MOSS-5: A Fast Method of Approximating Counts of 5-Node Graphlets in Large Graphs

Pinghui Wang, Junzhou Zhao, Xiangliang Zhang, Zhenguo Li, Jiefeng Cheng, John C.S. Lui, Don Towsley, Jing Tao, and Xiaohong Guan

Abstract—Counting 3-, 4-, and 5-node graphlets in graphs is important for graph mining applications such as discovering abnormal/evolution patterns in social and biology networks. In addition, it is recently widely used for computing similarities between graphs and graph classification applications such as protein function prediction and malware detection. However, it is challenging to compute these metrics for a large graph or a large set of graphs due to the combinatorial nature of the problem. Despite recent efforts in counting triangles (a 3-node graphlet) and 4-node graphlets, little attention has been paid to characterizing 5-node graphlets. In this paper, we develop a computationally efficient sampling method to estimate 5-node graphlet counts. We not only provide fast sampling methods and unbiased estimators of graphlet counts, but also derive simple yet exact formulas for the variances of the estimators which is of great value in practice—the variances can be used to bound the estimates’ errors and determine the smallest necessary sampling budget for a desired accuracy. We conduct experiments on a variety of real-world datasets, and the results show that our method is several orders of magnitude faster than the state-of-the-art methods with the same accuracy.

Index Terms—graphlet kernel, subgraph sampling, graph mining.

1 INTRODUCTION

For complex networks such as online social networks (OSNs), computer networks, and biological networks, designing tools for estimating the counts (or frequencies) of 3-, 4-, and 5-node connected subgraph patterns (i.e., graphlets) shown in Fig. 1 is fundamental for detecting evolution and anomaly patterns in a large graph and computing graph similarities for graph classification, which have been widely used for a variety of graph mining and learning tasks. To explore patterns in a large graph, Milo et al. [1] defined network motifs as graphlets occurring in networks at numbers that are significantly larger than those found in random networks. Network motifs have been used for pattern recognition in gene expression profiling [2], evolution patterns in OSNs [3]–[6], and Internet traffic classification and anomaly detection [7,8]. In addition to mining a single large graph, graphlet counts also have been used to classify a large number of graphs. The graphlet kernel [9] (the dot product of two vectors of normalized graphlet counts) and RGF-distance [10] (Euclidean distance between two vectors of normalized graphlet counts) are widely used for graph similarity comparison, which is an important problem in application areas as disparate as bioinformatics, chemoinformatics, and software engineering. For example, 1) protein function prediction: identifying whether a given protein is an enzyme is important for understanding its function in biology. The biological network of a protein is usually represented as a undirected graph where a node in the graph represents an atom and an edge represents the existence of a chemical bond (i.e., a lasting attraction) between two atoms. Thus, one can infer whether a given protein is an enzyme or not by computing the similarities between the graph topologies of the protein and a large set of enzymes given in advance [11,12]; 2) compound function prediction. Similarly, chemical compounds are usually represented as a graph, and computing the similarity between them is important for applications such as predicting activity or adverse effects of potential drugs [13,14]; 3) node and community clustering. In addition to biological and chemical applications, Yanardag and Vishwanathan [15] reveal that computing similarities between the ego-networks of nodes (e.g., researchers in coauthor networks) in other networks such as coauthor networks and OSNs is useful for predicting the node classes (e.g., the field of researchers). Similarly, they represent each online discussion thread on OSN Reddit1 as a graph where nodes correspond to users and there is an edge between two nodes if at least one of them responded to another’s comment. Yanardag and Vishwanathan [15] observe that computing the similarities between these graphs is effective for the task of identifying whether a given graph belongs to a question/answer-based community or a discussion-based community; 4) malware detection. Attackers currently use two effective and convenient ways to generate and distribute attack payloads: (a) reuse existing malicious code to generate new malware variants, (b) use repackaging techniques to inject a small piece of malicious...

1. http://www.reddit.com
code into popular mobile Apps such as Angry Bird. Meanwhile, they can easily avoid traditional detectors based on pure syntax. Recently, [16]–[18] observe that the malwares generated by the above two ways keep a large fraction of relationships between subroutines and classes in the original computer programs, which can be recovered from disassembly of their executable binaries (software reverse engineering). They define graphs (e.g., view graph in [16], component graph in [17], and call graph in [18]) to depict the relationships between subroutines and classes in softwares, therefore comparing topology similarities between graphs is useful for detecting the above malwares.

Figure 1. 3-, 4-, and 5-node undirected graphlets. Combinatorial explosion: A node $v$ with degree $d_v$ in the graph of interest is included in at least $\frac{1}{2}d_v(d_v - 1)$ 3-node CISes, $\frac{1}{2}d_v(d_v - 1)(d_v - 2)$ 4-node CISes, and $\frac{1}{2}d_v(d_v - 1)(d_v - 2)(d_v - 3)$ 5-node CISes.

![Diagram of 3-node, 4-node, and 5-node graphlets](image)

Due to the combinatorial explosion of the problem, it is computationally intensive to enumerate and compute graphlet counts even for a moderately sized graph. For example, for two medium-size networks Slashdot [19] and Epinions [20] that each only contains $10^5$ nodes and $10^6$ edges, each has more than $10^{10}$ 4-node connected and induced subgraphs (CISes) and $10^{13}$ 5-node CISes. To address this problem, approximate methods such as sampling could be used in place of the brute-force enumeration approach. As shown in Fig. 2 (the graphical user interface of our system), a practical sampling method desires an important feature that it can stop its sampling procedure as soon as it achieves a desired estimation accuracy or when the sampling budget runs out, and then returns 1) graphlet count estimates and 2) estimation errors.

Despite recent progress in counting triangles [21]–[26] and 4-node graphlets [27], little attention has been given to developing fast tools for characterizing and counting 5-node graphlets. Recently, Pinar et al. [28] give a fast method ESCAPE for counting 5-node undirected graphlets by utilizing the relationships between 3-, 4-, and 5-node graphlets counts. However, ESCAPE is not scalable to large graphs, which requires more than 10 hours to handle graphs with millions of nodes and edges. To address this challenge, in this paper we propose a novel sampling method MOSS-5 to estimate the counts of 5-node graphlets. MOSS-5 consists of two sub-methods: T-5 and Path-5, which are customized to fast sample 5-node CISes in two specific graphlet groups respectively. Based on the samples of T-5 and Path-5, we estimate all 5-node graphlet counts and bound the estimates’ errors. Our contributions are summarized as:

- Our method for sampling 5-node CISes and estimating 5-node graphlet counts are scalable and computationally efficient.
- To validate our method, we perform an in-depth analysis of our method’s accuracy. We find that our method provides unbiased estimates of 5-node graphlet counts. We also derive simple and exact formulas for the variances of the estimators, which is critical in practice such as bounding the estimates’ errors and determining a proper sampling budget to achieve a desired accuracy. This has been lacking for previous estimators.
- We conduct experiments on a variety of publicly available datasets, and experimental results demonstrate that our method...
significantly outperforms the state-of-the-art methods. To guarantee the reproducibility of the experimental results, we release the source code of MOSS-5 in open source.

The rest of this paper is organized as follows. The problem formulation is presented in Section 2. Section 3 presents our 5-node graphlet sampling method MOSS-5. The performance evaluation and testing results are presented in Section 4. Section 5 summarizes related work. Concluding remarks then follow.

2 Problem Formulation

Let \( G = (V, E) \) be an undirected graph, where \( V \) and \( E \) are the sets of nodes and edges respectively. To define graphlet counts of \( G \), let us first introduce some notation. A subgraph \( G' \) of \( G \) is a graph whose set of nodes and set of edges are both subsets of \( G \). An induced subgraph of \( G \), \( G' = (V', E') \), is a subgraph that consists of some nodes of \( G \) and all of the edges that connect these nodes in \( G \), i.e. \( V' \subseteq V, E' = \{(u, v) : u, v \in V', (u, v) \in E\} \).

