767K	6M	121K	0.060	0.055	0.095
web-BerkStan	685K	6649K	27983M	64691K	0.007	0.634	54.777
web-Google	876K	4322K	727M	13392K	0.055	0.624	0.894
web-Stanford	282K	1993K	3944M	11329K	0.009	0.629	6.987
wiki-Talk	2394K	4660K	12594M	9204K	0.002	0.201	20.572
youtube	1158K	2990K	1474M	3057K	0.006	0.128	2.740
flickr	1861K	15555K	14670M	548659K	0.112	0.375	567.160
livejournal	5284K	48710K	7519M	310877K	0.124	0.345	102.142

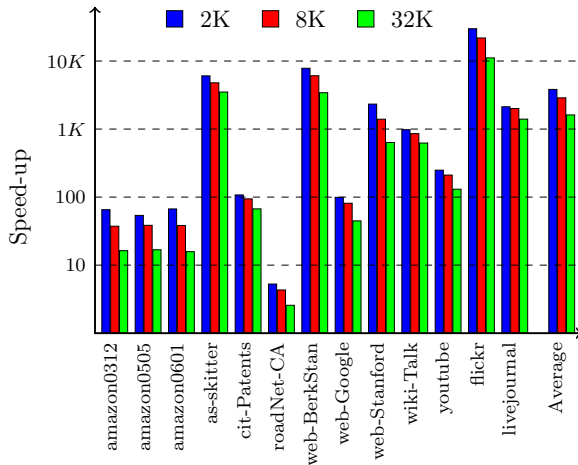


Figure 3: Speed-up over enumeration (in log-scale) for transitivity computation with increasing numbers of wedge samples.

an 8GB memory. We performed our experiments on 13 graphs from SNAP [37] and per private communication with the authors of [21]. In all cases, directionality is ignored, and repeated and self-edges are omitted. The properties of these matrices are presented in Tab. 2. The last column reports the times for the enumeration algorithm. This is based on the principles of [8, 25, 10, 29]: each edge is assigned to the vertex with a smaller degree (using the vertex numbering as a tie-breaker), and then vertices only check wedges formed by edges assigned to them for closure.

As seen in Fig. 3, wedge sampling is orders of magnitude faster than the enumeration algorithm. The timing results show tremendous savings; for instance, wedge sampling only takes 0.026 seconds on *as-skitter* while full enumeration takes 90 seconds.

Fig. 4 shows the accuracy of the wedge sampling algorithm. At 99.9% confidence ($\delta = 0.001$), the upper bound on the error we expect for 2K, 8K, and 32K samples is .043, .022, and .011, respectively. Most of the errors are much less than the bounds would suggest. For instance, the maximum error for 2K samples is .007, much less than that 0.43 given by the upper bound.

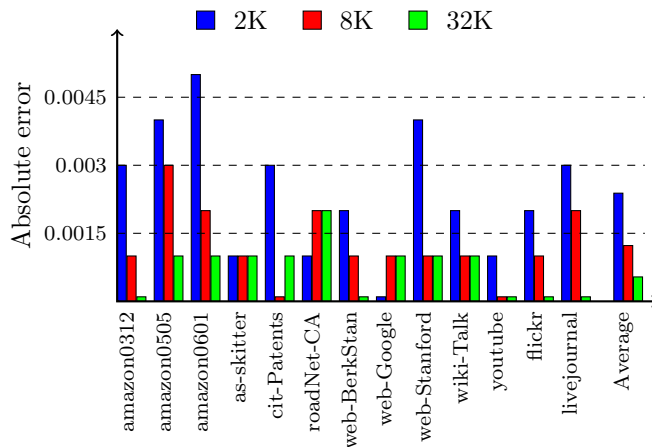


Figure 4: Absolute error in transitivity for increasing numbers of wedge samples.

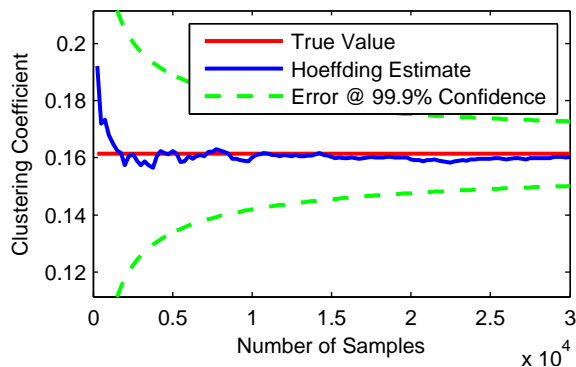


Figure 5: Convergence of transitivity estimate as the number of samples increases for `amazon0505`.

We show the rate of convergence in Fig. 5 for the graph `amazon0505` as the number of samples increases. The dashed line shows the error bars at 99.9% confidence. Note that the convergence is fast and the error bars are conservative in this instance.

4 Computing the local clustering coefficient

We now demonstrate how a small change to the underlying distribution on wedges allows us to compute the clustering coefficient, C . Alg. 2 shows the procedure `C-wedge sampler`. The only difference between `C-wedge sampler` and κ -`wedge sampler` is in picking random vertex centers. Vertices are picked uniformly instead of from the distribution $\{p_v\}$.

Algorithm 2 Computing the local clustering coefficient (`C-wedge sampler`)

- 1: Pick k uniform random vertices (with replacement).
 - 2: For each selected vertex v , choose two neighbors of v (uniformly at random without replacement) to generate a random wedge. The set of all wedges comprises the sample set.
 - 3: Output the fraction of closed wedges in the sample set as an estimate for C .
-

THEOREM 4.1. *Set $k = \lceil 0.5\varepsilon^{-2} \ln(2/\delta) \rceil$. The algorithm `C-wedge sampler` outputs an estimate \bar{X} for the clustering coefficient C such that $|\bar{X} - C| < \varepsilon$ with probability greater than $(1 - \delta)$.*

Proof. Let us consider a single sampled wedge w , and let $X(w)$ be the indicator random variable for the wedge being closed. Let \mathcal{V} be the uniform distribution on edges. For any vertex v , let \mathcal{N}_v be the uniform distribution on pairs of neighbors of v . Observe that

$$\mathbf{E}[X] = \Pr_{v \sim \mathcal{V}} \left[\Pr_{(u, u') \sim \mathcal{N}_v} [\text{wedge } \{(u, v), (u', v)\} \text{ is closed}] \right]$$

