SAND2014-15835C

Path Sampling: A Fast and Provable Method for Estimating 4-Vertex Subgraph Counts

Madhav Jha
Sandia National Laboratories
7011 East Avenue
Livermore, CA 94550
Email: mjha@sandia.gov

Ali Pinar
Sandia National Laboratories
7011 East Avenue
Livermore, CA 94550
Email: apinar@sandia.gov

C. Seshadhri
Sandia National Laboratories
7011 East Avenue
Livermore, CA 94550
Email: scomand@sandia.gov

***Abstract*—Counting the frequency of small subgraphs is a fundamental technique in network analysis across various domains, most notably in bioinformatics and social networks. The special case of triangle counting has received much attention. Getting results for 4-vertex patterns is highly challenging, and there are few practical results known that can scale to massive sizes. Indeed, even a highly tuned enumeration code takes more than a day on a graph with millions of edges. Most previous work that runs for truly massive graphs employ clusters and massive parallelization.**

We provide a sampling algorithm that provably and accurately approximates *all* 4-vertex pattern subgraphs. Our algorithm is based on a novel technique of *3-path sampling* and a special pruning scheme to decrease the variance in estimates. We provide theoretical proofs for the accuracy of our algorithm, and give formal bounds for the error and confidence of our estimates. We perform a detailed empirical study and show that our algorithm provides estimates within 1% relative error for all subpatterns (over a large class of test graphs). Our algorithm takes less than a minute (on a single commodity machine) to process an Orkut social network with 300 million edges.

I. Introduction

Counting the number of occurrences of small subgraphs in a graph is a fundamental network analysis technique used across diverse domains: bioinformatics, social sciences, and infrastructure networks studies [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. The subgraphs whose counts are desired are variously referred as "pattern subgraphs", "motifs", or "graphlets". It is repeatedly observed that certain small subgraphs occur substantially more often in real-world networks than in a randomly generated network [1], [14], [4]. Motifs distributions have been used in bioinformatics to evaluate network models [6], [15]. Analysis of triadic (3-vertex) motifs has a long history in social network analysis and modeling [1], [5], [7], [16], [17]. Work in the data mining community has applied motif frequencies for spam detection and group classification of sets of nodes [9], [18].

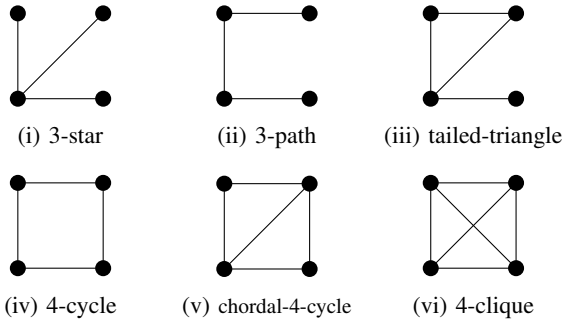
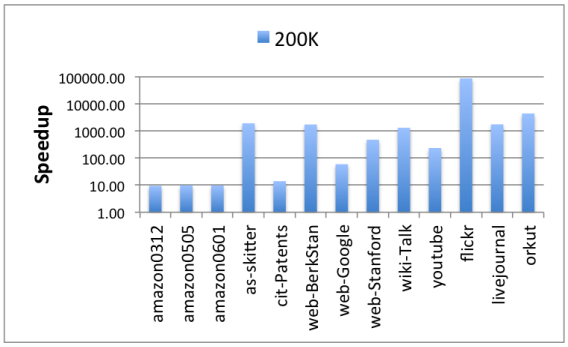


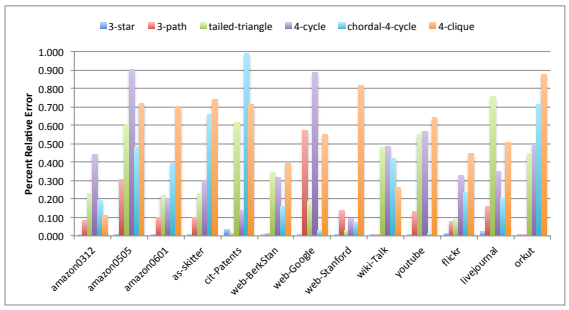
Fig. 1: List of all connected 4-vertex motifs

The main challenge of motif counting is combinatorial explosion. Even in a moderately sized graph with millions of edges, the subgraph counts (even for 4-vertex patterns) is in the billions. Any exhaustive enumeration method (no matter how cleverly designed) is forced to touch each occurrence of the subgraph, and cannot truly scale. One may apply massive parallelism to counteract this problem, but that does not avoid the fundamental combinatorial explosion. An alternative approach is based on *sampling*. Here, we try to count the number of subgraphs using a randomized algorithm. The difficulty is in designing a fast algorithm that also provides an accurate estimate. The holy grail is to get mathematically provable bounds on accuracy with quantifiable error bars.

Sampling approaches have been employed for triangle counting with good success [19], [20], [21], [22], [23]. There also exists work for counting larger motifs, as we shall discuss later. Most methods (especially in bioinformatics) [15], [24], [25], [26] works for graphs of at most 100K edges, much smaller than the massive social networks we encounter.



(i) Speedup



(ii) Relative error

Fig. 2: Summary of 3-path sampling algorithm behavior over a large variety of datasets: The left figure shows speedup over a tuned enumeration code. The right figure shows the relative error of each estimate, which is always less than 1% (and mostly much smaller).

A. *The main problem*

We focus on estimating frequency of all connected 4-vertex subgraphs on massive input graphs. There are six connected 4-vertex graphs (Fig. 1): (i) the 3-path, (ii) the 3-star, (iii) the tailed-triangle, (iv) the 4-cycle, (v) the chordal-4-cycle, and (vi) the 4-clique. Throughout this work, we refer to these motifs by their numbering in this list. For example the "6-th motif" is the 4-clique.

Our aim is to give an accurate and fast estimate of all 4-vertex subgraph counts. Triadic analysis is now a standard aspect of network analysis. Recent work of Ugander et al [18] specifically use 4-vertex pattern counts to provide a "map" of egonets, and show significant patterns among these counts. Such analyses require fairly precise frequency counts.

B. *Summary of our contributions*

We design a new randomized algorithm, based on *3-path sampling*, which outputs accurate estimates of all 4-vertex subgraphs counts. We stress that the algorithm is provably correct and makes no distributional assumption on the graph. All probabilities are over the internal randomness of the algorithm itself (which is independent of the instance). We run detailed simulations on a large variety of datasets, including product co-purchasing networks, web networks, autonomous systems networks, and social networks. All experiments are done on a single commodity machine using 64GB memory.

