Triadic Measures on Graphs: The Power of Wedge Sampling

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 a graph. Some of the most useful graph metrics, especially those measuring social cohesion, are based on triangles. Despite the importance of these triadic measures, associated algorithms can be extremely expensive. We discuss the method of wedge sampling. This versatile technique allows for the fast and accurate approximation of all current variants of clustering coefficients and enables rapid uniform sampling of the triangles of a graph. Our methods come with provable and practical time-approximation tradeoffs for all computations. We provide extensive results that show our methods are orders of magnitude faster than the state-of-the-art, while providing nearly the accuracy of full enumeration. Our results will enable more wide-scale adoption of triadic measures for analysis of extremely large graphs, as demonstrated on several real-world examples.

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. The primary of these is the triangle, consider a manifestation of homophily (people become friends with those similar to themselves) and transitivity (friends of friends become friends). The abundance of triangles, along with the information they reveal, motivates metrics such as the clustering coefficient and the transitivity ratio [31, 32]. The triangle structure of a graph is commonly used in the social sciences for positing various theses on behavior [11, 21, 7, 14]. Triangles have also been used in graph mining applications such as spam detection and finding common topics on the WWW [13, 4]. The authors’ earlier work used distribution of degree-wise clustering coefficients as the driving force for a new generative model, Blocked Two-Level Erdös-Rényi [24]. The authors’ have also observed that relationships among degrees of triangle vertices can be a descriptor of the underlying graph [12].

1.1 Clustering coefficients The information about triangles is usually summarized in terms of clustering coefficients. Let \( G \) be a simple undirected graph with \( n \) vertices and \( m \) edges. Let \( T \) denote the number of triangles in the graph and \( W \) be the number of wedges (a path of length 2). The most common measure is the global clustering coefficient 
\[
C = \frac{3T}{W},
\]

which measures how often friends of friends are also friends. We show that we can achieve speed-ups of up to four orders of magnitude with extremely small errors; see Fig. 1 and Fig. 2.

Figure 1: Speed-up over enumeration for global clustering coefficient computation with increasing numbers of wedge samples

Our approach is not limited to clustering coefficients, however. A per-vertex clustering coefficient, \( C_v \), is defined as the fraction of wedges centered at \( v \) that...
1.2 Related Work

There has been significant work on exact enumeration of all triangles \([8, 23, 18, 5, 9]\). Recent work by Cohen [10] and Suri and Vassilvitskii [26] give parallel implementations of these algorithms. Arifulazzaman 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 [27] 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.

Most relevant to our work are sampling mechanisms. Tsourakakis et al. [28] started the use of sparsification methods, the most important of which is Doulion [30]. 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 [17, 29, 20]. One of the main benefits of Doulion is that it reduces large graphs to smaller ones that can be loaded into memory. However, their estimate can suffer from high variance [33]. Alternative sampling mechanisms have been proposed for streaming and semi-streaming algorithms [3, 16, 4, 6].

Yet, all these fast sampling methods only estimate the number of triangles and give no information about other triadic measures.

1.3 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, as opposed to a single algorithm, as various algorithms can be obtained from the same basic structure.

The authors recently discovered an independent older paper by Schank and Wagner introducing the ideas of wedge sampling [22]. The analysis is pretty much identical to ours, although they do not deal with degree-wise clustering coefficients. We perform a more exhaustive empirical validation on much larger graphs. Nonetheless, the use of wedge sampling for counting triangles is older than our work. (Unfortunately, the Schank-Wagner paper is almost never cited by papers in data mining or social network analysis.)

Some of the salient features of this method are:

- **Versatility of wedge sampling**: We show how to use wedge sampling to approximate the various clustering coefficients: \(C\), \(\overline{C}\), and \(\{C_d\}\). From these, we can estimate the numbers of triangles: \(T\) and \(\{T_d\}\). Moreover, our techniques can even be extended to find uniform random triangles, which is useful for computing triadic statistics. Other sampling methods are usually geared towards only \(T\) and \(C\).

- **Provably good results with precise bounds**: The mathematical analysis of this method is a direct consequence of standard Chernoff-Hoeffding bounds.