We will show that this is exactly C .

$$\begin{aligned} C &= n^{-1} \sum_v C_v = \mathbf{E}_{v \sim \mathcal{V}} [C_v] \\ &= \mathbf{E}_{v \sim \mathcal{V}} [\text{frac. of closed wedges centered at } v] \\ &= \mathbf{E}_{v \sim \mathcal{V}} \left[\Pr_{(u, u') \sim \mathcal{N}_v} [\{(u, v), (u', v)\} \text{ is closed}] \right] \\ &= \Pr_{v \sim \mathcal{V}} \left[\Pr_{(u, u') \sim \mathcal{N}_v} [\{(u, v), (u', v)\} \text{ is closed}] \right] = \mathbf{E}[X] \end{aligned}$$

For a single sample, the probability that the wedge is closed is exactly C . The bound of [Thm. 2.1](#) completes the proof. \square

[Fig. 6](#) presents the results of our experiments for computing the clustering coefficients. Experimental setup and the notation are the same as before. The results again show that wedge sampling provides accurate estimations with tremendous improvements in runtime. For instance, for 8K samples, the average speedup is 10K.

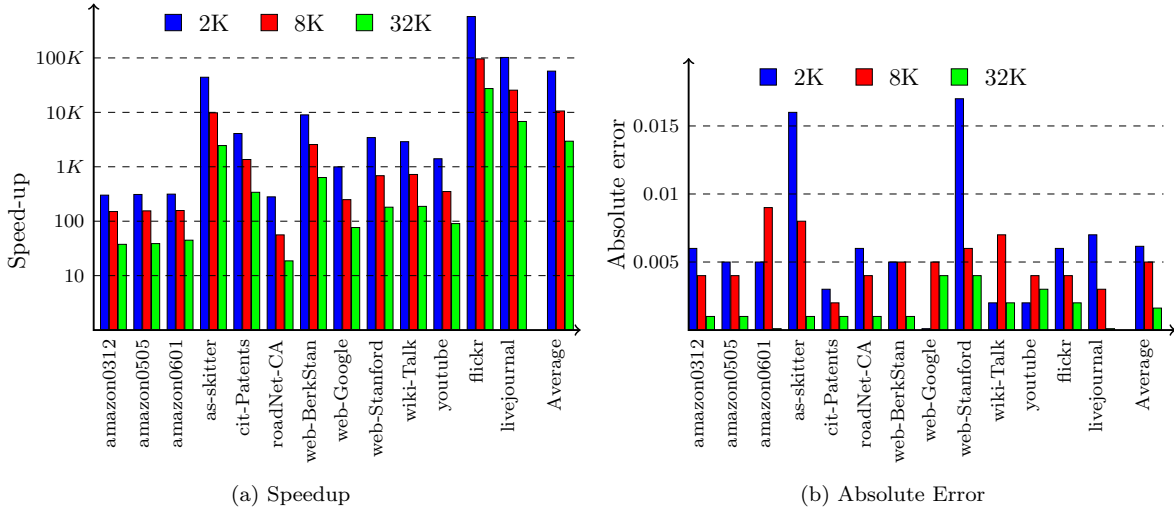


Figure 6: Local clustering coefficient results for increasing numbers of wedge samples.

5 Computing degree-wise clustering coefficients

The real power of wedge sampling is demonstrated by estimating the degree-wise clustering coefficients $\{C_d\}$. [Alg. 3](#) shows procedure `C_d -wedge sampler`.

THEOREM 5.1. *Set $k = \lceil 0.5\epsilon^{-2} \ln(2/\delta) \rceil$. The algorithm `C_d -wedge sampler` outputs an estimate \bar{X} for the clustering coefficient C_d such that $|\bar{X} - C_d| < \epsilon$ with probability greater than $(1 - \delta)$.*