Extremely fast. Our algorithm relies on a sampling based approach making it extremely fast even on very large graphs. Indeed, there are instances where a finely tuned enumeration code takes almost a day to compute counts of 4-vertex motifs whereas our algorithm only takes less than a minute to output accurate estimates. Refer to Fig. 2i for speedup over a well-tuned enumeration code. Our algorithm takes less a minute on an Orkut social network with 200 million edges, where the total count of each motif is over a billion (and most counts are over 10 billion). An input Flickr social network has more than 10 billion 6-cliques; we get estimate of this number with less than 0.5% error within 30 seconds on a commodity machine. We do not preprocess any of the graphs, and simply read them as a list of edges.

Excellent empirical accuracy. We empirically validate our algorithm on a large variety of datasets, and it consistently gives extremely accurate answers. Refer to Fig. 2ii. We get $< 1\%$ relative error *for all subgraph counts on all datasets*, even those with more than 100M edges. (Exact counts were obtained by brute-force enumerations that took several days.) This is much more accurate than any existing method to count such motifs.

Provable guarantees with error bars. Our algorithm has a provable guarantee on accuracy and running time. Furthermore, we can quantify the accuracy/confidence on real inputs and runs of our algorithm. For a given number of samples, we can have a method to put an explicit error bar on our estimate, based on asymptotically tight versions of Chernoff's bound. While these error bars are not as tight as the real errors in Fig. 2ii, we can still mathematically prove that the errors are mostly within 5% and always within 10%.

Trends in 4-vertex pattern counts: Given the rapid reporting of 4-vertex pattern counts, our algorithm can be used as a tool for motif analysis. We detect common trends among a large variety of graphs. Not surprisingly, the 3-star is the most frequent 4-vertex motif in all graphs we experimented upon. The least frequent is either the 4-cycle or the 4-clique.

The chordal-4-cycle frequency is always more than that of the 4-cycle or 4-clique. Ugander et al [18] study what trends are merely implied by graph theory, and what are actually features of real-world graphs. Such analyses require accurate estimates quickly, which our algorithm can provide. It is a promising direction to use our algorithm to provide more input to such studies.

C. Comparison with previous work

We give a more detailed history of previous work in §III. Here, we highlight the difference from previous results on subgraph counting. The work (beyond triangle counting) involves color coding methods [15], [24], [27], MCMC based sampling algorithms [28], incremental pattern building algorithms for Map-Reduce [29], [30] edge sampling algorithms [26].

These methods are quite general and work for large subgraphs, and in that sense, are more general than our algorithm. But our focus on the specific six subgraphs in Fig. 1 allows for the design of highly accurate and fast algorithms, that work better than more generic methods. The differences are highlighted below.

Scalability and speed. Previous work either employ Map-Reduce clusters or max out at a million (or so) edges. Our algorithm runs on a single commodity machine with 64GB memory, and easily handles graphs with more than a hundred million edges. No previous result can get such scalability for 4-vertex pattern counting.

Accuracy. Our algorithm's accuracy is both empirical and provable. Previous methods [28] for motif analyses get the frequencies correct up to an order of magnitude. Our relative errors of 1% are much smaller than any of the previous results we are aware of. Furthermore, we can explicitly put realistic error bars on all our estimates. Again, this distinguishes our algorithm from the state-of-the-art.

Comprehensive results for 4-vertex patterns. We get detailed results for all 4-vertex counts on a large number of graphs, and believe this is useful for further data analysis (as done in [18]). Previous work usually focuses on a small, specific set of larger motifs [15], [24], [27], or gives coarser approximations for more motifs [28].

II. Formal description of the problem

Our input is an undirected simple graph $G = (V, E)$, with n vertices and m edges. For vertex v, d_v is the degree of v.

It is critical to distinguish subgraphs from *induced subgraphs*. A subgraph is simply some subset of edges. An induced subgraph is obtained by taking a subset V' of vertices, and consider *all edges* among these vertices. Refer to Fig. 3. The edges $(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1)$ form a 4-cycle, but the vertex set $\{v_1, v_2, v_3, v_4\}$ induces a chordal-4-cycle. We collectively refer to the 4-vertex subgraphs as "motifs".

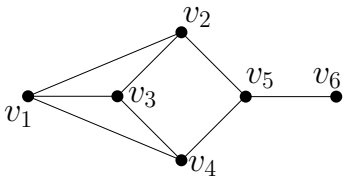


Fig. 3: An example graph.

It is technically convenient to think of induced subgraph counts. We denote the number of induced occurrences of the i-th subgraph (of Fig. 1) by C_i. So, C_4 is the number of induced 4-cycles in G, which is the number of distinct subsets of 4 vertices that induce a 4-cycle. When we talk of a "vanilla" subgraph, we mean the usual subgraph setting (a subset of edges). In general, if we do not say "induced", we mean vanilla.

Our aim is to get an estimate of all C_i values. Let N_i to denote the number of (vanilla) subgraph occurrences of the ith subgraph, There is a simple linear relationship between induced and non-induced counts, given below. The (i, j) entry of the matrix A below is simply the number of distinct copies of the ith subgraph in the jth subgraph (so $A_{2,4} = 4$, the number of 3-paths in the 4-cycle).

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 2 & 4 \\ 0 & 1 & 2 & 4 & 6 & 12 \\ 0 & 0 & 1 & 0 & 4 & 12 \\ 0 & 0 & 0 & 1 & 1 & 3 \\ 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix} = \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{pmatrix}. \quad (1)$$

III. Related Work

Motif counting for bioinformatics was arguably initiated by a seminal paper of Milo *et al.* [4]. This technique has been used for graph modeling [6], [15], graph comparisons [6], [31], and even decomposing a network [32]. Refer to [24], [25] for more details.

Triangle counting has a rich history in social sciences and related analyses, that we simply refer the reader to the related work sections of [22], [23]. The significance of 4-vertex patterns was studied in recent work of Ugander et al. [18], who propose a "coordinate system" for graphs based on the motifs distribution. This is used for improved network classification, and

the input graphs were comparatively small (thousands of vertices).

Previous studies tailored to 4-vertex patterns [33], [34] provide both exact and approximation algorithms. These are limited to small graphs. For example, a graph with 90K edges requires 40 minutes of processing [34] (on a single machine).

Most relevant to this work are previous studies on *wedge sampling* [19], [23], [35]. This method samples paths of length 2 to estimate various triangle statistics. Our method of 3-path sampling can be seen as building on wedge sampling. But we require new techniques of path pruning to get the algorithm to work accurately. These pruning techniques are inspired by degeneracy ordering algorithms for triangle counting [36], [37]. We can actually provide mathematical error bars for real runs and instances (as opposed to just a theoretical proof of convergence of estimate).