### Table 1: Graph notation and clustering coefficients

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>number of vertices</td>
</tr>
<tr>
<td>(n_d)</td>
<td>number of vertices of degree (d)</td>
</tr>
<tr>
<td>(m)</td>
<td>number of edges</td>
</tr>
<tr>
<td>(d_v)</td>
<td>degree of vertex (v)</td>
</tr>
<tr>
<td>(V_d)</td>
<td>set of degree-(d) vertices</td>
</tr>
<tr>
<td>(W)</td>
<td>total number of wedges</td>
</tr>
<tr>
<td>(W_v)</td>
<td>number of wedges centered at vertex (v)</td>
</tr>
<tr>
<td>(T)</td>
<td>total number of triangles</td>
</tr>
<tr>
<td>(T_v)</td>
<td>number of triangles centered to vertex (v)</td>
</tr>
<tr>
<td>(T_d)</td>
<td>number of triangles incident to a degree (d) vertex</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formula</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C = \frac{3T}{W})</td>
<td>global clustering coefficient</td>
</tr>
<tr>
<td>(C_v = \frac{T_v}{W_v})</td>
<td>clustering coefficient of vertex (v)</td>
</tr>
<tr>
<td>(\overline{C} = \frac{1}{n} \sum_v C_v)</td>
<td>local clustering coefficient</td>
</tr>
<tr>
<td>(C_d = \frac{n_d}{W} \sum_{v \in V_d} C_v)</td>
<td>degree-wise clustering coefficient</td>
</tr>
</tbody>
</table>

participate in triangles. The mean value of \(C_v\) is called the **local clustering coefficient**, \(\overline{C}\). A more nuanced view of triangles is given by the **degree-wise clustering coefficients**. This is a set of values \(\{C_d\}\) indexed by degree, where \(C_d\) is the average clustering coefficient of degree \(d\) vertices. In addition to degree distribution, many graphs are characterized by plotting the clustering coefficients, i.e., \(C_d\) versus \(d\). We summarize our notation and give formal definitions in Tab. 1.
We obtain explicit time-error-accuracy tradeoffs. In other words, if we want an estimate for \( C \) with error at most 10\% with probability at least 99.9\% (say), we know 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).

- **Fast and scalable:** Our estimates converge rapidly, well within the theoretical bounds provided. Although there is no other method that competes directly with wedge sampling, we compare with Doullon [30]. Our experimental studies show that our wedge sampling algorithm is far faster, when the variances of the two methods are similar. We do not compare to eigenvalue-based approaches since they are much more expensive for larger graphs.

2 The wedge sampling method

We present the general method of wedge sampling for estimating clustering coefficients. In later sections, we instantiate this for different algorithms.

We say a wedge is closed if it is part of a triangle; otherwise, we say the wedge is open. Thus, in Fig. 3, \( 3-4-6 \) 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 \( 3-4-6 \) and \( 3-4-5 \) are centered at \( 4 \).

![Figure 3: Example graph with 12 wedges and 1 triangle.](image)

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 \( \mu = \mathbb{E}[X] \).

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

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

\[
\Pr \left\{ |\bar{X} - \mu| \geq \varepsilon \right\} \leq 2 \exp(-2k\varepsilon^2).
\]

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

Fig. 4 shows the number of samples needed for different error rates. We show three different curves for difference 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 the desired error.

![Figure 4: The number of samples needed for different error rates and different levels of confidence. A few data points at 99.9\% confidence are highlighted.](image)

3 Computing the global clustering coefficient and the number of triangles

We use the wedge sampling scheme to estimate the global clustering coefficient, \( C \). Consider the uniform distribution on wedges. We can interpret \( \mathbb{E}[X] \) 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 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 \( 1/W \).
Table 2: Properties of the graphs and runtimes for enumeration, wedge sampling for C, and wedge sampling for 2C. The sampling procedures use 32K samples for all graphs.