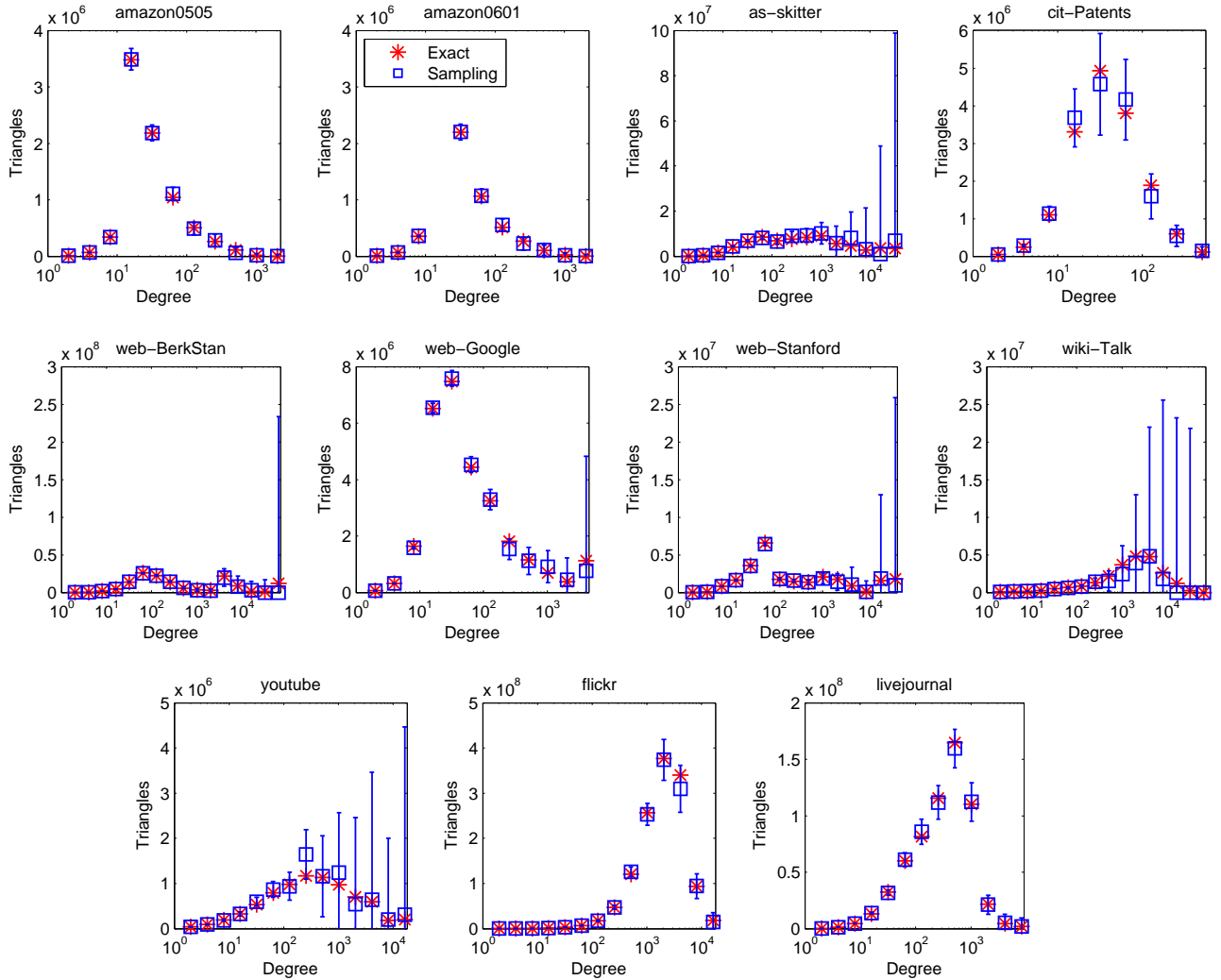


Figure 7: Computing degree-wise clustering coefficients using wedge sampling

Proof. The proof is analogous to that of [Thm. 4.1](#). Since C_d is the average clustering coefficient of a degree- d vertex, we can apply the same arguments as in [Thm. 4.1](#). \square

Algorithms in the previous section present how to compute the clustering coefficient of vertices of a given degree. In practice, it may be sufficient to compute clustering coefficients over bins of degrees. Wedge sampling algorithms can be performed for bins of degrees by a small adjustment of the sampling. Within each bin, we weight each vertex according to the number of wedges it produces. This guarantees that each wedge in the bin is equally likely to be selected. For instance, if we bin degree-3 and degree-4 vertices together, we will weight a degree-4 vertex twice as much as a degree-3-vertex, since a degree-3 vertex generates $\binom{3}{2} = 3$ wedges whereas a degree-4 vertex has $\binom{4}{2} = 6$. See [\[18\]](#) for details of binned computations.

[Fig. 7](#) shows results of [Alg. 3](#) on three graphs for clustering coefficients. In these experiments, we chose to group vertices in logarithmic bins of degrees, i.e., $\{2\}, \{3, 4\}, \{5, 6, 7, 8\}, \dots$. In other words, $2^{i-1} < d_v \leq 2^i$ form the i -th bin. The same algorithm can be used for an arbitrary

Algorithm 3 Computing clustering coefficients by degree (C_d -wedge sampler)

- 1: Pick k uniform random vertices of degree d (with replacement).
 - 2: For each selected vertex v , choose a uniform random pair of neighbors of v to generate a wedge. choose two neighbors of v (uniformly at random without replacement) to generate a random wedge. The set of all wedges comprises the sample set.
 - 3: Output the fraction of closed wedges in the sample set as an estimate for C_d .
-

binning of the vertices. We show the estimates with increasing number of samples. At 8K samples, the error is expected to be less than 0.02, which is apparent in the plots. Observe that even 500 samples yields a reasonable estimate in terms of the differences by degree.

Fig. 8 shows the time to calculate the binned C_d values compared to enumeration; there is a tremendous savings in runtime as a result of using wedge sampling. In this figure, runtimes are normalized with respect to the runtimes of full enumeration. As the figure shows, wedge sampling takes only a tiny fraction of the time of full enumeration, especially for large graphs.

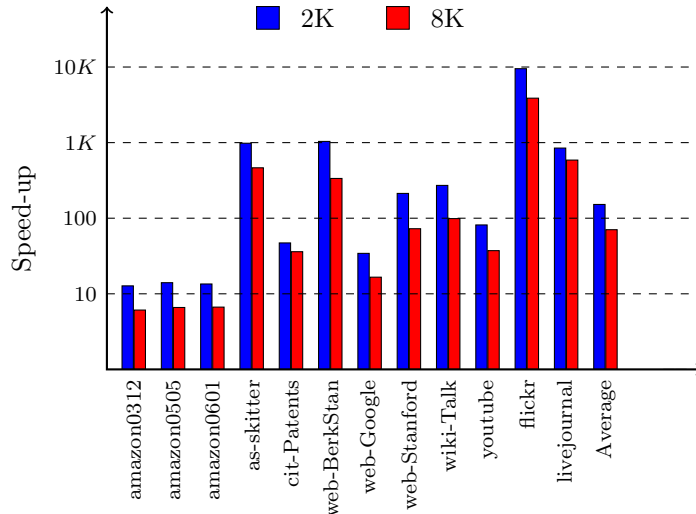


Figure 8: Speed-up in binned degree-wise clustering coefficient computation time for increasing numbers of wedge samples

6 Counting undirected triangles per degree

By modifying the template given in §2, we can also get estimates for T_d (the number of triangles incident to degree- d vertices). Instead of counting the fraction of closed wedges, we take a weighted sum. Alg. 4 describes the procedure T_d -wedge sampler. We let $W_d = n_d \cdot \binom{d}{2}$ denote the total number of wedges centered at degree- d vertices.