IV. The basic algorithm: estimating counts via 3-path sampling

Our algorithm for estimating counts of 4-vertex motifs is based on 3-path sampling. In this section, we discuss a basic version of this method. In the next section, we enhance it to get better accuracies

We begin with a simple procedure that samples a uniform (vanilla) random 3-path. For each edge $e = (u, v) \in E$, denote $\tau_e = (d_u - 1)(d_v - 1)$. We denote $W = \sum_e \tau_e$.

Algorithm 1: `sample`

1. Compute τ_e for all edges and set $p_e = \tau_e/W$.
2. Pick edge $e = (u, v)$ with probability p_e.
3. Pick uniform random neighbor u' of u *other than* v.
4. Pick uniform random neighbor v' of u *other than* u.
5. Output the three edges $\{(u', u), (u, v), (v, v')\}$.

Observe that the output of `sample` can either be a triangle (if $u' = v'$) or a 3-path. The following claim is critical.

Claim IV.1. *Fix any 3-path. The probability that* `sample` *outputs this 3-path is exactly* $1/W$.

Proof: Fix a 3-path $(u', u), (u, v), (v, v')$ (u, u', v, v' are all distinct). The probability that $e = (u, v)$ is selected as the middle edge (in Step 2) is exactly $(d_u - 1)(d_v - 1)/W$. Conditioned on this event, the probability that u' is selected as a neighbor of u is $1/(d_u - 1)$ (note that the neighbor v is excluded). Similarly, v' is selected with probability $1/(d_v - 1)$. Putting it all together, the 3-path is chosen with probability $[(d_u-1)(d_v-1)/W]\cdot[1/(d_u-1)]\cdot[1/(d_v-1)] = 1/W$. The probability is the same for all 3-paths, proving our claim. ∎

Now, observe that all motifs of Fig. 1, except the 3-star, contain a 3-path. So one can perform the following experiment. Run `sample` to get a collection of edges, and hence a set of (at most 4) vertices. Check the edges among these vertices to see what motif it induces. Repeat this experiment many times to estimate the true counts C_i ($i \in [2, 6]$). Finally, we use the formula of (1) to estimate C_1. This is exactly the algorithm `3-path-sampler`, as given in Alg. 2. We remind the reader that $A_{2,i}$ is the number of 3-paths in the ith motif.

Algorithm 2: `3-path-sampler`
Input: graph $G = (V, E)$, samples k

1. Run `sample` k times to get k sets of edges. Let S_ℓ denote the set of corresponding vertices for the ℓth set.
2. Initialize $count_i = 0$ for $i \in [2, 6]$.
3. For $\ell \in [1, k]$,
4. Determine subgraph induced by S_ℓ.
5. If this is the ith motif, increment $count_i$.
6. For each $i \in [2, 6]$,
7. Set $\widehat{C}_i = (count_i/k) \cdot (W/A_{2,i})$.
8. Set $N_1 = \sum_{v \in V} \binom{d_v}{3}$.
9. Set (induced 3-stars) $\widehat{C}_1 = N_1 - \widehat{C}_3 - 2\widehat{C}_5 - 4\widehat{C}_6$.

We prove that `3-path-sampler` outputs unbiased estimates for all C_is.

Theorem IV.2. *For every* $i \in [1, 6]$, $\mathbf{E}[\widehat{C}_i] = C_i$.

Proof: First, let use deal with subgraphs other than the 3-star, so fix some $i \neq 1$. For each $\ell \in [k]$, let X_ℓ be the indicator random variable for S_ℓ inducing the ith motif. So X_ℓ is 1 iff the ℓth call to `sample` outputs a 3-path contained in a copy of the ith motif. The total number of (distinct) such 3-paths is exactly $A_{2,i} \cdot C_i$. By Claim IV.1, the probability that $X_\ell = 1$ is $A_{2,i} \cdot C_i/W$. Hence, $\mathbf{E}[X_\ell] = C_i \cdot A_{2,i}/W$.

By linearity of expectation, $\mathbf{E}[count_i] = \sum_{\ell=1}^{k} \mathbf{E}[X_\ell] = (kC_iA_{2,i})/W$. Hence, $\mathbf{E}[\widehat{C}_i] = C_i$. Now, we detail with $\widehat{C}_1$. Note that N_1, the number of 3-star subgraphs, is exactly $\sum_{v \in V} \binom{d_v}{3}$. By linearity of expectation, $\mathbf{E}[\widehat{C}_1] = N_1 - \mathbf{E}[\widehat{C}_3] - 2\mathbf{E}[\widehat{C}_5] - 4\mathbf{E}[\widehat{C}_6]$, which is $N_1 - C_3 - 2C_5 - 4C_6 = C_1$ (as given by (1)). ∎

We can also prove concentration results using the Hoeffding bound [38]. This is useful as a proof of concept, but do not give useful bounds in practice. (We

give more details later.) This analysis is analogous to that of wedge sampling results [37], [23].

Theorem IV.3 (Hoeffding [38]). *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 $\mu = \mathbf{E}[\bar{X}]$. Then for $\varepsilon \in (0,1)$, we have*

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

We can derive concentration results quite directly from this bound.

Theorem IV.4. *Fix $\delta, \varepsilon \in (0,1)$ and set $k = \lceil (2\varepsilon)^{-2}\ln(2/\delta)\rceil$. For all $i \in [2,6]$: with probability at least $1-\delta$, $|\widehat{C}_i - C_i| < \varepsilon W/A_{2,i}$. With probability at least $1-\delta$, $|\widehat{C}_1 - C_1| < \varepsilon W$.*

Proof: We first fix $i \neq 1$. Define X_ℓ as in the previous proof. Apply Theorem IV.3 to $X_1, \ldots, X_k$. With probability at most δ, $|\bar{X} - \mathbf{E}[\bar{X}]| \geq \varepsilon$ (we use the notation from Theorem IV.3). It remains to interpret $\bar{X}$. Note that $\widehat{C}_i = (count_i/k)\cdot(W/A_{2,i})$. Since $count_i = \sum_{\ell=1}^{k} X_\ell$, $\widehat{C}_i = \bar{X}\cdot(W/A_{2,i})$. So $|\bar{X} - \mathbf{E}[\bar{X}]| \geq \varepsilon$ implies $|\widehat{C}_i - C_i| \geq \varepsilon$, as desired.