Until we explicitly say “induced” in this paper, otherwise a subgraph is not necessarily induced. All undirected graphs’ 5-node graphlets \( G_1^{(5)}, \ldots, G_{21}^{(5)} \) studied in this paper are shown in Fig. 1. Denote by \( C_i^{(5)} \) the set of 5-node CISes in \( G \), and \( C_i \) the set of 5-node CISes in \( G \) isomorphic to graphlet \( G_i \). The graphlet count of \( G_i^{(5)} \) is defined as \( \eta_i = |C_i^{(5)}|, 1 \leq i \leq 21 \). As discussed above, it is computationally expensive to enumerate and count all 5-node CISes in large graphs. In this paper, we develop a fast sampling method to accurately estimate 5-node graphlet counts \( \eta_1, \ldots, \eta_{21} \). For ease of reading, we list notation used throughout the paper in Table 1.

### Table 1

| \( G = (V, E) \) | is an undirected graph
| \( N_v \) | the set of neighbors of a node \( v \) in \( G \)
| \( d_v \) | \( d_v = |N_v| \), the cardinality of set \( N_v \)
| \( G_1^{(5)}, \ldots, G_{21}^{(5)} \) | 5-node graphlets
| \( C_i^{(5)} \) | the set of all 5-node CISes in \( G \)
| \( C_i^{(5)}, 1 \leq i \leq 21 \) | is the set of 5-node CISes in \( G \) isomorphic to graphlet \( G_i^{(5)} \)
| \( \eta_1, \ldots, \eta_{21} \) | \( \eta_i = |C_i|, 1 \leq i \leq 21 \)
| \( K_1, K_2 \) | sampling budgets of T-5 and Path-5
| \( K = K_1 + K_2 \) | sampling budgets of MOSS-5
| \( \phi_1^{(1)}, 1 \leq i \leq 21 \) | the number of subgraphs in a CIS \( s \in C_i^{(5)} \) that are isomorphic to \( G_i^{(5)} \)
| \( \phi_2^{(1)}, 1 \leq i \leq 21 \) | the number of subgraphs in a CIS \( s \in C_i^{(5)} \) that are isomorphic to \( G_i^{(5)} \)
| \( \phi_3^{(1)}, 1 \leq i \leq 21 \) | the number of subgraphs in a CIS \( s \in C_i^{(5)} \) that are isomorphic to \( G_i^{(5)} \)

3 Our Method of Estimating 5-NODE Undirected Graphlet Counts

In this section, we introduce our method MOSS-5. We observe that 1) except CISes in \( C_1^{(5)} \cup C_2^{(5)} \cup C_6^{(5)} \), 5-node CISes include


3. Two graphs \( G_1 = (V_1, E_1) \) and \( G_2 = (V_2, E_2) \) are said to be isomorphic if there exists at least one bijection \( f : V_1 \rightarrow V_2 \) such that any two nodes \( u \) and \( v \) in \( V_1 \) are adjacent in \( G_1 \) if and only if \( f(u) \) and \( f(v) \) are adjacent in \( G_2 \).

at least one subgraph isomorphic to graphlet \( G_i^{(5)} \), 2) except CISes in \( C_2^{(5)} \cup C_3^{(5)} \cup C_6^{(5)} \), 5-node CISes include at least one subgraph isomorphic to graphlet \( G_1^{(5)} \). Let \( \Omega_1 = \{1, \ldots, 21\} \setminus \{1, 2, 6\} \) and \( \Omega_2 = \{1, \ldots, 21\} \setminus \{2, 3, 8\} \). Inspired by the above two observations, as shown in Fig. 3, we develop a method MOSS-5 consisting of two sub-methods: T-5 and Path-5, where T-5 is customized to fast sample 5-node CISes isomorphic to \( G_i^{(5)}, i \in \Omega_1 \), and Path-5 is customized to fast sample 5-node CISes isomorphic to \( G_j^{(5)}, j \in \Omega_2 \). For any \( i \in \Omega_1 \), we provide an unbiased estimate \( \hat{\eta}_i^{(1)} \) of \( \eta_i \) based on sampled CISes of T-5, and derive a closed-form formula of the variance of \( \hat{\eta}_i^{(1)} \). For any \( j \in \Omega_2 \), similarly, we provide an unbiased estimate \( \hat{\eta}_j^{(2)} \) of \( \eta_j \) based on sampled CISes of Path-5, and derive a closed-form formula of the variance of \( \hat{\eta}_j^{(2)} \). Based on \( \hat{\eta}_i^{(1)} \) and \( \hat{\eta}_j^{(2)} \), we propose a more accurate estimator \( \hat{\eta}_k \), \( k \in \Omega_1 \cup \Omega_2 = \{1, \ldots, 21\} \setminus \{2\} \) and provide an unbiased estimator \( \hat{\eta}_k \) of \( \eta_k \). To bound the error of \( \hat{\eta}_k \), \( k \in \{1, \ldots, 21\} \), we also derive the variance of each \( \hat{\eta}_k \).

### Figure 3

Overview of MOSS-5. MOSS-5 smartly combines two T-5 and Path-5 sampling methods, which are two novel sampling methods proposed in this paper.

### 3.1 The T-5 Sampling Method

Denote

\[
\Gamma_v^{(1)} = (d_v - 1)(d_v - 2) \sum_{x \in N_v} (d_x - 1), \quad v \in V.
\]

We assign a weight \( \Gamma_v^{(1)} \) to each node \( v \in V \). Define \( \Gamma^{(1)} = \sum_{v \in V} \Gamma_v^{(1)} \) and \( \rho^{(1)} = \frac{\Gamma^{(1)}}{\Gamma_{\text{tot}}} \). To sample a 5-node CIS, T-5 uses six steps:

**Step 1.** Sample a node \( v \) from \( V \) according to the distribution \( \rho^{(1)} = \{\rho_v^{(1)} : v \in V\} \).

**Step 2.** Sample a node \( u \) from \( N_v \) according to the distribution \( \sigma^{(v)} = \{\sigma_u^{(v)} : u \in N_v\} \), where \( \sigma_u^{(v)} \) is defined as

\[
\sigma_u^{(v)} = \frac{d_u - 1}{\sum_{x \in N_v} (d_x - 1)}, \quad u \in N_v;
\]

**Step 3.** Sample a node \( w \) from \( N_v \setminus \{u\} \) at random;

**Step 4.** Sample a node \( r \) from \( N_v \setminus \{u, w\} \) at random;

**Step 5.** Sample a node \( t \) from \( N_v \setminus \{\{u, v, w, r, t\} \) at random;

**Step 6.** Return the CIS \( s \) that includes nodes \( v, u, w, r, \) and \( t \).

One may wonder why not sample \( v \) from \( V \) and \( u \) from \( N_v \) uniformly in the first two steps? This is because it is difficult to
compute and remove the sampling bias of \( s \) when sampling \( v \) from \( V \) and \( u \) from \( \mathcal{N}_v \), uniformly. In contrast, sampling \( v \) and \( u \) according to specific distributions \( \rho_v \) and \( \sigma(u) \) leads to the sampling bias of T-5 that can be easily derived and removed, which will be discussed later (Theorems 1 and 2). We run the above procedure \( K \) times to obtain \( K \) CISes \( s^{(1)}_1, \ldots, s^{(1)}_K \). The pseudo-code of T-5 is shown in Algorithm 1. In Algorithm 1, function \( \text{WeightRandomVertex}(V, \rho_v) \) returns a node sampled from \( V \) according to the distribution \( \rho_v \) = \( \{ \rho_v : v \in V \} \), function \( \text{RandomVertex}(X) \) returns a node sampled from \( X \) at random, and function \( \text{CIS}(\{v, u, w, r, t\}) \) returns the CIS with the node set \( \{v, u, w, r, t\} \) in \( G \).