THEOREM 6.1. *Set $k = \lceil 0.5\varepsilon^{-2} \ln(2/\delta) \rceil$. The algorithm T_d -wedge sampler outputs an estimate $W_d \cdot \bar{Y}$ for the T_d with the following guarantee: $|W_d \cdot \bar{Y} - T_d| < \varepsilon W_d$ with probability greater than $1 - \delta$.*

Proof. For a single sampled wedge w_i , we define Y_i . We will argue below that the expected value of $\mathbf{E}[Y]$ is exactly T_d/W_d . Once we have that, an application of the Hoeffding bound of Thm. 2.1

Algorithm 4 Computing triangles by degree (T_d -wedge sampler)

- 1: Pick k uniform random vertices of degree d (with replacement).
- 2: For each selected vertex v , choose two neighbors of v (uniformly at random without replacement) to generate a random wedge. The set of all wedges comprises the sample set.
- 3: For each wedge w_i in the sample set, let Y_i be the associated random variable such that

$$Y_i = \begin{cases} 0 & \text{if } w \text{ is open,} \\ \frac{1}{3} & \text{if } w \text{ is closed and has 3 vertices in } V_d, \\ \frac{1}{2} & \text{if } w \text{ is closed and has 2 vertices in } V_d, \\ 1 & \text{if } w \text{ is closed and has 1 vertex in } V_d. \end{cases}$$

- 4: Set $\bar{Y} = \frac{1}{k} \sum_i Y_i$.
 - 5: Output $W_d \cdot \bar{Y}$ as the estimate for T_d .
-

shows that $|\bar{Y} - T_d/W_d| < \varepsilon$ with probability greater than $1 - \delta$. Multiplying this inequality by W_d , we get $|W_d \cdot \bar{Y} - T_d| < \varepsilon W_d$, completing the proof.

To show $\mathbf{E}[Y] = T_d/W_d$, partition the set of all wedges centered on degree d vertices into four sets S_0, S_1, S_2 , and S_3 . The set S_i ($i \neq 0$) contains all closed wedges containing exactly i degree- d vertices. The remaining open wedges go into S_0 . For a sampled wedge w , if $w \in S_i, i \neq 0$, then $Y_i = 1/i$. If $w \in S_0$, then $Y_i = 0$. The wedge w is a uniform random wedge from those centered on degree- d vertices. Hence, $\mathbf{E}[Y] = W_d^{-1}(|S_1| + |S_2|/2 + |S_3|/3)$.

Now partition the set of triangles involving degree d vertices into three sets S'_1, S'_2, S'_3 , where S'_i is the set of triangles with i degree d vertices. Observe that $|S_i| = i|S'_i|$. If a triangle has i vertices of degree d , then there are exactly i wedges centered in degree d vertices (in that triangle). So, $|S_1| + |S_2|/2 + |S_3|/3 = |S'_1| + |S'_2| + |S'_3| = T_d$. Therefore, $\mathbf{E}[Y] = T_d/W_d$. \square

The results of [Alg. 4](#) are shown in [Fig. 9](#). Once again, the data is grouped in logarithmic bins of degrees, i.e., $\{2\}, \{3, 4\}, \{5, 6, 7, 8\}, \dots$. In other words, $2^{i-1} < d_v \leq 2^i$ form the i -th bin. The number of samples is $k = 2000$. Here we also show the error bars corresponding to 99% confidence ($\delta = 0.01$).

7 Counting directed triangles

Counting triangles in directed graphs is more difficult, because there are six types of wedges and seven types of directed triangles (up to isomorphism); see [Fig. 10](#). As mentioned earlier, a reciprocal edge is a pair of one-way edges $\{(i, j), (j, i)\}$ that are treated as a *single* reciprocal edge. For each vertex v , we have three associated degrees: the indegree, outdegree, and reciprocal degree, denoted by d_v^{\leftarrow} , d_v^{\rightarrow} , and d_v^{\leftrightarrow} , respectively.

We can generalize wedge sampling to count the different triangles types in [Fig. 10b](#). This is done by randomly sampling the different wedges, and looking for various directed closures. We need some notation to formalize various directed triangle concepts, which is adapted from [\[27\]](#).

Naturally, the first step is to construct methods for sampling the different wedge types (see [Fig. 10a](#)). For wedge type ψ , let W_ψ denote the number of wedges of this type. For vertex v , let $W_{v,\psi}$ be the number of ψ -wedges centered at v . It is straightforward to compute $W_{v,\psi}$ given the degrees of v ; see the formulas in [Fig. 10c](#).

Given the number of ψ -wedges per vertex, we can use our standard template to sample a uniform

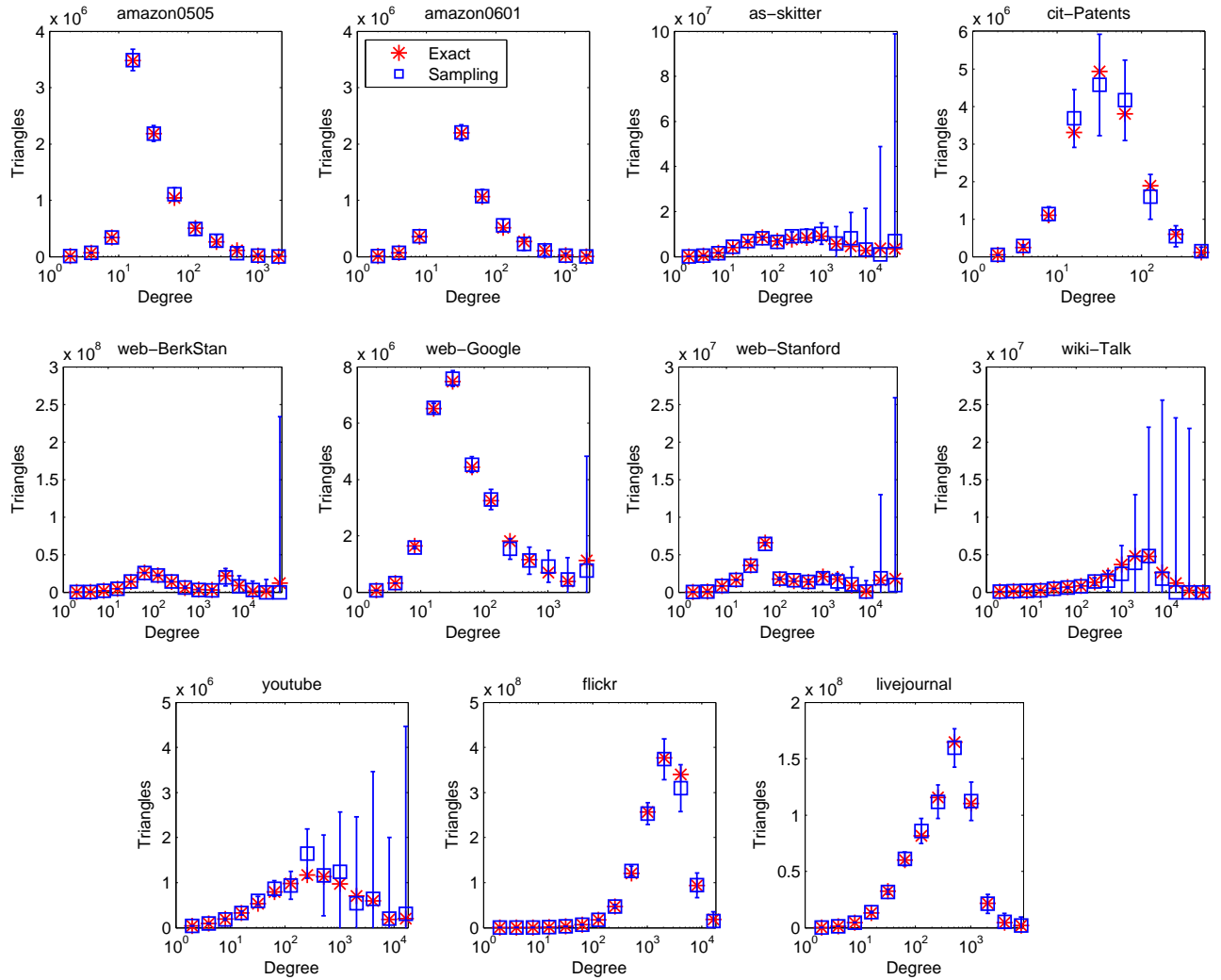
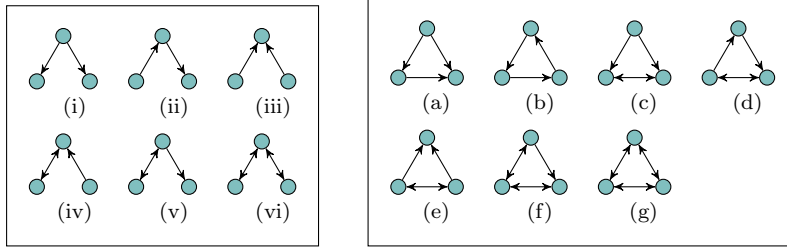