Since $\widehat{C}_1$ is obtained by subtracting out other terms, it appears that the errors could add up. With a little care, we can get the same bound as the other $\widehat{C}_i$'s. Define random variable Y_ℓ as follows: if S_ℓ induces a tailed triangle, $Y_\ell = 1/A_{2,3} = 1/2$. If S_ℓ induces a chordal-4-cyle, $Y_\ell = 2/A_{2,5} = 2/6$. If S_ℓ induces a 4-clique, $Y_\ell = 4/A_{2,6} = 4/12$. In all other cases, $Y_\ell = 0$. We have constructed this random variable, so that $\mathbf{E}[Y_\ell] = (C_3 + 2C_5 + 4C_6)/W$.

Observe that $\widehat{C}_1$ can also be expressed as $N_1 - (\sum_\ell Y_\ell/k)W$. The additive error $|\widehat{C}_1 - C_1|$ is the same as $W\cdot|\bar{Y} - \mathbf{E}[\bar{Y}]|$. Applying Theorem IV.3, with probability at least $1-\delta$, $|\widehat{C}_1 - C_1| < \varepsilon W$. ∎

A. The challenge of cycle-based motifs

Theorem IV.2 and Theorem IV.4 seem to give us all we want, so why aren't we done? The catch is that the concentration bound of Theorem IV.4 is actually too weak to give reasonable estimates for real world graphs. Let us do some rough calculations, ignoring the constants. To get an estimate such that $|\widehat{C}_i - C_i| < \varepsilon W$, we require $k \approx 1/\varepsilon^2$. But for such an estimate to be useful, we need to understand how W relates to C_i. So ε needs to be of the order of C_i/W, and consequently, k needs to be $(W/C_i)^2$.

Refer to Tab. I for the values of W and a few C_is. (For convenience, we just give the order of magnitude of each number. Full numbers are given later.) For $i \in \{1,2,3\}$ (3-star, 3-path, and tailed triangle), $(W/C_i)^2$ is usually $< 10^4$. This is fairly reasonable number of samples to take, and leads to an efficient and accurate algorithm. On the other hand, for $i \in \{4,5,6\}$ (4-cycle, chordal-4-cycle, and 4-clique), $(W/C_i)^2$ is often $> 10^8$, which is too many samples to take.

TABLE I: W vs C_i: counts given as orders of magnitude.

Graph	W	C_2	C_3	C_4	C_6
amazon0312	E+09	E+08	E+08	E+06	E+06
as-skitter	E+12	E+11	E+11	E+10	E+08
orkut	E+13	E+13	E+12	E+10	E+09

In other words, `3-path-sampler` does not perform well for motifs containing a 4-cycle. This leads us to a new algorithm for dealing with these motifs, as described in the next section.

V. Improved estimation of 4-cycle-based motifs via *centered* 3-paths

We denote the 4-cycle, chordal-4-cycle, and 4-clique as *cycle-based* motifs. We design a better algorithm to estimate them. While the algorithm is provably correct for any graph, the fact that it gives a significant improvement is dependent on the structure of real-world graphs.

Our aim is to find a subset $\mathcal{S}$ of 3-paths with the following properties:

- Every cycle-based motif is guaranteed to contain a fixed number of 3-paths from $\mathcal{S}$.
- It is possible to quickly generate uniform random samples from $\mathcal{S}$.
- $|\mathcal{S}|$ is significantly smaller than $W = \sum_{e=(u,v)}(d_u - 1)(d_v - 1)$.

Let us go back to `sample`, and think of enumerating all 3-paths. For edge (u,v), we take every neighbor of u and every neighbor of v to generate a 3-path. We basically take the Cartesian product of the adjacency lists of u and v. Could we prune the adjacency lists so this product is smaller?

Suppose we order all vertices based on degree and vertex id. So we say $u \prec v$ if: $d_u < d_v$ or, if $d_u = d_v$, the vertex id of u is less than that of v. We could prune the lists using this ordering. When looking for 3-paths where (u,v) is the middle edge, we only look at the portion of u's list "greater" than v, and the portion of v's list greater than u. In general, many 3-paths are generated when d_u and d_v are large. But in that case, we hope that many neighbors of u and v are of lower degree. The pruning ignores such vertices and (hopefully) reduces the set of 3-paths considered. Let us formalize the set $\mathcal{S}$ of centered 3-paths.

Definition V.1 (Centered 3-path)**.** *A 3-path formed by edges $\{(t,u),(u,v),(v,w)\}$ is* centered *if: $v \prec t$, $u \prec w$, and the edge (t,w) exists in the graph (so t,u,v,w form a 4-cycle).*

We prove the important property that every cycle-based motif contains a fixed number of centered 3-paths.

Lemma V.2. *Every induced 4-cycle and chordal-4-cycle contains exactly one centered 3-path. Every induced 4-clique contains exactly three centered 3-paths.*

Proof: Consider any (vanilla) 4-cycle, formed by vertices (in order) t,u,v,w. Pick the smallest vertex, say u. Pick the neighbor of u that is smaller, say v. We show that the 3-path $\{(t,u),(u,v),(v,w)\}$ is the only centered 3-path in this 4-cycle.

By the choice of (u,v), $v \prec t$ and $u \prec w$. Hence, $\{(t,u),(u,v),(v,w)\}$ is centered. The only other possible centered 3-path is $\{(u,t),(t,w),(w,v)\}$. Because $v \prec t$, this path cannot be centered. That completes the proof for the induced 4-cycle case.

Now, suppose t,u,v,w forms an induced chordal-4-cycle. The extra 3-paths contain the chord in the middle, and such 3-paths do not lie on a 4-cycle. So there only exists one centered 3-path.

A 4-clique contains three 4-cycles that partition the 12 different 3-paths. Each of these 4-cycles has a centered 3-path, yielding a total of three such 3-paths. ■

We now show how to sample a uniform random centered 3-path. It is quite analogous to `sample`. First, some notation. Let $L_{u,v}$ be the number of neighbors of u greater than v. By sorting all adjacency lists according to vertex degree and id, we can compute for every edge $e=(u,v)$, the value $\lambda_e = L_{u,v}L_{v,u}$. Let $\Lambda = \sum_e \lambda_e$.

Algorithm 3: `sample-centered`

1. Compute λ_e for all edges and set $p_e = \lambda_e/S$.
2. Pick edge $e=(u,v)$ with probability p_e.
3. Pick uniform random neighbor u' of u such that $v \prec u'$.
4. Pick uniform random neighbor v' of v such that $u \prec v'$.
5. Output the three edges $\{(u',u),(u,v),(v,v')\}$.

Note that it is possible that `sample-centered` outputs a 3-path that is not centered (if the 3-path does not lie on a 4-cycle). Nonetheless, analogous to Claim IV.1, we have the following.