**Algorithm 1: The pseudo-code of T-5.**

```plaintext
input : \( G = (V, E) \) and \( K \).
output: \( \hat{\eta}_i^{(1)} \).
for \( i \in \Omega_1 \) do
    \[ \hat{\eta}_i^{(1)} \leftarrow 0; \]
end
for \( k \in [1, K] \) do
    \begin{align*}
    &v \leftarrow \text{WeightRandomVertex}(V, \rho_v^{(1)}); \\
    &u \leftarrow \text{WeightRandomVertex}(\mathcal{N}_v, \sigma(v)); \\
    &w \leftarrow \text{RandomVertex}(\mathcal{N}_v - \{u\}); \\
    &r \leftarrow \text{RandomVertex}(\mathcal{N}_v - \{u, w\}); \\
    &t \leftarrow \text{RandomVertex}(\mathcal{N}_v - \{u, v\}); \\
    &\hat{\sigma}_k^{(1)} \leftarrow \text{CIS}(\{v, u, w, r, t\});
    
    &\text{if } t \neq w \text{ and } t \neq r \text{ then}
    \begin{align*}
    &i \leftarrow G^{(5)}(s_k^{(1)}); \\
    &\hat{\eta}_i^{(1)} \leftarrow \hat{\eta}_i^{(1)} + \frac{1}{K_i p_i^{(1)}};
    \end{align*}
    \end{align*}
end
```

For a CIS \( s \) isomorphic to graphlet \( G^{(5)}_i \), \( 1 \leq i \leq 21 \), let \( \phi_i^{(1)} \) denote the number of subgraphs in \( s \) that are isomorphic to graphlet \( G^{(5)}_i \). In other word, \( s \) contains \( \phi_i^{(1)} \) subgraphs that are isomorphic to \( G^{(5)}_i \). The value of \( \phi_i^{(1)} \) is given in Table 2. The following theorem shows the sampling bias of T-5, which is critical for estimating \( \eta_i \).

**Theorem 1.** Using the sampling procedure once (i.e., \( K_1 = 1 \)), T-5 returns a CIS \( s \in G^{(5)}_i \) sampled with probability

\[
P_i^{(1)} = \frac{2 \phi_i^{(1)}}{\Gamma(1)}, \quad 1 \leq i \leq 21.
\]

**Proof.** As shown in Fig. 4, we can easily find that there exist two ways to sample a subgraph isomorphic to graphlet \( G^{(5)}_i \) by T-5. Each happens with probability

\[
\rho_v \times \sigma_u \times \frac{1}{d_v - 1} \times \frac{1}{d_u - 2} \times \frac{1}{d_u - 1} = \frac{1}{\Gamma(1)}.
\]

For a 5-node CIS \( s \in G^{(5)}_i \), \( s \) has \( \phi_i^{(1)} \) different subgraphs isomorphic to graphlet \( G^{(5)}_i \), \( 1 \leq i \leq 21 \). Therefore, the probability of T-5 sampling \( s \) is \( \frac{2 \phi_i^{(1)}}{\Gamma(1)} \).

We let \( G^{(5)}(s) \) be the 5-node graphlet ID of \( s \) when \( s \) is a 5-node CIS, and \( -1 \) otherwise. We define

\[
m_i^{(1)} = \sum_{k=1}^{K_1} 1(G^{(5)}(s_k^{(1)}) = i).
\]

For a CIS \( s \) isomorphic to graphlet \( G^{(5)}_i \), \( 1 \leq i \leq 21 \), let \( \phi_i^{(1)} \) denote the number of subgraphs in \( s \) that are isomorphic to graphlet \( G^{(5)}_i \). In other word, \( s \) contains \( \phi_i^{(1)} \) subgraphs that are isomorphic to \( G^{(5)}_i \). The value of \( \phi_i^{(1)} \) is given in Table 2. The following theorem shows the sampling bias of T-5, which is critical for estimating \( \eta_i \).

**Theorem 2.** For \( i \in \Omega_1 \), \( \hat{\eta}_i^{(1)} \) is an unbiased estimator of \( \eta_i \), i.e.,

\[
\mathbb{E}(\hat{\eta}_i^{(1)}) = \eta_i,
\]

and the variance of \( \hat{\eta}_i^{(1)} \) is

\[
\text{Var}(\hat{\eta}_i^{(1)}) = \frac{\eta_i}{K_1 p_i^{(1)}} - \eta_i.
\]

**Proof.** For \( i \in \Omega_1 \) and \( 1 \leq k \leq K_1 \), we have

\[
P(G^{(5)}(s_k^{(1)}) = i) = \sum_{s \in C^{(5)}} P(s_k^{(1)} = s) 1(G^{(5)}(s_k^{(1)}) = i) = p_i^{(1)} \eta_i.
\]

Since \( s_k^{(1)} \) are sampled independently, the random variable \( m_i^{(1)} \) follows the binomial distribution with parameters \( K_1 \) and \( p_i^{(1)} \eta_i \). Then, the expectation and variance of \( m_i^{(1)} \) are

\[
\mathbb{E}(m_i^{(1)}) = K_1 p_i^{(1)} \eta_i,
\]

\[
\text{Var}(m_i^{(1)}) = K_1 p_i^{(1)} (1 - p_i^{(1)} \eta_i).
\]

Therefore, the expectation and variance of \( \hat{\eta}_i^{(1)} \) are computed as

\[
\mathbb{E}(\hat{\eta}_i^{(1)}) = \frac{\mathbb{E}(m_i^{(1)})}{K_1 p_i^{(1)}} = \frac{\mathbb{E}(m_i^{(1)})}{K_1 p_i^{(1)}} = \eta_i,
\]

\[
\text{Var}(\hat{\eta}_i^{(1)}) = \frac{\eta_i}{K_1 p_i^{(1)}} (1 - \frac{1}{p_i^{(1)}} \eta_i).
\]
For $i \neq j$ and $i, j \in \Omega_1$, the covariance of $\hat{\eta}_i^{(1)}$ and $\hat{\eta}_j^{(1)}$
\[ \text{Cov}(\hat{\eta}_i^{(1)}, \hat{\eta}_j^{(1)}) = \frac{\text{Cov}(\sum_{k=1}^{K_1} \mathbf{1}(G(5)(s_k^{(1)}) = i), \sum_{k=1}^{K_1} \mathbf{1}(G(5)(s_k^{(1)}) = j))}{K_1^2 p_i^{(1)} p_j^{(1)}} \]
\[ = \sum_{k=1}^{K_1} \sum_{l=1}^{K_1} \text{Cov}(\mathbf{1}(G(5)(s_k^{(1)}) = i), \mathbf{1}(G(5)(s_l^{(1)}) = j)). \]

Each sampled 5-node CIS is obtained independently, so we have
\[ \text{Cov}(1(M(5)(s_k^{(1)}) = i), 1(M(5)(s_l^{(1)}) = j)) = 0, \quad k \neq l. \]

In addition, we have $P(M(5)(s_k^{(1)}) = i \land M(5)(s_l^{(1)}) = j) = 0$ when $i \neq j$. Therefore, we obtain
\[ \text{Cov}(1(M(5)(s_k^{(1)}) = i), 1(M(5)(s_l^{(1)}) = j)) = \text{E}(1(M(5)(s_k^{(1)}) = i)1(M(5)(s_l^{(1)}) = j)) - \text{E}(1(M(5)(s_k^{(1)}) = i))\text{E}(1(M(5)(s_l^{(1)}) = j)) \]
\[ = 0 - p_i^{(1)} p_j^{(1)} \hat{\eta}_i \hat{\eta}_j \]
\[ = -p_i^{(1)} p_j^{(1)} \frac{\hat{\eta}_i \hat{\eta}_j}{K_1}. \]

Now, we easily have
\[ \text{Cov}(\hat{\eta}_i^{(1)}, \hat{\eta}_j^{(1)}) = \sum_{k=1}^{K_1} \sum_{l=1}^{K_1} \text{Cov}(1(G(5)(s_k^{(1)}) = i), 1(G(5)(s_l^{(1)}) = j)) \frac{K_1^2 p_i^{(1)} p_j^{(1)}}{K_1} \]
\[ = -\frac{\hat{\eta}_i \hat{\eta}_j}{K_1}. \]

### 3.2 The Path-5 Sampling Method

The pseudo-code of Path-5 is shown in Algorithm 2. Let
\[ \Gamma_v^{(2)} = \left( \sum_{x \in N_v} (d_x - 1) \right)^2 - \sum_{x \in N_v} (d_x - 1)^2, \quad v \in V. \]