<table>
<thead>
<tr>
<th>Graph</th>
<th>n</th>
<th>m</th>
<th>W</th>
<th>T</th>
<th>C</th>
<th>C</th>
<th>Enum</th>
<th>Wedge C</th>
<th>Wedge C</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazon0312</td>
<td>401K</td>
<td>2350K</td>
<td>69M</td>
<td>3686K</td>
<td>0.160</td>
<td>0.421</td>
<td>0.261</td>
<td>0.016</td>
<td>0.008</td>
</tr>
<tr>
<td>amazon0505</td>
<td>410K</td>
<td>2439K</td>
<td>73M</td>
<td>3951K</td>
<td>0.162</td>
<td>0.427</td>
<td>0.269</td>
<td>0.016</td>
<td>0.008</td>
</tr>
<tr>
<td>amazon0601</td>
<td>403K</td>
<td>2443K</td>
<td>72M</td>
<td>3987K</td>
<td>0.166</td>
<td>0.430</td>
<td>0.268</td>
<td>0.017</td>
<td>0.007</td>
</tr>
<tr>
<td>as-skitter</td>
<td>1696K</td>
<td>11095K</td>
<td>16022M</td>
<td>28770K</td>
<td>0.005</td>
<td>0.296</td>
<td>90.897</td>
<td>0.026</td>
<td>0.036</td>
</tr>
<tr>
<td>cit-Patents</td>
<td>377K5</td>
<td>16519K</td>
<td>336M</td>
<td>7515K</td>
<td>0.067</td>
<td>0.092</td>
<td>3.764</td>
<td>0.056</td>
<td>0.012</td>
</tr>
<tr>
<td>roadNet-CA</td>
<td>1965K</td>
<td>2767K</td>
<td>6M</td>
<td>121K</td>
<td>0.060</td>
<td>0.055</td>
<td>0.995</td>
<td>0.037</td>
<td>0.006</td>
</tr>
<tr>
<td>web-BerkStan</td>
<td>685K</td>
<td>6649K</td>
<td>27983M</td>
<td>64691K</td>
<td>0.007</td>
<td>0.634</td>
<td>54.777</td>
<td>0.016</td>
<td>0.085</td>
</tr>
<tr>
<td>web-Google</td>
<td>876K</td>
<td>4322K</td>
<td>727M</td>
<td>13392K</td>
<td>0.055</td>
<td>0.624</td>
<td>0.894</td>
<td>0.020</td>
<td>0.013</td>
</tr>
<tr>
<td>web-Stanford</td>
<td>282K</td>
<td>1993K</td>
<td>3944M</td>
<td>11329K</td>
<td>0.009</td>
<td>0.629</td>
<td>6.987</td>
<td>0.111</td>
<td>0.038</td>
</tr>
<tr>
<td>wiki-Talk</td>
<td>2394K</td>
<td>4660K</td>
<td>12594M</td>
<td>9204K</td>
<td>0.002</td>
<td>0.201</td>
<td>20.572</td>
<td>0.033</td>
<td>0.108</td>
</tr>
<tr>
<td>youtube</td>
<td>1158K</td>
<td>2990K</td>
<td>1474M</td>
<td>3057K</td>
<td>0.006</td>
<td>0.128</td>
<td>2.740</td>
<td>0.021</td>
<td>0.031</td>
</tr>
<tr>
<td>flickr</td>
<td>1861K</td>
<td>15555K</td>
<td>14670M</td>
<td>548659K</td>
<td>0.112</td>
<td>0.375</td>
<td>567.160</td>
<td>0.051</td>
<td>0.021</td>
</tr>
<tr>
<td>livejournal</td>
<td>5284K</td>
<td>48710K</td>
<td>7519M</td>
<td>310877K</td>
<td>0.124</td>
<td>0.345</td>
<td>102.142</td>
<td>0.073</td>
<td>0.015</td>
</tr>
</tbody>
</table>

Algorithm 1 C-wedge sampler
1: Compute $p_v$ for all vertices
2: Select $k$ random vertices (with replacement) according to probability distribution defined by $\{p_v\}$.
3: For each selected vertex $v$, choose a uniform random pair of neighbors of $v$ to generate $w$.
4: Output fraction of closed wedges as estimate of $C$.

Algorithm 1 shows the randomized algorithm C-wedge sampler for estimating $C$ in a graph $G$. Combining the bound of Thm. 2.1 with Claim 3.1, we get the following theorem.

**Theorem 3.1.** Set $k = \lceil 0.5e^{-2\ln(2/\delta)} \rceil$. The algorithm C-wedge sampler outputs an estimate $\hat{X}$ for the clustering coefficient $C$ such that $|\hat{X} - C| < \varepsilon$ with probability greater than $(1 - \delta)$.

Note that the number of samples required is independent of the graph size, but computing $p_v$ does depend on the number of edges, $m$.