Figure 9: Number of triangles per degree bin. The red asterisk is the exact count, while the blue box is the estimate based on 2000 samples. The bars show the error at 99% confidence.



(a) Directed wedges

(b) Directed triangles

ψ	i	ii	iii	iv	v	vi
$W_{v,\psi}$	$\binom{d_v^{\rightarrow}}{2}$	$d_v^{\leftarrow} d_v^{\rightarrow}$	$\binom{d_v^{\leftarrow}}{2}$	$d_v^{\leftarrow} d_v^{\leftrightarrow}$	$d_v^{\rightarrow} d_v^{\leftrightarrow}$	$\binom{d_v^{\leftrightarrow}}{2}$

(c) Number of directed wedges for each wedge type

		Wedge type (ψ)					
		i	ii	iii	iv	v	vi
Triangle type (ρ)	a	1	1	1			
	b		3				
	c	1				2	
	d		1		1	1	
	e			1	2		
	f				1	1	1
	g						3

(d) The values of $\chi(\psi, \rho)$, indicating of many wedges of type ψ participate in triangles of type ρ .

Figure 10: Information for counting triangles in directed graphs.

random ψ -wedge. This is given in the first two steps of [Alg. 5](#). The procedure to select a uniform random ψ -wedge centered at v depends on the wedge type. For example, to sample a type-(i) wedge, we pick a uniform random pair of out-neighbors of v (without replacement). To sample a type-(ii) wedge, we pick a uniform random out-neighbor, and a uniform random in-neighbor. All other wedge types are sampled analogously.

To get the triangle counts, we need a directed transitivity that measures the fraction of closed directed wedges. Of course, a directed wedge can close into many different types of triangles. For example, an type-(i) wedge can close into a type-(a) or type-(c) triangle. To measure this meaningfully, a set of directed closure measures are defined in [\[27\]](#). We restate these definitions and show how to count triangles using these measures.

Let $\chi(\psi, \rho)$ denote the number of type- ψ wedges in a type- ρ triangle. For example, a type-(a) triangle contains exactly one type-(i) wedge, but a type-(b) triangle contains three type-(ii) wedges. The list of these values is given in [Fig. 10d](#).

We say that a ψ -wedge is ρ -closed if the wedge participates in a ρ -triangle. The number of such wedges is exactly $\chi(\psi, \tau)T_\rho$ (where T_ρ is the number of ρ -triangles). The (ψ, ρ) -closure, $\kappa_{\psi, \rho}$, is the fraction of ψ -wedges that are ρ -closed.

$$\kappa_{\psi, \rho} = \frac{\chi(\psi, \rho)T_\rho}{W_\psi}$$

There are 15 non-trivial (ψ, ρ) -closures, corresponding to the non-zero entries in [Fig. 10d](#). By the wedge sampling framework, we can estimate any $\kappa_{\psi, \rho}$ value. This value can be used to estimate T_ρ .

Algorithm 5 Computing directed closure ratios ($\kappa(\psi, \rho)$ -sampler)

- 1: Select k random vertices (with replacement) according to the probability distribution defined by $\{p_v\}$ where $p_v = W_{v, \psi}/W_\psi$.
 - 2: For each selected vertex v , pick a uniform random ψ -wedge centered at v . Note that if both edges are of the same type (i.e., two out-edges), then sampling is done without replacement. Otherwise, since the two edge-types are distinct and the sampling is independent.
 - 3: Determine k' , the number of ρ -closed wedges among the sampled wedges.
 - 4: Compute estimate $\hat{\kappa}_{\psi, \rho} = k'/k$ for $\kappa_{\psi, \rho}$.
 - 5: Output estimate $\hat{\kappa}_{\psi, \rho} \cdot W_\psi/\chi(\psi, \rho)$ for T_ρ .
-

THEOREM 7.1. Fix a triangle type ρ and wedge type ψ such that $\chi(\psi, \rho) \neq 0$. Set $k = \lceil 0.5\varepsilon^{-2} \ln(2/\delta) \rceil$. The algorithm $\kappa(\psi, \rho)$ -sampler outputs an estimate \hat{T}_ρ for T_ρ such that $|\hat{T}_\rho - T_\rho| < \varepsilon W_\psi/\chi(\psi, \rho)$ with probability greater than $(1 - \delta)$.