We assign a weight $\Gamma_v^{(2)}$ to each node $v \in V$. Define $\Gamma^{(2)} = \sum_{v \in V} \Gamma_v^{(2)}$ and $\rho_v^{(2)} = \frac{\Gamma_v^{(2)}}{\Gamma^{(2)}}$. To sample a 5-node CIS, Path-5 mainly consists of six steps:

**Step 1.** Sample a node $v$ from $V$ according to the distribution $\rho_v^{(2)} = \{ \rho_v^{(2)} : v \in V \}$.

**Step 2.** Sample a node $u$ from $N_v$ according to the distribution $\tau_v^{(u)} = \{ \tau_v^{(u)} : u \in N_v \}$, where we define
\[ \tau_v^{(u)} = \frac{(d_u - 1)(\sum_{y \in N_v \setminus \{u\}} (d_y - 1))}{\Gamma_v^{(2)}}, \quad u \in N_v; \]

**Step 3.** Sample a node $w$ from $N_v - \{u\}$ according to the distribution $\mu_v^{(w,u)} = \{ \mu_v^{(w,u)} : w \in N_v - \{u\} \}$, where we define
\[ \mu_v^{(w,u)} = \frac{d_w - 1}{\sum_{y \in N_v - \{u\}} (d_y - 1)}, \quad w \in N_v - \{u\}; \]

**Step 4.** Sample a node $r$ from $N_u - \{v\}$ at random.

**Step 5.** Sample a node $t$ from $N_w - \{v\}$ at random.

**Step 6.** Return the CIS $s$ that includes nodes $v, u, w, r, t$.

**Algorithm 2:** The pseudo-code of Path-5.

**input:** $G = (V, E)$ and $K_2$.

**output:** $\hat{\eta}_i^{(2)}$.

**for** $i \in \Omega_2$ **do**
\[ \hat{\eta}_i^{(2)} \leftarrow 0; \]
**end**

**for** $k \in [1, K_2]$ **do**
\[ v \leftarrow \text{WeightRandomVertex}(V, \rho_v^{(2)}); \]
\[ u \leftarrow \text{WeightRandomVertex}(N_v, \tau_v^{(u)}); \]
\[ w \leftarrow \text{WeightRandomVertex}(N_u - \{u\}, \mu_v^{(w,u)}); \]
\[ r \leftarrow \text{RandomVertex}(N_u - \{v\}); \]
\[ t \leftarrow \text{RandomVertex}(N_w - \{v\}); \]
\[ s^{(2)} \leftarrow \text{CIS}(v, u, w, r, t); \]
**if** $t \neq u$ and $r \neq w$ and $t \neq r$ **then**
\[ i \leftarrow G(5)(s^{(2)}); \]
\[ \hat{\eta}_i^{(2)} \leftarrow \hat{\eta}_i^{(2)} + \frac{1}{K_2 \rho_v^{(2)}}, \]
**end**
**end**

We run the above procedure $K_2$ times to obtain $K_2$ CISes $s^{(2)}_1, \ldots, s^{(2)}_{K_2}$. For a CIS $s$ isomorphic to graphlet $G^{(5)}_i$, $1 \leq i \leq 21$, let $\phi^{(2)}_i$ denote the number of subgraphs in $s$ that are isomorphic to $G^{(5)}_i$. In other word, $s$ contains $\phi^{(2)}_i$ subgraphs that are isomorphic to $G^{(5)}_i$. The value of $\phi^{(2)}_i$ is given in Table 2. The following theorem shows the sampling bias of Path-5.

**Theorem 3.** Using the sampling procedure once (i.e., $K_2 = 1$), Path-5 samples a CIS $s \in C_i^{(5)}$ with probability
\[ p_i^{(2)} = \frac{2\phi^{(2)}_i}{\Gamma^{(2)}}, \quad 1 \leq i \leq 21. \]

**Proof.** As shown in Fig. 5, we can see that there exist two ways to sample a subgraph isomorphic to graphlet $G^{(5)}_i$ by Path-5. Each one happens with probability
\[ \frac{\phi^{(2)}_i \times \tau^{(u)} \times \mu_v^{(w,u)} 	imes \frac{1}{d_u - 1} \times \frac{1}{d_w - 1}}{\Gamma^{(2)}} = \frac{1}{\Gamma^{(2)}}. \]

For a 5-node CIS $s \in C_i^{(5)}$, $s$ has $\phi^{(2)}_i$ subgraphs isomorphic to graphlet $G^{(5)}_i$, $1 \leq i \leq 21$. Thus, the probability of Path-5 sampling $s$ is $\frac{2\phi^{(2)}_i}{\Gamma^{(2)}}$. \qed
Figure 5. The ways of Path-5 sampling a subgraph \( s \) isomorphic to graphlet \( G(5) \), where \( v, u, w, r, \) and \( t \) are the variables in Algorithm 2, i.e., the nodes sampled at the 1\(^{\text{st}}, \) 2\(^{\text{nd}}, \) 3\(^{\text{rd}}, \) 4\(^{\text{th}}, \) and 5\(^{\text{th}} \) steps respectively.

Denote

\[
m_i^{(2)} = \sum_{k=1}^{K_2} 1(G(5)(s_k^{(2)}) = i).
\]

Then, we have

\[
E(m_i^{(2)}) = K_2 \hat{p}_i^{(2)} \eta_i.
\]

For \( i \in \Omega_2, \hat{p}_i^{(2)} \) is larger than zero and we then estimate \( \eta_i \) as

\[
\hat{\eta}_i^{(2)} = \frac{m_i^{(2)}}{K_2 \hat{p}_i^{(2)}}.
\]

**Theorem 4.** For \( i \in \Omega_2, \hat{\eta}_i^{(2)} \) is an unbiased estimator of \( \eta_i \) and its variance is

\[
\text{Var}(\hat{\eta}_i^{(2)}) = \frac{\eta_i}{K_2^2} \left( \frac{1}{\hat{p}_i^{(2)}} - \eta_i \right).
\]

We estimate \( \text{Var}(\hat{\eta}_i^{(2)}) \) by replacing \( \eta_i \) with \( \hat{\eta}_i^{(2)} \) in (2). The covariance of \( \hat{\eta}_i^{(2)} \) and \( \hat{\eta}_j^{(2)} \) is

\[
\text{Cov}(\hat{\eta}_i^{(2)}, \hat{\eta}_j^{(2)}) = -\frac{\eta_i \eta_j}{K_2^2}, \quad i \neq j, i, j \in \Omega_2;
\]

which is used to compute the variance of the estimate of \( \eta_2 \) given in Section 3.3.

Its proof is similar to the proof of Theorem 2.

### 3.3 Hybrid Estimator of 5-Node Graphlet Counts

We estimate \( \eta_i \) as \( \hat{\eta}_i^{(1)} \) and \( \hat{\eta}_i^{(2)} \) for \( i \in \Omega_1 \cap \Omega_2 \) and \( i \in \Omega_2 - \Omega_1 \), respectively. When \( i \in \Omega_1 \cap \Omega_2 \), according to (29), we estimate \( \eta_i \) based on its two unbiased estimates \( \hat{\eta}_i^{(1)} \) and \( \hat{\eta}_i^{(2)} \). Formally, let

\[
\lambda_i^{(1)} = \frac{\text{Var}(\hat{\eta}_i^{(1)})}{\text{Var}(\hat{\eta}_i^{(1)}) + \text{Var}(\hat{\eta}_i^{(2)})}, \quad \lambda_i^{(2)} = \frac{\text{Var}(\hat{\eta}_i^{(2)})}{\text{Var}(\hat{\eta}_i^{(1)}) + \text{Var}(\hat{\eta}_i^{(2)})}.
\]