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

3.1 Experimental results 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 an 8GB memory. We performed our experiments on 13 graphs from SNAP [34] and per private communication with the authors of [19]. In all cases, directionality is ignored, and repeated and self-edges are omitted. The properties of these matrices are presented in Tab. 2, where $n$, $m$, $W$, and $T$ are the numbers of vertices, edges, wedges, and triangles, respectively. And $C$ and $\overline{C}$ correspond to the global and local clustering coefficients. The last column reports the times for the enumeration algorithm. Our enumeration algorithm is based on the principles of [8, 23, 10, 26], such that each edge is assigned to its 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. 1 wedge sampling works 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. 2 show 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. Fig. 5 shows the fast convergence of the clustering coefficient estimate (for the graph amazon0505) as the number of samples increases. The dashed line shows the error bars at 99.9% confidence. In all our experiments, the real error is always much smaller than what is indicated by Thm. 2.1.

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

Unauthorized reproduction of this article is prohibited.
Algorithm 2 $C$-wedge sampler

1: Pick $k$ uniform random vertices.
2: For each selected vertex $v$, choose a uniform random pair of neighbors of $v$ to generate a wedge.
3: Output the fraction of closed wedges 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 $X(w) = 1$ if the wedge is closed, and $X(w) = 0$ otherwise. Let $V$ be the uniform distribution on wedges. For any vertex $v$, let $N_v$ be the uniform distribution on pairs of neighbors of $v$. Observe that

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

We will show that this is exactly $C$.

$$C = n^{-1} \sum_v C_v = E_{v \sim V} [C_v]$$

We have

$$C = \frac{1}{n} \sum_v C_v = \frac{1}{n} \sum_v \frac{1}{2} \left( \sum_{u, u'} \Pr_{(u, u') \sim N_v} \left[ \text{wedge } \{(u, v), (u', v)\} \text{ is closed} \right] \right)$$

For a single sample, the probability that the wedge is closed is exactly $C$. The bound of Thm. 2.1 completes the proof.

Fig. 6 and Fig. 7 present the results of our experiments for computing the local clustering coefficients. Experimental setup and the notation are the same as in §3.1. The results again show that wedge sampling provides accurate estimations with tremendous improvements in runtime. In this case, we come closer to the theoretical error bounds. For instance, the largest different in the 2K sample case is 0.017, which is much closer to the theoretical error bound of 0.043.

Figure 5: Convergence of clustering coefficient estimate as the number of samples increases for amazon0505.

Figure 6: Absolute error in local clustering coefficient for increasing numbers of wedge samples

Figure 7: Speed-up in local clustering coefficient computation time for increasing numbers of wedge samples

5 Computing degree-wise clustering coefficients and triangle estimates

We demonstrate the power of wedge sampling by estimating the degree-wise clustering coefficients $\{C_d\}$. We also provide a sampling algorithm to estimate $T_d$, the number of triangles incident to degree-$d$ vertices. Alg. 3 shows procedure $C_d$-wedge sampler.

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

Algorithm 3 $C_d$-wedge sampler

1: Pick $k$ uniform random vertices of degree $d$.
2: For each selected vertex $v$, choose a uniform random pair of neighbors of $v$ to generate a wedge.
3: Output the fraction of closed wedges as an estimate for $C_d$.

Proof. The proof of the following is similar 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. 

By modifying the template given in §2, we can also get estimates for $T_d$. Now, instead of simply 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.

Algorithm 4 $T_d$-wedge sampler

1: Pick $k$ uniform random vertices of degree $d$.
2: For each selected vertex $v$, choose a uniform random pair of neighbors of $v$ to generate a wedge.
3: For each wedge generated, 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: $\bar{Y} = \frac{1}{k} \sum Y_i$.
5: Output $W_d \cdot \bar{Y}$ as the estimate for $T_d$.

Theorem 5.2. Set $k = \lceil 0.5e^{-2 \ln(2/\delta)} \rceil$. 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 that the expected value of $E[Y]$ is exactly $T_d/W_d$ below. Once we have that, an application of the Hoeffding bound of Thm. 2.1 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 $E[Y] = T_d/W_d$, partition the set $S$ of wedges centered on degree $d$ vertices into four sets $S_0, S_1, S_2, S_3$. 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 nothing is added. The wedge $w$ is a uniform random wedge from those centered on degree-$d$ vertices. Hence, $E[Y] = |S|^{-1} (|S_1| + |S_2|/2 + |S_3|/3)$.