Proof. Using the Hoeffding bound, we can argue (as before) that $|\hat{\kappa}_{\psi, \rho} - \kappa_{\psi, \rho}| < \varepsilon$ with probability at least $(1 - \delta)$. We multiply this inequality by $W_\psi/\chi(\psi, \rho)$ and observe that $T_\rho = \kappa_{\psi, \rho} \cdot W_\psi/\chi(\psi, \rho)$.

The experimental setup is the same as in the previous sections. We performed our experiments on 8 directed graphs, described in [Tab. 3](#). The results of applying [Alg. 5](#) are shown in [Fig. 11](#). We use 32K wedges samples for each of four specific wedge types (detailed below). As can be seen, we get accurate results for all triangle types, except possibly type-(b) triangles. These are so few that they do not even appear in the plot.

Table 3: Properties of the directed graphs

Graph Name	Nodes	Dir. Edges	Wedges	Triangles	r	κ
amazon0505	410K	3357K	73M	3951K	0.55	0.162
web-Stanford	282K	2312K	3944M	11330K	0.28	0.009
web-BerkStan	685K	7601K	27983M	64691K	0.25	0.007
wiki-Talk	2394K	5021K	12594M	9204K	0.14	0.002
web-Google	876K	5105K	727M	13392K	0.31	0.055
youtube	1158K	4945K	1474M	3057K	0.79	0.006
flickr	1861K	22614K	14675M	548659K	0.62	0.112
livejournal	5284K	76938K	7519M	310877K	0.73	0.124