Claim V.3. *Fix any centered 3-path. The probability that* `sample-centered` *outputs this 3-path is exactly* $1/\Lambda$.

Now, we give the algorithm that estimates the number of cycle-based motifs. It is analogous to `centered-sampler`, only using centered 3-paths. For convenience, let B_i denote the number of centered 3-paths in the ith motif. So $B_4 = B_5 = 1$ and $B_6 = 3$, by Lemma V.2.

Algorithm 4: `centered-sampler`
Input: graph $G=(V,E)$, samples k

1. Run `sample-centered` k times to get k set of edges. Let T_ℓ denote the set of corresponding edges for the ℓth set.
2. Initialize $count_i = 0$ for $i \in [4,6]$.
3. For $\ell \in [1,k]$,
4. If T_ℓ is a centered 3-path,
5. Determine subgraph induced by S_ℓ.
6. If this is the ith motif, increment $count_i$.
7. For each $i \in [4,6]$,
8. Set $\widehat{C}_i = (count_i/k)\cdot(\Lambda/B_i)$.

Analogous to Theorem IV.4, we can prove the following. Observe how W is replaced by Λ.

Theorem V.4. *Fix $\delta,\varepsilon \in (0,1)$ and set $k = \lceil (2\varepsilon)^{-2}\ln(2/\delta)\rceil$. For all $i \in [4,6]$: with probability at least $1-\delta$, $|\widehat{C}_i - C_i| < \varepsilon\Lambda/B_i$.*

A. Why centered 3-paths help

We put the value of W and Λ for various real world networks in Tab. II. Observe how Λ is at least an order of magnitude smaller than W (except for a road network). This is a huge difference when it comes to the sampling bounds in Theorem IV.4 and Theorem V.4. These bounds show that *two orders of magnitude less samples* suffice for the same error (in estimating cycle-based motifs). This improvement is extremely significant for getting good accuracy with fewer samples.

TABLE II: Difference between the number of 3-paths and the number of centered 3-paths.

Graph	W	Λ	W/Λ
amazon0312	1.40E+09	9.36E+07	15
amazon0505	1.59E+09	1.02E+08	16
amazon0601	1.57E+09	1.01E+08	15
as-skitter	1.43E+12	9.05E+10	16
cit-Patents	9.16E+09	8.78E+08	10
web-BerkStan	1.69E+12	1.28E+11	13
web-Google	2.05E+10	6.34E+08	32
web-Stanford	1.85E+11	1.36E+10	14
wiki-Talk	1.31E+12	9.08E+09	144
youtube	1.19E+11	1.68E+09	71
flickr	1.31E+13	8.42E+11	16
livejournal	1.67E+12	1.14E+11	15
orkut	2.22E+13	9.48E+11	23

The final algorithm is simply obtained by running both `3-path-sampler` and `centered-sampler`.

The former gives estimates for C_1, C_2, C_3 (we simply discard the remaining estimates), and the latter estimates C_4, C_5, C_6.

VI. Getting practical error bars

While the Hoeffding bound used above provides theoretical convergence, we do not get practical error bars from it. In this section, we show how to get useful error bars for our algorithm on real instances.

All of our sampling algorithms have the same underlying primitive: try to estimate the expectation p of a Bernoulli random variable. We generate a binomial random variable $X \sim B(k,p)$ (by performing k i.i.d. Bernoulli trials), and hope that the outcome is close enough to the expectation.

We employ a standard Bayesian viewpoint to generate an error bar. Suppose, our outcome of the binomial draw is $X = r$. Conditioned on a choice of p, we calculate the probability that $X = r$. This gives a prior on p. Of course, this cannot be done explicitly because of computational issues, but we can use tail bounds for $B(k,p)$ to get appropriate estimates. We use the following theorem of Chernoff [39] (we use notation of Equation 1.4 from [40]) that gives good tail bounds for $B(k,p)$.

Theorem VI.1 (Chernoff)**.** *Suppose $X \sim B(k,p)$. Fix $\alpha \in (0,1)$.*

$$\Pr[X/k \geq \alpha] \leq \exp(-D(\alpha,p)k) \quad \text{if } \alpha > p$$
$$\Pr[X/k \leq \alpha] \leq \exp(-D(\alpha,p)k) \quad \text{if } \alpha < p$$

where $D(a,b) = a\ln(a/b) + (1-a)\ln((1-a)/(1-b))$ (the KL-divergence between Bernoulli distributions with expectation a and b).

Suppose the outcome of $X/k = \alpha$. We can use the Chernoff bound to get a range of likely values of p. Think of $\exp(-D(\alpha,p)k)$ as a function of p. The basic properties of the KL-divergence (and simple algebra) imply that $\exp(-D(\alpha,p)k)$ is a unimodel function with a maximum value of 1 at $p = \alpha$ and a minimum of 0 at $p = 0, 1$. That motivates the following definition.

Definition VI.2. *Fix $k, \alpha, x \in (0,1)$. Then $p_l(k,\alpha,x)$ (lower) and $p_u(\alpha,x)$ (upper) are the two unique values of p such that $\exp(-D(\alpha,p)k) = x$.*

With this definition, we can give precise error bars. In other words, given the outcome of a binomial random variable $B(k,p)$, we can give an interval of plausible values (up to any desired confidence δ) for p.

Corollary VI.3 (Error bar for binomial distribution)**.** *Fix binomial distribution $B(k,p)$, and $\alpha, \delta \in (0,1)$ Then, for any $p \notin [p_l(k,\alpha,\delta), p_u(k,\alpha,\delta)]$,*

$$\Pr_{X \sim B(k,p)}[X/k = \alpha] \leq \delta$$

How does this relate to our algorithms? Observe that in both Alg. 2 and Alg. 4, the variables $count_i$ are binomial random variables. So we can produce errors bars for $count_i/k$ using the above corollary. The final estimates are of the form $\widehat{C}_i = (count_i/k) \cdot K_i$ ($i \neq 1$, and K_i is some fixed scaling, depending on the algorithm and i). So error bars for $count_i/k$ directly translate to error bars for $\widehat{C}_i$. For $i = 1$ (3-stars), we simply add up the errors (in `3-path-sampler`) for $\widehat{C}_3$, $2\widehat{C}_5$, and $4\widehat{C}_6$.

VII. Experimental Results

Preliminaries: We implemented our algorithms in `C` and ran our experiments on a computer equipped with a 2x2.4GHz Intel Xeon processor with 6 cores and 256KB L2 cache (per core), 12MB L3 cache, and 64GB memory. We performed our experiments on 13 graphs from SNAP [41] and per private communication with the authors of [42]. In all cases, directionality is ignored, and duplicate edges are omitted. The properties of these matrices are presented in Tab. III, where $|V|$ and $|E|$ are the numbers of vertices and edges, respectively.