Here \( \text{Var}(\hat{\eta}_i^{(1)}) \) and \( \text{Var}(\hat{\eta}_i^{(2)}) \) are estimated by Theorems 2 and 4. For \( i \in \Omega_1 \cup \Omega_2 = \{1, 3, 4, 5, \ldots, 21\} \), we finally estimate \( \eta_i \) as

\[
\hat{\eta}_i = \begin{cases} 
\lambda_i^{(1)} \hat{\eta}_i^{(1)} + \lambda_i^{(2)} \hat{\eta}_i^{(2)} & , i \in \Omega_1 \cap \Omega_2, \\
\hat{\eta}_i^{(1)} & , i \in \Omega_1 - \Omega_2, \\
\hat{\eta}_i^{(2)} & , i \in \Omega_2 - \Omega_1.
\end{cases}
\]

Note that \( \Omega_1 \cup \Omega_2 = \{1, 2, \ldots, 21\} - \{2\} \). Next, we discuss our method for estimating \( \eta_2 \). For a CIS \( s \) isomorphic to graphlet \( G(5)_i, 1 \leq i \leq 21 \), let \( \phi_i^{(3)} \) denote the number of subgraphs in \( s \) that are isomorphic to graphlet \( G(5)_2 \). In other word, \( s \) contains \( \phi_i^{(3)} \) subgraphs that are isomorphic to \( G(5)_2 \). The value of \( \phi_i^{(3)} \) is given in Table 2. Let

\[
\Lambda_4 = \sum_{v \in V} \left( \frac{d_v}{4} \right).
\]

Then, the number of all 5-node subgraphs (not necessarily induced) in \( G \) isomorphic to graphlet \( G(5)_2 \) is \( \Lambda_4 \). Let \( \Omega_3 = \{ j : \phi_j^{(3)} > 0 \} \) and \( \Omega_3' = \Omega_3 - \{2\} \). We observe that

\[
\sum_{i \in \Omega_3} \phi_i^{(3)} \eta_i = \Lambda_4.
\]

Since \( \phi_2^{(3)} = 1 \), we estimate \( \eta_2 \) as

\[
\hat{\eta}_2 = \Lambda_4 - \sum_{i \in \Omega_3'} \phi_i^{(3)} \eta_i.
\]

**Theorem 5.** \( \hat{\eta}_i \) is an unbiased estimator of \( \eta_i, 1 \leq i \leq 21 \). For \( i \in \Omega_1 \cup \Omega_2 = \{1, 2, \ldots, 21\} - \{2\} \), the variance of \( \hat{\eta}_i \) is

\[
\text{Var}(\hat{\eta}_i) = \begin{cases} 
\text{Var}(\hat{\eta}_i^{(1)}) + \text{Var}(\hat{\eta}_i^{(2)}), & i \in \Omega_1 \cap \Omega_2, \\
\text{Var}(\hat{\eta}_i^{(1)}), & i \in \Omega_1 - \Omega_2, \\
\text{Var}(\hat{\eta}_i^{(2)}), & i \in \Omega_2 - \Omega_1.
\end{cases}
\]

In the above equation, we estimate \( \text{Var}(\hat{\eta}_i^{(1)}) \) and \( \text{Var}(\hat{\eta}_i^{(2)}) \) using the methods in Theorems 2 and 4. We compute the variance \( \text{Var}(\hat{\eta}_2) \) as

\[
\sum_{i \in \Omega_3} (\phi_i^{(3)})^2 \text{Var}(\hat{\eta}_i) + \sum_{i, j \in \Omega_3, i \neq j} \phi_i^{(3)} \phi_j^{(3)} \text{Cov}(\hat{\eta}_i, \hat{\eta}_j),
\]

where \( \text{Cov}(\hat{\eta}_i, \hat{\eta}_j) \) is

\[
\begin{cases} 
-\sum_{i=1,2} \lambda_i^{(i)} \lambda_j^{(i)} \eta_i \eta_j & , i, j \in \Omega_1 \cap \Omega_2, \\
-\lambda_j^{(1)} \eta_i \eta_j & , i \in \Omega_1 - \Omega_2, j \in \Omega_1 \cap \Omega_2, \\
-\lambda_j^{(1)} \eta_i \eta_j & , i \in \Omega_1 \cap \Omega_2, j \in \Omega_1 - \Omega_2, \\
-\lambda_j^{(2)} \eta_i \eta_j & , i \in \Omega_2 - \Omega_1, j \in \Omega_1 \cap \Omega_2, \\
-\lambda_j^{(2)} \eta_i \eta_j & , i \in \Omega_2 - \Omega_1, j \in \Omega_1 - \Omega_2, \\
0 & , i \in \Omega_1 - \Omega_2, j \in \Omega_2 - \Omega_1, \\
0 & , i \in \Omega_2 - \Omega_1, j \in \Omega_1 - \Omega_2.
\end{cases}
\]

**Proof.** For \( i \in \Omega_1 \cup \Omega_2 \), Theorems 2 and 4 tell us that \( \hat{\eta}_i^{(1)} \) and \( \hat{\eta}_i^{(2)} \) are unbiased estimators of \( \eta_i^{(1)} \), and they are independent. Moreover, \( \lambda_i^{(1)} + \lambda_i^{(2)} = 1 \). Therefore, we easily find that \( \hat{\eta}_i \) is also an unbiased estimator of \( \eta_i^{(1)} \), and its variance is (4). Next, we study the expectation and variance of \( \hat{\eta}_2 \). The expectation of \( \hat{\eta}_2 \) is computed as

\[
\text{E}(\hat{\eta}_2) = \Lambda_4 - \sum_{i \in \Omega_3'} \phi_i^{(3)} \text{E}(\hat{\eta}_i) = \Lambda_4 - \sum_{i \in \Omega_3'} \phi_i^{(3)} \eta_i = \eta_2.
\]

Before we compute the covariance of \( \hat{\eta}_i \) and \( \hat{\eta}_j \) for \( i, j \in \Omega_1 \cup \Omega_2 \) and \( i \neq j \), we first introduce three equations: (I) for any \( i, j \in \Omega_3 \),
in memory, where $ACC_{\Gamma}(i) = \sum_{j=1}^{i} V[j]$, $1 \leq i \leq |V|$. Note that $ACC_{\Gamma}(i) = \Gamma(i)$.
Let $ACC_{\Gamma}(0) = 0$. Then, $\text{WeightRandomVertex}(V, \rho(1))$ is easily achieved by the following three steps: $step\ 1$ select a number $rnd$ from \{1, $\ldots$, $\Gamma(1)$\} at random; $step\ 2$ find $i$ such that $ACC_{\Gamma}(i) < rnd \leq ACC_{\Gamma}(i+1)$, which is solved by binary search; $step\ 3$ return $V[i]$. Thus, the space and time complexities of $\text{WeightRandomVertex}(V, \rho(1))$ are $O(|V|)$ and $O(\log |V|)$ respectively.

$\text{WeightRandomVertex}(V, \rho(2))$: It is achieved similarly to that for $\text{WeightRandomVertex}(V, \rho(1))$.

$\text{WeightRandomVertex}(N_v, \sigma(\cdot))$: We use a list $N_v[1, \ldots, d_v]$ to store the neighbors of $v$. We store an array $ACC_{\sigma}(\cdot)[1, \ldots, d_v]$ in memory, where $ACC_{\sigma}(i) = \sum_{j=1}^{i} d_{N_v[j]} - 1$, $1 \leq i \leq d_v$. Let $ACC_{\sigma}(0) = 0$. Then, $\text{WeightRandomVertex}(N_v, \sigma(\cdot))$ is easily achieved by the following three steps: $step\ 1$ select a random number $rnd$ from \{1, $\ldots$, $ACC_{\sigma}(d_v)$\} at random; $step\ 2$ find $i$ such that $ACC_{\sigma}(i) < rnd \leq ACC_{\sigma}(i+1)$, which is solved by binary search; $step\ 3$ return $N_v[i]$. Thus, the space and time complexities of $\text{WeightRandomVertex}(N_v, \sigma(\cdot))$ are $O(d_v)$ and $O(\log d_v)$ respectively.

$\text{WeightRandomVertex}(N_v, \tau(\cdot))$: It is achieved similarly to that for $\text{WeightRandomVertex}(N_v, \sigma(\cdot))$.