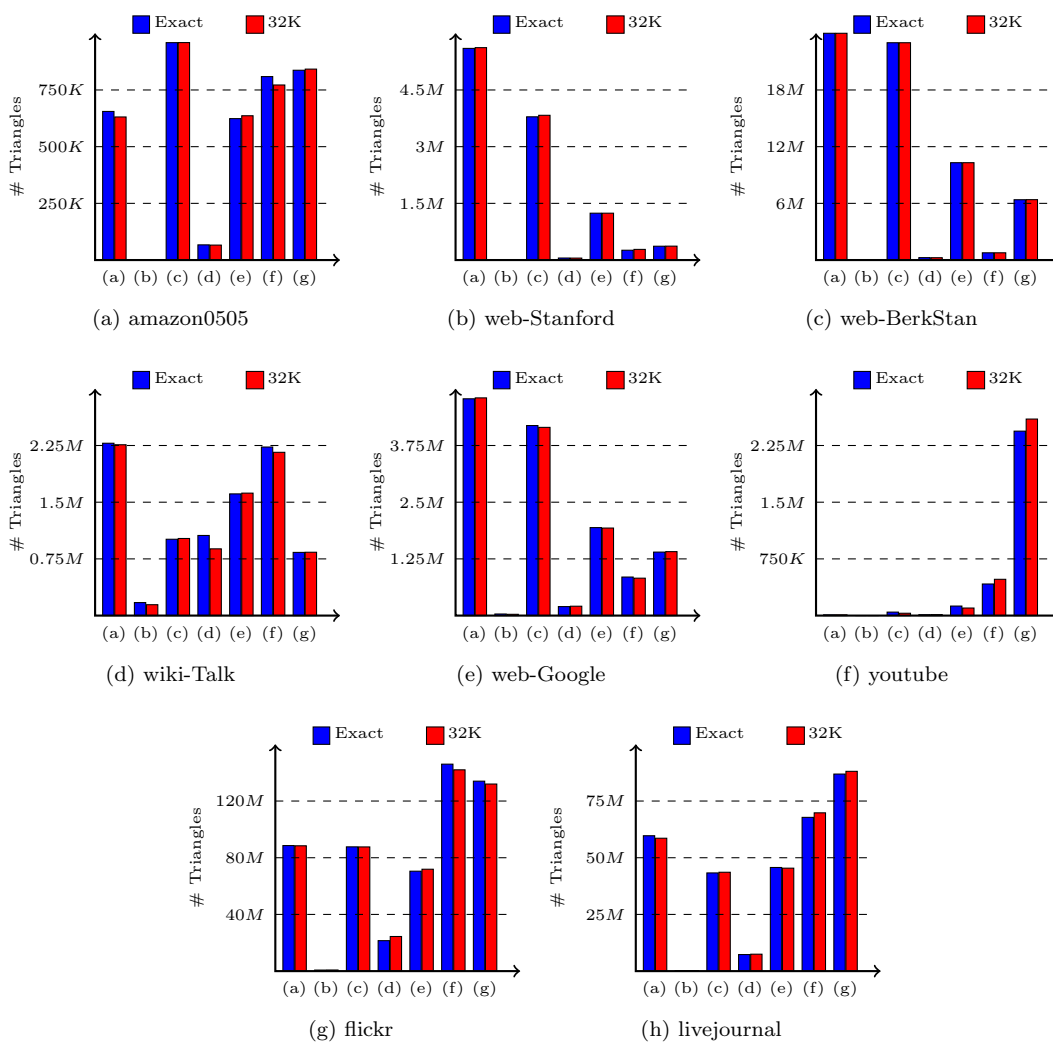


Figure 11: Directed triangle counts using wedge sampling

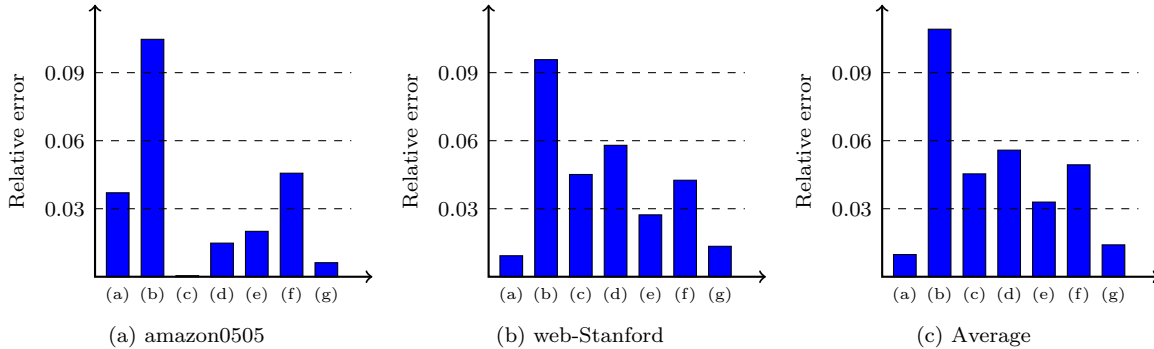


Figure 12: Relative error for the 7 triangle types

We take a closer look at the results, using *relative error* plots. For brevity, we provide these only for `amazon0505` and `web-Stanford` graphs, and give an average over all the graphs. A clear pattern appears, where the relative error for type-(b) triangles is worse than the others. This is where a weakness of wedge sampling becomes apparent: When a triangle type is extremely infrequent, wedge sampling is limited in the quality of the estimate. More details are shown in [Tab. 4](#), where we show the relative error for these triangles, as well the fraction of these triangles (with respect to the total count). Across the board, we see that type-(b) triangles are extremely rare; hence, wedge sampling is unable to get sharp estimates. It would be very interesting to design a fast method that accurately counts such rare triangles.

Table 4: A closer look at type-(b) triangles

Graph	Relative Error	Count	Ratio to all triangles
amazon0505	0.105	623	0.00013
web-Stanford	0.061	9016	0.00053
web-BerkStan	0.063	23035	0.00026
wiki-Talk	0.162	171903	0.01497
web-Google	0.166	34418	0.00189
flickr	0.030	773098	0.00111
youtube	0.113	118	0.00002
livejournal	0.173	188666	0.00047

There are some subtleties about the implementation that are worth mentioning. We have a choice of which wedge types to use to count ρ -triangles. We can get some reuse if we use the same wedge type to count multiple triangles. For instance, type-(a) triangles can be counted using sampling over type (i), (ii), or (iii) wedges. By [Thm. 7.1](#), sampling over ψ -wedges yields error is $\varepsilon W_\psi / \chi(\psi, \rho)$. Hence, *less frequent wedge types give better approximations* for the same triangle type. Symmetrically, the same wedge type can be used to count multiple triangle types. For instance, (ii)-wedges can be used to count type (a), (b), and (d) triangles. We use the following wedges for our triangle counts:

- type-(ii) wedges are used to count triangles of type (a) and (b),
- type-(v) wedges are used to count triangles of type (c) and (f),
- type-(iv) wedges are used to count triangles of type (d) and (e), and

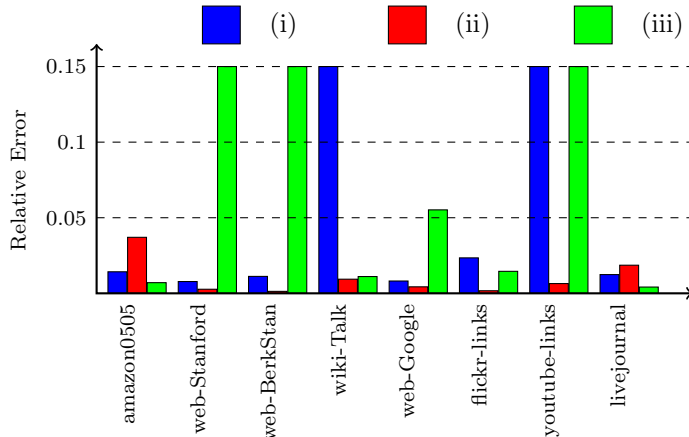


Figure 13: Estimates of the number of type-(a) triangles using different wedges types. Sample size is fixed to 32K.

- type-(vi) wedges are used to count triangles of type (g).

The combinations that we use are boldface in Fig. 10d. The choice of wedge can impact the accuracy of the directed triangle counts. For example, type-(a)-triangles can be counted using wedges of type (i), (ii), or (iii). In Fig. 13, we show the relative error in the estimate for type-(a) triangles using 32K samples via the different wedge types. Because of the high frequency of type-(iii) wedges, the error in triangle counts using these wedges is quite large. Other than one case, (ii)-wedges are the best choice.

8 Conclusions and future work

We proposed a series of wedge-based algorithms for computing various triadic measures on graphs. Our algorithms come with theoretical guarantees in the form of specific error and confidence bounds. The number of samples required to achieve a specified error and confidence bound is independent of the graph size. For instance, 38,000 samples suffice for an additive error in the transitivity of less than 0.01 with 99.9% confidence *for any graph*. The limiting factors have to do with determining the sampling proportions; for instance, we need to know the degree of each vertex and the overall degree distribution to compute the transitivity.

The flexibility of wedge sampling along with the hard error bounds essentially redefines what is feasible in terms of graph analysis. The very expensive computation of clustering coefficient is now much faster and we can consider much larger graphs than before. In an extension of this work, we described a MapReduce implementation of this method that scales to billions of edges, leading to some of the first published results at these sizes [18].

With triadic analysis no longer being a computational burden, we can extend our horizons into new territories and look at attributed triangles (e.g., we might compare the clustering coefficient for “teenage” versus “adult” nodes in a social network), evolution of triadic connections, higher-order structures such as 4-cycles and 4-cliques, and so on.

References

- [1] S. M. ARIFUZZAMAN, M. KHAN, AND M. V. MARATHE, *PATRIC: A parallel algorithm for counting triangles and computing clustering coefficients in massive networks*, NDSSL Technical Report 12-