Exact counts for the motifs are obtained by a well-tuned enumeration (counts and runtime given in Tab. III). We do not get into details, but note that the enumeration code processes million edge Amazon networks in only 5 seconds[1].

For getting 3-path sampling estimates, we run both `3-path-sampler` and `centered-sampler` as described earlier, with $k = 200K$. We use the outputs of $\widehat{C}_1, \widehat{C}_2, \widehat{C}_3$ as given by `3-path-sampler`, and $\widehat{C}_4, \widehat{C}_5, \widehat{C}_6$ from `centered-sampler`. The runtimes are in the last column of Tab. III.

Convergence of estimates: To show convergence, we perform detailed runs on the as-skitter graph. We choose this because it is the most difficult to get accurate estimates, since the cycle-based motif counts are small relative to the graph size. We vary the numbers of samples in increments of 2.5K. For each choice of the number of samples, we perform 50 runs of our algorithm. We plot those results in Fig. 4 for tailed-triangles, chordal-4-cycles, and 4-cliques. (Other patterns are omitted due the space considerations, and had even better convergence.) The output of each run (for a given number of samples) is depicted by a blue dot. For 4-clique counts, we can see the spread of outputs

[1]This is quite competitive with the best existing numbers in the literature of [34], whose algorithm takes 40 minutes on a 90K autonomous systems graph.

TABLE III: Exact values of pattern counts and runtimes (in seconds).

Datasets	$\|V\|$	$\|E\|$	3-star	3-path	Tailed triangle	4-cycle	Chordal 4-cycle	4-clique	Enum. time	3-path samp. time
amazon0312	4.01E+5	2.35E+6	1.07E+10	8.44E+08	1.90E+08	3.23E+06	1.71E+07	3.98E+06	4.42	0.47
amazon0505	4.10E+5	2.44E+6	1.21E+10	9.63E+08	2.19E+08	3.30E+06	1.91E+07	4.36E+06	4.75	0.48
amazon0601	4.03E+5	2.44E+6	1.11E+10	9.41E+08	2.17E+08	3.22E+06	1.92E+07	4.42E+06	4.74	0.48
as-skitter	1.70E+6	1.11E+7	9.64E+13	8.19E+11	1.62E+11	4.27E+10	1.96E+10	1.49E+08	5128.93	2.7
cit-Patents	3.77E+6	1.65E+7	6.11E+9	6.54E+09	5.52E+08	2.69E+08	6.28E+07	3.50E+06	46.46	3.33
flickr	1.86E+6	1.56E+7	1.90E+13	6.89E+12	1.18E+11	1.18E+11	2.30E+11	2.67E+10	217274.39	2.53
livejournal	5.28E+6	4.87E+7	4.46E+12	1.14E+12	1.26E+11	5.21E+09	1.90E+10	1.14E+10	11894.63	6.86
orkut	3.07E+6	2.24E+8	9.78E+13	1.86E+13	1.51E+12	7.01E+10	4.78E+10	3.22E+09	70966.96	16.24
web-BerkStan	6.85E+5	6.65E+6	3.82E+14	3.14E+10	4.76E+11	2.53E+10	9.86E+10	1.07E+09	6462.56	3.77
web-Google	8.76E+5	4.32E+6	6.50E+11	4.06E+09	6.72E+09	3.80E+07	3.82E+08	3.99E+07	52.29	0.88
web-Stanford	2.82E+5	1.99E+6	2.51E+13	1.28E+10	5.08E+10	4.48E+09	8.60E+09	7.88E+07	831.5	1.76
wiki-Talk	2.39E+6	4.66E+6	1.92E+14	1.17E+12	6.41E+10	9.24E+08	1.03E+09	6.49E+07	1346.76	1.04
youtube	1.16E+6	4.95E+6	5.73E+12	9.15E+10	1.24E+10	2.32E+08	2.22E+08	4.99E+06	141.78	0.61

reducing. The figure only goes up to 35K samples. (The convergence is so rapid that at around 50K samples, the spread is impossible to see.)

Accuracy: Fig. 2ii presents the relative errors for all 13 graphs and all 6 motifs, using 200K samples for both `3-path-sampler` and `centered-sampler`. *All relative errors are less than 1% in all instances.* As expected the relative errors tend to be larger for the less frequent patterns such as 4-cycles and 4-cliques.

Speedup: Fig. 2i presented the speedups achieved over full enumeration by using our path sampling algorithm. Enumeration for flickr and orkut takes order of a day. Since the motifs counts are in the order of tens of billions, there is no hope of getting any scalability. Our algorithms takes less than a minute (even including I/O) for all these graphs.

The benefit of centered 3-paths: We could simply use the basic 3-path sampling given in `3-path-sampler` to approximate all counts. We compare this approach to our final algorithm that use `centered-sampler` for C_4, C_5, C_6 estimates. Comparisons between the relative errors for C_4, C_5, C_6 are given in Fig. 5. ("Basic" denotes simply using `3-path-sampler`, and "centered" is the main algorithm.) We used 200K samples for both algorithms. Some instances of using `3-path-sampler` give somewhat large errors, and `centered-sampler` really cuts these errors down. It shows the power of centered 3-path sampling.

Error bounds: We use Corollary VI.3 (as explained in §VI) to compute 99% confidence error bounds for all of our runs. So, for a single run of our algorithm on a candidate graph, we have a mathematical bound on the error that is solely based on output estimates. (These are critical in the situation where we do not know the true answer, and need confidence that the estimates are accurate.) Fig. 6 shows the accuracy of our error bounds with 99% confidence, so $\delta = 0.01$ in Corollary VI.3. In all cases, the provable bounds on the error are always