$\text{RandomVertex}(N_v - \{u\}, \mu(v,\cdot))$: As mentioned, we use a list $N_v[1, \ldots, d_v]$ to store the neighbors of $v$, and store an array $\text{G}_{\sigma}(\cdot)[1, \ldots, d_v]$ in memory, where $\text{G}_{\sigma}(i) = \sum_{j=1}^{i} d_{N_v[j]} - 1$, $1 \leq i \leq d_v$. Let $POS_{v,u}$ be the index of $u$ in $N_v[1, \ldots, d_v]$, i.e., $N_v[POS_{v,u}] = u$. Then, function $\text{WeightRandomVertex}(N_v - \{u\}, \mu(v,\cdot))$ is easily achieved by the following three steps: $step\ 1$ select a random number $rnd$ from \{1, $\ldots$, $\text{G}_{\sigma}(d_v)$\} at random; $step\ 2$ find $i$ such that $\text{G}_{\sigma}(i) < rnd \leq \text{G}_{\sigma}(i+1)$, which is solved by binary search; $step\ 3$ return $N_v[i]$. Therefore, the time complexity of $\text{WeightRandomVertex}(N_v - \{u\}, \mu(v,\cdot))$ is $O(\log d_v)$.

$\text{RandomVertex}(N_v - \{u\})$: Function $\text{RandomVertex}(N_v - \{u\})$ is achieved by two steps: $step\ 1$ select a number $rnd$ from \{1, $\ldots$, $d_v\} - \{POS_{v,u}\}$ at random; $step\ 2$ return $N_v[rnd]$. Thus, the computational complexity of $\text{RandomVertex}(N_v - \{u\})$ is $O(1)$.

$\text{RandomVertex}(N_v - \{u, w\})$. It is achieved by two steps: $step\ 1$ select a number $rnd$ from \{1, $\ldots$, $d_v\} - \{POS_{v,u}, POS_{v,w}\}$ at random; $step\ 2$ return $N_v[rnd]$. Thus, the computational complexity of $\text{RandomVertex}(N_v - \{u, w\})$ is $O(1)$.

In summary, the space and time complexities of T-S sampling $K_1$ CIses are $O(|V| + |E|)$ and $O(|V| + K_1 \log |V|)$ respectively, and the space and time complexities of Path-5 sampling $K_2$ CIses are $O(|V| + |E|)$ and $O(|E| + K_2 \log |V|)$ respectively. Therefore, the space and time complexities of MOSS-5 are $O(|V| + |E|)$ and $O(|E| + (K_1 + K_2) \log |V|)$ respectively.

### 3.5 Parameter Settings

From Theorem 5, we can see that the variance of $\tilde{\eta}_i$ greatly depends on the sampling budget $K_1$ for $i \in \Omega_1 - \Omega_2$. In contrast, $K_2$ is used to guarantee the desired variance of $\tilde{\eta}_i$, $i \in \Omega_2 - \Omega_1$. Thus, $K_1$ and $K_2$ can be set according to the above observations. Given a total sampling budget $K$ (i.e., $K = K_1 + K_2$), how
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2756836, IEEE Transactions on Knowledge and Data Engineering

Figure 6. Real values of 5-node graphlet counts.

to set $K_1$ and $K_2$? Our empirical study shows that $p_i^{(1)}$ and $p_i^{(2)}$ have similar values. Therefore, T-5 and Path-5 exhibit similar estimation errors when $K_1 = K_2$ and we set $K_1 = K_2 = \frac{K}{2}$ in this paper for simplicity.

4 Evaluation

4.1 Datasets

We perform our experiments on a variety of publicly available graph datasets with 0.1 to 117 million edges taken from the Stanford Network Analysis Platform (SNAP)

4. www.snap.stanford.edu

4.2 Metric

We study the normalized root mean square error (NRMSE) to measure the relative error of the graphlet count estimates $\hat{\eta}_i$ with respect to its true value $\eta_i$, $i = 1, \ldots, 21$. It is defined as

$$\text{NRMSE}(\hat{\eta}_i) = \frac{\sqrt{\text{MSE}(\hat{\eta}_i)}}{\hat{\eta}_i}, \quad i = 1, \ldots, 21,$$

where MSE($\hat{\eta}_i$) denotes the mean square error of $\hat{\eta}_i$, i.e.,

$$\text{MSE}(\hat{\eta}_i) = \mathbb{E}((\hat{\eta}_i - \eta_i)^2) = \text{Var}(\hat{\eta}_i) + (\mathbb{E}(\hat{\eta}_i) - \eta_i)^2.$$

We can see that MSE($\hat{\eta}_i$) decomposes into a sum of the variance and bias of the estimator $\hat{\eta}_i$. Both quantities are important and need to be as small as possible to achieve good estimation performance. When $\hat{\eta}_i$ is an unbiased estimator of $\eta_i$, we have $\text{MSE}(\hat{\eta}_i) = \text{Var}(\hat{\eta}_i)$ and then NRMSE($\hat{\eta}_i$) is equivalent.
to the normalized standard error of $\hat{\eta}_i$, i.e., $\text{NRMSE}(\hat{\eta}_i) = \sqrt{\text{Var}(\hat{\eta}_i)}/\eta_i$. In addition, we define $\frac{\eta_i - \hat{\eta}_i}{\eta_i}$ as the relative error of $\hat{\eta}_i$, and also study the CCDF (complementary cumulative distribution function) of $\frac{\eta_i - \hat{\eta}_i}{\eta_i}$, that is,

$$\text{CCDF}\left(\frac{\eta_i - \hat{\eta}_i}{\eta_i} > x\right) = P\left(\frac{\eta_i - \hat{\eta}_i}{\eta_i} > x\right).$$

In our experiments, we calculate the NRMSE and CCDF over 1,000 runs. Our experiments are conducted on a server with a Quad-Core AMD Opteron (tm) 8379 HE CPU 2.39 GHz processor and 128 GB DRAM memory.

### 4.3 Runtime
Table 4 shows the computational time of MOSS-5. We can see that MOSS-5 is quite computationally efficient, which takes less than 200 seconds to sample 10 million 5-node CISes for all graphs studied in this paper. We observe that the sampling budget $K$ does not offer a strictly linear increase in runtime. This is because the time cost of computing $\Gamma(v)$ cannot be neglected for all nodes $v \in V$, which equals 68, 30, 20, 10, 9, 4, 3, 1, and 0.05 seconds for graphs com-Orkut, LiveJournal, Pokec, Filckr, Xiami, Wiki-Talk, Web-Google, YouTube, and ca-HepPh respectively. Moreover, we observe that sampling large graphs such as com-Orkut is computationally expensive than sampling small graphs such as ca-HepPh. From the experimental results in Section 4.5 (Table 5), we show that MOSS-5 requires less than 2 minutes to compute 5-node graphlet counts with NRMSEs smaller than 0.1 for graphs with millions of nodes and edges.

### 4.4 Accuracy
Figure 7 shows NRMSEs of MOSS-5 with sampling budget $K = 100,000, 1,000,000, 10,000,000$. For all graphs, most 5-node graphlets’ NRMSEs are smaller than 0.1 when $K = 100,000, 10,000,000$. In addition, we observe that NRMSEs are proportional to $\frac{1}{\sqrt{K}}$, which is consistent with Theorem 5. For example, in Fig. 7 we see that a one order of magnitude increase in $K$ decreases NRMSEs by $\frac{1}{\sqrt{K}}$. Fig. 8 shows the CCDF of relative error $\frac{\eta_i - \hat{\eta}_i}{\eta_i}$, $1 \leq i \leq 21$, given by MOSS-5 with sampling budget $K = 1,000,000$. We can see that more than 99% of estimates $\hat{\eta}_i$ obtained by 1,000 independent runs have a relative error smaller than $3\text{NRMSE}(\hat{\eta}_i)$. From Figs. 6, 7 and 8, we observe that MOSS-5 exhibits smaller estimation errors for graphlets with large graphlet counts (i.e., frequent graphlets) than graphlets with small graphlet counts (i.e., rare graphlets).