- 042, Network Dynamics and Simulation Science Laboratory, Virginia Polytechnic Institute and State University, July 2012.
- [2] H. AVRON, *Counting triangles in large graphs using randomized matrix trace estimation*, in KDD-LDMTA'10, 2010.
 - [3] Z. BAR-YOSSEF, R. KUMAR, AND D. SIVAKUMAR, *Reductions in streaming algorithms, with an application to counting triangles in graphs*, in SODA'02, 2002, pp. 623–632.
 - [4] L. BECCHETTI, P. BOLDI, C. CASTILLO, AND A. GIONIS, *Efficient semi-streaming algorithms for local triangle counting in massive graphs*, in KDD'08, 2008, pp. 16–24.
 - [5] J. W. BERRY, L. FOSVEDT, D. NORDMAN, C. A. PHILLIPS, AND A. G. WILSON, *Listing triangles in expected linear time on power law graphs with exponent at least $\frac{7}{3}$* , Tech. Report SAND2010-4474C, Sandia National Laboratories, 2011.
 - [6] L. S. BURIOL, G. FRAHLING, S. LEONARDI, A. MARCHETTI-SPACCAMELA, AND C. SOHLER, *Counting triangles in data streams*, in PODS'06, 2006, pp. 253–262.
 - [7] R. S. BURT, *Structural holes and good ideas*, American Journal of Sociology, 110 (2004), pp. 349–399.
 - [8] N. CHIBA AND T. NISHIZEKI, *Arboricity and subgraph listing algorithms*, SIAM J. Computing, 14 (1985), pp. 210–223.
 - [9] S. CHU AND J. CHENG, *Triangle listing in massive networks and its applications*, in KDD'11, 2011, pp. 672–680.
 - [10] J. COHEN, *Graph twiddling in a MapReduce world*, Computing in Science & Engineering, 11 (2009), pp. 29–41.
 - [11] J. S. COLEMAN, *Social capital in the creation of human capital*, American Journal of Sociology, 94 (1988), pp. S95–S120.
 - [12] N. DURAK, A. PINAR, T. G. KOLDA, AND C. SESHADHRI, *Degree relations of triangles in real-world networks and graph models*, in Proceedings of the 21st ACM international conference on Information and knowledge management, CIKM '12, New York, NY, USA, 2012, ACM, pp. 1712–1716.
 - [13] J.-P. ECKMANN AND E. MOSES, *Curvature of co-links uncovers hidden thematic layers in the World Wide Web*, PNAS, 99 (2002), pp. 5825–5829.
 - [14] B. FOUCAULT WELLES, A. VAN DEVENDER, AND N. CONTRACTOR, *Is a friend a friend?: Investigating the structure of friendship networks in virtual worlds*, in CHI-EA'10, 2010, pp. 4027–4032.
 - [15] W. HOEFFDING, *Probability inequalities for sums of bounded random variables*, J. American Statistical Association, 58 (1963), pp. 13–30.
 - [16] M. JHA, C. SESHADHRI, AND A. PINAR, *A space efficient streaming algorithm for triangle counting using the birthday paradox*, in KDD'13, 2013, pp. 589–597.
 - [17] H. JOWHARI AND M. GHODSI, *New streaming algorithms for counting triangles in graphs*, in COCOON'05: Computing and Combinatorics, L. Wang, ed., vol. 3595 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 710–716.
 - [18] T. G. KOLDA, A. PINAR, T. PLANTENGA, C. SESHADHRI, AND C. TASK, *Counting triangles in massive graphs with MapReduce*. arXiv:1301.5887, January 2013.
 - [19] M. N. KOLOUNTZAKIS, G. L. MILLER, R. PENG, AND C. TSOURAKAKIS, *Efficient triangle counting in large graphs via degree-based vertex partitioning*, in WAW'10, 2010.
 - [20] M. LATAPY, *Main-memory triangle computations for very large (sparse (power-law)) graphs*, Theoretical Computer Science, 407 (2008), pp. 458–473.
 - [21] A. MISLOVE, M. MARCON, K. P. GUMMADI, P. DRUSCHEL, AND B. BHATTACHARJEE, *Measurement and analysis of online social networks*, in IMC'07, ACM, 2007, pp. 29–42.
 - [22] R. PAGH AND C. TSOURAKAKIS, *Colorful triangle counting and a mapreduce implementation*, Information Processing Letters, 112 (2012), pp. 277–281.
 - [23] A. PORTES, *Social capital: Its origins and applications in modern sociology*, Annual Review of Sociology, 24 (1998), pp. 1–24.
 - [24] T. SCHANK AND D. WAGNER, *Approximating clustering coefficient and transitivity*, Journal of Graph Algorithms and Applications, 9 (2005), pp. 265–275.
 - [25] —, *Finding, counting and listing all triangles in large graphs, an experimental study*, in Experimental

- and Efficient Algorithms, Springer Berlin / Heidelberg, 2005, pp. 606–609.
- [26] C. SESHADHRI, T. G. KOLDA, AND A. PINAR, *Community structure and scale-free collections of Erdős-Rényi graphs*, Physical Review E, 85 (2012), p. 056109.
 - [27] C. SESHADHRI, A. PINAR, N. DURAK, AND T. G. KOLDA, *The importance of directed triangles with reciprocity: Patterns and algorithms*. arXiv:1302.6220, February 2013.
 - [28] C. SESHADHRI, A. PINAR, AND T. G. KOLDA, *Triadic measures on graphs: The power of wedge sampling*, in SDM13: Proceedings of the 2013 SIAM International Conference on Data Mining, May 2013, pp. 10–18.
 - [29] S. SURI AND S. VASSILVITSKII, *Counting triangles and the curse of the last reducer*, in WWW'11, 2011, pp. 607–614.
 - [30] C. TSOURAKAKIS, P. DRINEAS, E. MICHELAKIS, I. KOUTIS, AND C. FALOUTSOS, *Spectral counting of triangles in power-law networks via element-wise sparsification*, in ASONAM'09, 2009, pp. 66–71.
 - [31] C. TSOURAKAKIS, M. N. KOLOUNTZAKIS, AND G. MILLER, *Triangle sparsifiers*, Journal of Graph Algorithms and Applications, 15 (2011), pp. 703–726.
 - [32] C. E. TSOURAKAKIS, *Fast counting of triangles in large real networks, without counting: Algorithms and laws*, in ICDM 2008, 2008, pp. 608–617.
 - [33] C. E. TSOURAKAKIS, U. KANG, G. L. MILLER, AND C. FALOUTSOS, *Doulion: counting triangles in massive graphs with a coin*, in KDD '09, 2009, pp. 837–846.
 - [34] S. WASSERMAN AND K. FAUST, *Social Network Analysis: Methods and Applications*, Cambridge University Press, 1994.
 - [35] D. WATTS AND S. STROGATZ, *Collective dynamics of 'small-world' networks*, Nature, 393 (1998), pp. 440–442.
 - [36] J.-H. YOON AND S.-R. KIM, *Improved sampling for triangle counting with MapReduce*, in Convergence and Hybrid Information Technology, G. Lee, D. Howard, and D. Slezak, eds., vol. 6935 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2011, pp. 685–689.
 - [37] *Stanford Network Analysis Project (SNAP)*. Available at <http://snap.stanford.edu/>.