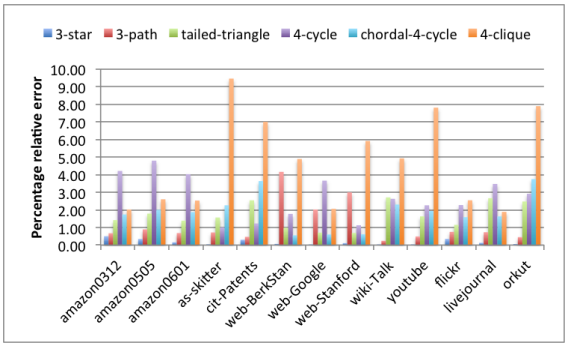
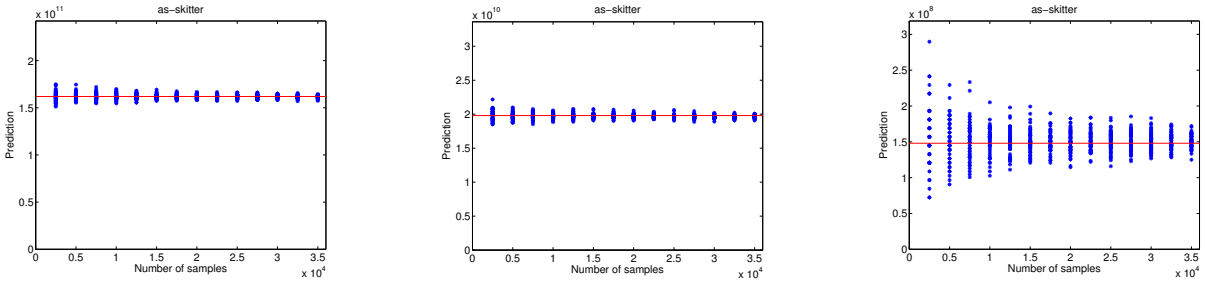


Fig. 6: Provable error bounds

less than 10% and mostly at most 5%. (We stress that the actual error is much smaller.) To the best of our knowledge, no previous sampling based algorithm for motif counting comes with hard mathematical error bars that are practically reasonable.

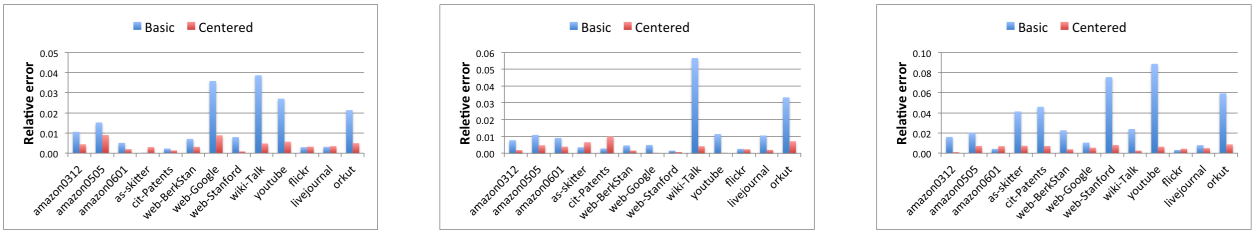
Edge sampling approaches: An alternative approach to motif counting is *edge sampling*, which was applied to triangle counting [22], [21]. In this approach, the graph is sparsified by retaining each edge independently with a fixed probability p. The motif counts is obtained in the sparsified graph, and scaled to give an unbiased estimate of the true count (For example, the 4-clique count of the sparsified graph is multiplied by p^{-6}). Because of limited space, we only provide comparisons between this approach and our algorithm for 4-clique counts. Naturally, as p decreases, the runtime decreases but so does the accuracy.

In Fig. 7, we show results for the edge sampling approach for $p = 0.05, 0.1, 0.2$. To compare with our algorithm (200K samples), we plot all results with time on the x-axis. Again, we take 50 runs of for all settings, and the output is given by a single dot (red for edge sampling, blue for path sampling). While $p = 0.05$ gives better runtimes, the estimates are completely wrong.



(i) tailed-triangle (ii) chordal-4-cycle (iii) 4-clique

Fig. 4: Increasing number of samples decreases error: Each blue dot is an output of a run of the algorithm with the number of samples in the x-axis. The red line is the true count.



(i) 4-cycle (ii) Chordal 4-cycle (iii) 4-clique

Fig. 5: Comparing the accuracy of estimations using basic 3-path sampling and centered 3-path sampling

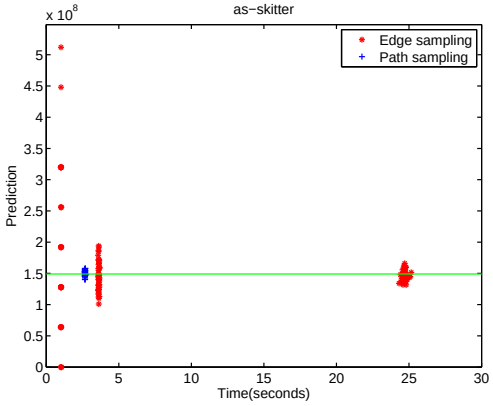


Fig. 7: Comparing edge sampling and 3-path sampling

For comparable in accuracy to our 3-path sampling algorithm, we need to set $p = 0.2$ for edge sampling (which then has a run time about 10 times more).

Trends in patterns: For lack of space, we do not get detailed studies of the relationships among the induced counts. We mention a few direct observations. The most frequent connected induced motif is the 3-star. The least frequent is either the 4-cycle or the 4-clique. We find it somewhat interesting that (among cycle-based motifs) the chordal-4-cycle is the most frequent. (The orkut graph is a notable exception in that 4-cycles are more frequent.) We find these counts fascinating, and a future direction is to connect these counts with the subgraph frequency approaches of [18].

VIII. Conclusions and Future Work

In this work we showed a 3-path sampling scheme that efficiently and accurately estimates counts of all 4-vertex motifs in large input graphs. It is natural to ask if we can extend this sampling scheme further to estimate counts of 5-vertex (or even higher order) motifs.

Acknowledgment

This work was funded by the DARPA GRAPHS 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.

REFERENCES

[1] P. Holland and S. Leinhardt, "A method for detecting structure in sociometric data," *American Journal of Sociology*, vol. 76, pp. 492–513, 1970. 1

[2] J. Coleman, "Social capital in the creation of human capital," *American Journal of Sociology*, vol. 94, pp. S95–S120, 1988. [Online]. Available: http://www.jstor.org/stable/2780243 1

[3] A. Portes, "Social capital: Its origins and applications in modern sociology," *Annual Review of Sociology*, vol. 24, no. 1, pp. 1–24, 1998. 1

[4] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: Simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002. 1, 3

[5] R. Burt, "Structural holes and good ideas," *American Journal of Sociology*, vol. 110, no. 2, pp. 349–399, 2004. [Online]. Available: http://www.jstor.org/stable/10.1086/421787 1

[6] N. Przulj, D. G. Corneil, and I. Jurisica, "Modeling interactome: scale-free or geometric?." *Bioinformatics*, vol. 20, no. 18, pp. 3508–3515, 2004. [Online]. Available: http://dblp.uni-trier.de/db/journals/bioinformatics/bioinformatics20.html#PrzuljCJ04 1, 3

[7] G. Fagiolo, "Clustering in complex directed networks," *Phys. Rev. E*, vol. 76, p. 026107, Aug 2007. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.76.026107 1