### 4.5 Comparison with Prior Art
MOSS vs. fast exact counting method ESCAPE [28]. Table 5 shows the expected smallest computational time of MOSS-5 required to obtain all estimates $\hat{\eta}_1, \ldots, \hat{\eta}_{21}$ with NRMSE smaller than 0.1. To compute $\eta_1, \ldots, \eta_{21}$, the state-of-the-art exact computing method ESCAPE requires 52 hours, 32 hours, 23 hours, 1 hour, 31 minutes, 10 minutes, 8 minutes, 3 minutes, and 2 minutes for graphs Filckr, com-Orkut, LiveJournal, Pokec, Wiki-Talk, ca-HepPh, Xiami, YouTube, and Web-Google respectively. We can see that the computational time of ESCAPE does not strictly increase with the graph size. For example, graph ca-HepPh is more than ten times smaller than graphs YouTube and Web-Google. To compute $\eta_1, \ldots, \eta_{21}$, however, ESCAPE requires much more time for ca-HepPh than YouTube and Web-Google. From Table 5, we see that our method MOSS-5 is 2 to 18,945 times faster than ESCAPE when providing accurate estimates with NRMSE smaller than 0.1. From the results in Section 4.4 (Fig. 7), we observe that when the maximum of NRMSEs of all graphlets’ estimates equals 0.1, NRMSEs of many graphlets’ estimates are much smaller than 0.1. In Table 5, we show the average NRMSE $\frac{1}{21} \sum_{i=1}^{21} \text{NRMSE}(\hat{\eta}_i)$ when $\max_{i=1,\ldots,21} \text{NRMSE}(\hat{\eta}_i) = 0.1$. We can see that the average NRMSE varies from 0.01 to 0.04 for all graphs studied in this paper.

### Table 3
Undirected Graph datasets used in our experiments. “max-degree” represents the maximum number of edges incident to a node in the undirected graph.

<table>
<thead>
<tr>
<th>Graph</th>
<th>nodes</th>
<th>edges</th>
<th>max-degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>com-Orkut</td>
<td>3,072,941</td>
<td>117,185,803</td>
<td>33,213</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>5,189,809</td>
<td>48,688,097</td>
<td>15,017</td>
</tr>
<tr>
<td>Pokec</td>
<td>1,632,803</td>
<td>22,301,964</td>
<td>14,854</td>
</tr>
<tr>
<td>Flickr</td>
<td>1,715,255</td>
<td>15,555,041</td>
<td>27,236</td>
</tr>
<tr>
<td>Xiami</td>
<td>1,753,620</td>
<td>16,018,571</td>
<td>19,727</td>
</tr>
<tr>
<td>Wiki-Talk</td>
<td>2,394,385</td>
<td>4,659,565</td>
<td>100,029</td>
</tr>
<tr>
<td>Web-Google</td>
<td>875,713</td>
<td>4,322,051</td>
<td>6,332</td>
</tr>
<tr>
<td>YouTube</td>
<td>1,138,499</td>
<td>2,990,443</td>
<td>28,754</td>
</tr>
<tr>
<td>ca-HepPh</td>
<td>12,008</td>
<td>118,490</td>
<td>491</td>
</tr>
</tbody>
</table>

### Table 4
Computational time (seconds) of MOSS-5.

<table>
<thead>
<tr>
<th>Graph</th>
<th>sampling budget $K$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100,000</td>
</tr>
<tr>
<td>com-Orkut</td>
<td>69.1 s</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>31.5 s</td>
</tr>
<tr>
<td>Pokec</td>
<td>20.7 s</td>
</tr>
<tr>
<td>Flickr</td>
<td>10.4 s</td>
</tr>
<tr>
<td>Xiami</td>
<td>10.4 s</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>4.6 s</td>
</tr>
<tr>
<td>Wiki-Talk</td>
<td>4.1 s</td>
</tr>
<tr>
<td>Web-Google</td>
<td>2.5 s</td>
</tr>
<tr>
<td>YouTube</td>
<td>0.53 s</td>
</tr>
<tr>
<td>ca-HepPh</td>
<td>0.03 s</td>
</tr>
</tbody>
</table>

### Table 5
Computational time (seconds) and accuracy of MOSS-5 in comparison with state-of-the-art exact counting method ESCAPE.

<table>
<thead>
<tr>
<th>Graph</th>
<th>ESCAPE (time)</th>
<th>MOSS-5, $\max_{i=1,\ldots,21} \text{NRMSE}(\hat{\eta}_i) = 0.1$ (time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flickr</td>
<td>189.450 s</td>
<td>10 s</td>
</tr>
<tr>
<td>com-Orkut</td>
<td>116.029 s</td>
<td>103 s</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>82.445 s</td>
<td>31 s</td>
</tr>
<tr>
<td>Pokec</td>
<td>3.696 s</td>
<td>31 s</td>
</tr>
<tr>
<td>Wiki-Talk</td>
<td>1.877 s</td>
<td>47 s</td>
</tr>
<tr>
<td>Xiami</td>
<td>518 s</td>
<td>82 s</td>
</tr>
<tr>
<td>Web-Google</td>
<td>112 s</td>
<td>25 s</td>
</tr>
<tr>
<td>YouTube</td>
<td>193 s</td>
<td>96 s</td>
</tr>
<tr>
<td>ca-HepPh</td>
<td>589 s</td>
<td>64 s</td>
</tr>
</tbody>
</table>

MOSS vs. sampling methods Guise [37] and Graft [38]. Most previous work focuses on estimating 5-node motif concentrations, i.e., $\omega_i = \sum_{j=1}^{21} \eta_{i,j}$, $i = 1, \ldots, 21$. We run MOSS-5 and the state-of-the-art sampling methods Guise [37] and Graft [38] on all above graphs and increase their sampling budgets until the estimation errors of motif concentrations are within 10%. Guise uses a Metropolis-Hastings Random Walk (MHRW) method to...
uniformly sample CISes from all 3-, 4-, and 5-node CISes. To conduct a fair comparison, we adapt GUISE to focus on 5-node CISes similarly to [39]. Graft samples a fraction of edges from $G$ at random, and then enumerates all 5-node CISes that include at least one edge in the set of sampled edges. In practice, it is not easy to obtain an estimation with a desired accuracy for GUISE [37] and Graft [38]. In our experiments, we increase their sampling budgets until the relative errors of their estimates are no more than 10% with respect to the real values of all graphlet concentrations. Fig. 9 shows the runtime of Graft and GUISE normalized with respect to the runtime of MOSS-5 (i.e. the runtime of MOSS-5 of unit 1). We can see that our method MOSS-5 is 2 to 3 orders of magnitude faster than Graft and GUISE.

5 RELATED WORK

In this paper, we study the problem of estimating the counts of 3-, 4-, and 5-node graphlets for a single large graph, which is much different from the problem of computing the number of subgraph patterns appearing in a large set of graphs [40]. A variety of centralized and distributed algorithms have been developed to enumerate and count all triangles in large undirected graphs [41]–[45]. Recently, a considerable attention has been given to designing fast algorithms for counting higher order subgraph patterns such as 4- and 5-node graphlets. [28,44]–[47] develop fast algorithms for counting 4- and 5-node undirected graphlets by utilizing the relationships between 3-, 4-, and 5-node graphlet counts. In addition, quite a few efforts have been devoted to designing sampling
methods for computing a large graph’s graphlet concentrations (or, motif concentrations) [37][39,48]-[51], but they fail to compute graphlet counts. Alon et al. [52] propose a color-coding method to reduce the computational cost of counting subgraphs. Color-coding reduces the computation by coloring nodes randomly and enumerating only colorful CISes (i.e., CISes that consist of nodes with distinct colors), but [27] reveals that the color-coding method is not scalable and is hindered by the sheer number of colorful CISes. [21][24] develop sampling methods to estimate the number of triangles of static and dynamic graphs. Jha et al. [27] develop sampling methods to estimate counts of 4-node graphlets. These methods cannot be used to sample and estimate 5-node graphlet counts. When the graph of interest is not available but given a RESampled graph that is obtained by sampling each edge of the graph’s graphlet concentrations from the RESampled graph. Moreover, [39,51,54] assume that the graph of interest is not given in advance, and they focus on designing crawling methods to query as less nodes as possible to characterize graphlets. In this paper, we assume that the entire graph of interest is given.