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, $|S_1| + |S_2|/2 + |S_3|/3 = |S'_1| + |S'_2|/2 + |S'_3| = T_d$. Therefore, $E[Y] = T_d/W_d$. 

5.1 Computing the clustering coefficient for bins of vertices 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 still be extended for bins of degrees by a small adjustment of the
sampling procedure. 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.

5.2 Experimental results Fig. 8 shows results on three graphs for clustering coefficients. The data is grouped in logarithmic bins of degrees, i.e., \{2\}, \{3, 4\}, \{5, 6, 7, 8\}, \ldots. In other words, \(2^{i-1} < d_v \leq 2^i\) form the \(i\)-th bin. 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. 9 shows the time to calculate all \(C_d\) values, showing 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.

6 Generating a uniform sample of the triangles

While most triadic measures focus on the number of triangles and their distribution, the triangles themselves, not only their count, can reveal a lot of information about the graph. The authors’ recent work [12] has looked at the relations among the degrees of the vertices of triangles. In these experiments, a full enumeration of the triangles was used, which limited the sizes of the graphs we could use. To avoid this computational burden, a uniform sampling of the triangles can be used.

Wedge sampling provides a uniform sampling the triangles of a graph. Consider the uniform wedge sampling of Alg. 1. Some of these wedges will be closed, so we generate a random set of triangles. Each such triangle is an independent, uniform random triangle. This is because wedges are chosen from the uniform distribution, and every triangles contains exactly 3 closed wedges. Fig. 10 presents the results using triangle sampling to estimate the percentage of triangles where the maximum to minimum degree ratio is \(\geq 10\), which is motivated by an experiment in [12]. The figure shows that accurate results can be achieved by using only 500 triangles and that wedge sampling provides an unbiased selection of triangles. The expected number of wedges to be sampled to generate \(T_s\) triangles is \(3T_s / C\), which means the method will be effective unless the clustering coefficient is extremely small.

Figure 9: Speed-up in degree-wise clustering coefficient computation time for increasing numbers of wedge samples

Figure 10: Error in computing percentage of triangles where the maximum to minimum degree ratio is \(\geq 10\) for increasing number of samples.

7 Comparison to Doulion

Doulion [30] is an alternative sampling mechanism for estimating the number of triangles in a graph. It has a single parameter \(p\). Each edge is chosen independently at random with probability \(p\), leading to a subgraph of expected size \(pm\) (\(m\) is the total number of edges). We count the number of triangles \(T'\) in this subgraph, and estimate the total number of triangles by \(T'/p^3\). It is not hard to verify that this is correct in expectation, but bounding the variance requires a lot more work. Some concentration bounds for this estimate are known [29, 20], but they depend on the maximum number of triangles incident to an edge in the graph. So they do not have the direct form of Thm. 3.1. Some bad examples for Doulion have been observed [33]. Doulion is extremely elegant and simple, and leads to an overall reduction in graph size (so a massive graph can now fit into memory). Common values of \(p\) used are \(p = 1/10\) and \(p = 1/25\).

We show that wedge sampling performs at least as good as Doulion in terms of accuracy, and has better runtimes. We run wedge sampling with 32K samples.
We start with setting the Doulion parameter $p = 1/25$, which has been used in the literature. Note that the size of the Doulion sample is $m/25$, which is much larger than 32K. (For Amazon0312, one of the smaller graphs we consider, $m/25 \approx 90K$.) We run each algorithm 100 times. The (average) runtimes are compared in Fig. 11, where we see that wedge sampling is always competitive. The accuracy of the estimate is comparable for both algorithms. Detailed results are available in the full version [25].

8 Significance and Impact

We have 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. We want to stress that the number of samples required to achieve a specified error and confidence bound is independent of the graph size. For instance, 38,000 samples guarantees and error less than 0.1% with 99.9% confidence for any graph, which gives our algorithms an incredible potential for scalability. 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 global clustering coefficient.

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 (enabling more near-real-time analysis of networks) and we can consider much larger graphs than before. In an extension of this work, we are pursuing a MapReduce implementation of this method that scales to $O(100M)$ nodes and $O(1B)$ edges, needing only a few minutes of compute time (based on preliminary results).

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

References