[8] F. Hormozdiari, P. Berenbrink, N. Prulj, and S. C. Sahinalp, "Not all scale-free networks are born equal: The role of the seed graph in ppi network evolution," *PLoS Computational Biology*, vol. 118, 2007. 1

[9] 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. 1

[10] K. Faust, "A puzzle concerning triads in social networks: Graph constraints and the triad census," *Social Networks*, vol. 32, no. 3, pp. 221–233, 2010. 1

[11] M. Szell and S. Thurner, "Measuring social dynamics in a massive multiplayer online game," *Social Networks*, vol. 32, pp. 313–329, 2010. 1

[12] B. 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. 1

[13] S. Son, A. Kang, H. Kim, T. Kwon, J. Park, and H. Kim, "Analysis of context dependence in social interaction networks of a massively multiplayer online role-playing game," *PLoS ONE*, vol. 7, no. 4, p. e33918, 04 2012. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0033918 1

[14] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998. 1

[15] F. Hormozdiari, P. Berenbrink, N. Przulj, and S. C. Sahinalp, "Not all scale-free networks are born equal: The role of the seed graph in ppi network evolution." *PLoS Computational Biology*, vol. 3, no. 7, 2007. [Online]. Available: http://dblp.uni-trier.de/db/journals/ploscb/ploscb3.html#HormozdiariBPS07 1, 3

[16] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of Erdös-Rényi graphs," *Physical Review E*, vol. 85, no. 5, p. 056109, May 2012. 1

[17] N. Durak, A. Pinar, T. G. Kolda, and C. Seshadhri, "Degree relations of triangles in real-world networks and graph models," in *CIKM'12*, 2012. 1

[18] J. Ugander, L. Backstrom, and J. M. Kleinberg, "Subgraph frequencies: mapping the empirical and extremal geography of large graph collections," in *WWW*, D. Schwabe, V. A. F. Almeida, H. Glaser, R. A. Baeza-Yates, and S. B. Moon, Eds. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 1307–1318. 1, 2, 3, 9

[19] T. Schank and D. Wagner, "Approximating clustering coefficient and transitivity," *Journal of Graph Algorithms and Applications*, vol. 9, pp. 265–275, 2005. 1, 4

[20] 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. 1

[21] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos, "Doulion: counting triangles in massive graphs with a coin," in *Knowledge Data and Discovery (KDD)*, 2009, pp. 837–846. 1, 8

[22] C. Tsourakakis, M. N. Kolountzakis, and G. Miller, "Triangle sparsifiers," *J. Graph Algorithms and Applications*, vol. 15, pp. 703–726, 2011. 1, 3, 8

[23] C. Seshadhri, A. Pinar, and T. G. Kolda, "Fast triangle counting through wedge sampling," in *Proceedings of the SIAM Conference on Data Mining*, 2013. [Online]. Available: http://arxiv.org/abs/1202.5230 1, 3, 4, 5

[24] N. Betzler, R. van Bevern, M. R. Fellows, C. Komusiewicz, and R. Niedermeier, "Parameterized algorithmics for finding connected motifs in biological networks," *IEEE/ACM Trans. Comput. Biology Bioinform.*, vol. 8, no. 5, pp. 1296–1308, 2011. 1, 3

[25] E. Wong, B. Baur, S. Quader, and C.-H. Huang, "Biological network motif detection: principles and practice," *Briefings in Bioinformatics*, vol. 13, no. 2, pp. 202–215, 2012. 1, 3

[26] M. Rahman, M. A. Bhuiyan, and M. A. Hasan, "Graft: An efficient graphlet counting method for large graph analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, no. 99, 2014. 1, 3

[27] Z. Zhao, G. Wang, A. Butt, M. Khan, V. S. A. Kumar, and M. Marathe, "Sahad: Subgraph analysis in massive networks using hadoop," in *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS)*, 2012, pp. 390–401. 3

[28] M. Bhuiyan, M. Rahman, M. Rahman, and M. A. Hasan, "Guise: Uniform sampling of graphlets for large graph analysis," in *Proceedings of International Conference on Data Mining*, 2012, pp. 91–100. 3

[29] T. Plantenga, "Inexact subgraph isomorphism in mapreduce," *Journal of Parallel and Distributed Computing*, no. 0, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731512002559 3

[30] M. A. Bhuiyan and M. A. Hasan, "Mirage: An iterative mapreduce based frequent subgraph mining algorithm," arXiv, Tech. Rep., 2013, http://arxiv.org/pdf/1307.5894.pdf. 3

[31] D. Hales and S. Arteconi, "Motifs in evolving cooperative networks look like protein structure networks." *NHM*, vol. 3, no. 2, pp. 239–249, 2008. [Online]. Available: http://dblp.uni-trier.de/db/journals/nhm/nhm3.html#HalesA08 3

[32] S. Itzkovitz, R. Levitt, N. Kashtan, R. Milo, M. Itzkovitz, and U. Alon, "Coarse-graining and self-dissimilarity of complex networks," vol. 71, no. 016127, January 2005. 3

[33] M. Gonen and Y. Shavitt, "Approximating the number of network motifs," *Internet Mathematics*, vol. 6, no. 3, pp. 349–372, 2009. 4

[34] D. Marcus and Y. Shavitt, "Efficient counting of network motifs," in *ICDCS Workshops*. IEEE Computer Society, 2010, pp. 92–98. 4, 7

[35] T. G. Kolda, A. Pinar, T. Plantenga, C. Seshadhri, and C. Task, "Counting triangles in massive graphs with MapReduce," *SIAM Journal of Scientific Computing*, 2013, to appear. 4

[36] N. Chiba and T. Nishizeki, "Arboricity and subgraph listing algorithms," *SIAM J. Comput.*, vol. 14, pp. 210–223, 1985. [Online]. Available: http://dx.doi.org/10.1137/0214017 4

[37] T. Schank and D. Wagner, "Finding, counting and listing all triangles in large graphs, an experimental study," in *Experimental and Efficient Algorithms*. Springer Berlin / Heidelberg, 2005, pp. 606–609. 4, 5

[38] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. American Statistical Association*, vol. 58, pp. 13–30, 1963. [Online]. Available: http://www.jstor.org/stable/2282952 4, 5

[39] H. Chernoff, "A measure of asymptotic efficiency for test of a hypothesis based on the sum of observations," *Annals of mathematical statistics*, vol. 23, pp. 493–507, 1952. 7

[40] D. Dubhashi and A. Panconesi, *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009. 7

[41] "Stanford Network Analysis Project (SNAP)," available at http://snap.stanford.edu/. 7

[42] 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. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298311 7