Figure 8. (Flickr) CCDF($\eta_i - \eta_j$/$\eta_i$, x), 1 ≤ i ≤ 21, given by MOSS-5 with sampling budget $K = 1,000,000$. 

(a) $|\eta_1 - \eta_1|/\eta_1$, NRMSE($\eta_1$) = 2.6E-03
(b) $|\eta_2 - \eta_2|/\eta_2$, NRMSE($\eta_2$) = 2.0E-04
(c) $|\eta_3 - \eta_3|/\eta_3$, NRMSE($\eta_3$) = 1.5E-03
(d) $|\eta_4 - \eta_4|/\eta_4$, NRMSE($\eta_4$) = 4.0E-03
(e) $|\eta_5 - \eta_5|/\eta_5$, NRMSE($\eta_5$) = 4.6E-03
(f) $|\eta_6 - \eta_6|/\eta_6$, NRMSE($\eta_6$) = 1.6E-02
(g) $|\eta_7 - \eta_7|/\eta_7$, NRMSE($\eta_7$) = 2.8E-03
(h) $|\eta_8 - \eta_8|/\eta_8$, NRMSE($\eta_8$) = 2.9E-03
(i) $|\eta_9 - \eta_9|/\eta_9$, NRMSE($\eta_9$) = 4.1E-03
(j) $|\eta_{10} - \eta_{10}|/\eta_{10}$, NRMSE($\eta_{10}$) = 6.7E-03
(k) $|\eta_{11} - \eta_{11}|/\eta_{11}$, NRMSE($\eta_{11}$) = 3.1E-03
(l) $|\eta_{12} - \eta_{12}|/\eta_{12}$, NRMSE($\eta_{12}$) = 9.7E-03
(m) $|\eta_{13} - \eta_{13}|/\eta_{13}$, NRMSE($\eta_{13}$) = 1.6E-02
(n) $|\eta_{14} - \eta_{14}|/\eta_{14}$, NRMSE($\eta_{14}$) = 4.9E-03
(o) $|\eta_{15} - \eta_{15}|/\eta_{15}$, NRMSE($\eta_{15}$) = 5.3E-03
(p) $|\eta_{16} - \eta_{16}|/\eta_{16}$, NRMSE($\eta_{16}$) = 8.6E-03
(q) $|\eta_{17} - \eta_{17}|/\eta_{17}$, NRMSE($\eta_{17}$) = 9.5E-03
(r) $|\eta_{18} - \eta_{18}|/\eta_{18}$, NRMSE($\eta_{18}$) = 1.1E-02
(s) $|\eta_{19} - \eta_{19}|/\eta_{19}$, NRMSE($\eta_{19}$) = 5.3E-03
(t) $|\eta_{20} - \eta_{20}|/\eta_{20}$, NRMSE($\eta_{20}$) = 8.2E-03
(u) $|\eta_{21} - \eta_{21}|/\eta_{21}$, NRMSE($\eta_{21}$) = 1.8E-02
in advance, and aim to design a fast sampling method to reduce the time of computing graphlet counts. When the entire graph is given in advance, Minfer [53] and crawling methods in [39,51,54] exhibit much larger errors than sampling methods such as our method MOSS-5 (aim to estimate 5-node graphlet counts), 3-path sampling [27] (aim to estimate 4-node graphlet counts), and wedge sampling [25] (aim to estimate 3-node graphlet counts) under the same computational time. In contrary, these sampling methods require the statistics of all nodes and edges such as degree, so they cannot be used to estimate graphlet counts when the entire graph is not given in advance. In addition, [55,56] accelerate the speed of exactly counting graphlets, and [57] develops a parallel algorithm to exactly count 3- and 4-node graphlets.

6 Conclusions and Future Work

We develop a computationally efficient sampling method MOSS-5 to estimate the counts of 5-node graphlets in a large graph. We provide unbiased estimators of 5-node graphlet counts, and derive simple yet exact formulas for the variances of the estimators. Meanwhile, we conduct experiments on a variety of publicly available datasets, and experimental results demonstrate that our method significantly outperforms the state-of-the-art methods. In future, we plan to extend MOSS-5 on parallel and distributed computing systems for greater scalability.

Acknowledgment

The authors wish to thank the anonymous reviewers for their helpful feedback. In addition, the authors also wish to thank Mr. Yiyan Qi and Miss Xiaotong Ren for discussions. This work was supported in part by Army Research Office Contract W911NF-12-1-0385, and ARL under Cooperative Agreement W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the ARL, or the U.S. Government. The research presented in this paper is supported in part by National Natural Science Foundation of China (U1301254, 61603290, 61602371), the Ministry of Education&China Mobile Research Fund (MCM20160311), the Natural Science Foundation of Jiangsu Province (SBK2014021758), 111 International Collaboration Program of China, the Prospective Joint Research of Industry-Academia-Research Joint Innovation Funding of Jiangsu Province (BY2014074), Shenzhen Basic Research Grant (JCYJ20160229195940462), China Postdoctoral Science Foundation (2015M582663), Natural Science Basic Research Plan in Shaanxi Province of China (2016JQ6034). Junzhou Zhao is the corresponding author.

References

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2756836, IEEE Transactions on Knowledge and Data Engineering


Zhenguo Li received the B.S. and M.S. degrees from the Department of Mathematics at Peking University, in 2002 and 2005, respectively, and the Ph.D. degree from the Department of Information Engineering at the Chinese University of Hong Kong, in 2008. He was an associate research scientist in the Department of Electrical Engineering at Columbia University. He is currently a senior researcher at Huawei Noah’s Ark Lab in Hong Kong. His research interests include machine learning and artificial intelligence.

Don Towsley holds a B.A. in Physics (1971) and a Ph.D. in Computer Science (1975) from University of Texas. From 1976 to 1985, he was a member of the faculty of the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst. He is currently a Distinguished Professor at the University of Massachusetts in the Department of Computer Science. His research interests include network measurement and modeling. He has held visiting positions at IBM T.J. Watson Research Center, Yorktown Heights, NY; Laboratoire MASI, Paris, France; INRIA, Sophia-Antipolis, France; AT&T Labs-Research, Florham Park, NJ; and Microsoft Research Lab, Cambridge, UK. His research interests include networks and performance evaluation. He served as Editor-in-Chief of IEEE/ACM Transactions on Networking and on the editorial boards of Journal of the ACM, IEEE Journal on Selected Areas in Communications, and numerous other editorial boards. He was Program Co-chair of the joint ACM SIGMETRICS and PERFORMANCE 92 conference and the Performance 2002 conference. He is a member of ACM and ORSA, and Chair of IFIP Working Group 7.3. He has received the 2007 IEEE Koji Kobayashi Award, the 2007 ACM SIGMETRICS Achievement Award, the 1998 IEEE Communications Society William Bennett Best Paper Award, and numerous best conference/workshop paper awards. Last, he has been elected Fellow of both the ACM and IEEE.

Jiefeng Cheng obtained his PhD from the Chinese University of Hong Kong in 2007. From 2007 to 2010, he worked as a research fellow at the Chinese University of Hong Kong and the Hong Kong University, respectively. He is currently an expert engineer with Tencent Cloud Security Lab in Shenzhen, China. Prior to joining Tencent, he was a senior researcher at Huawei Noah’s Ark Lab in Hong Kong and a faculty member at the University of Chinese Academy of Sciences in China. His research interests include graph mining and management.

Jing Tao received the B.S and M.S degrees in automatic control from Xi’an Jiaotong University, Xi’an, China, in 2001 and 2006 respectively. He is currently a teacher in Xi’an Jiaotong University and on-the-job Ph.D. candidate with the Systems Engineering Institute and SKLMS Laboratory, Xi’an Jiaotong University under the supervision of Prof. Xiaohong Guan. His research interests include Internet traffic measurement and modeling, traffic classification, abnormal detection, and botnet.

John C.S. Lui received the PhD degree in computer science from UCLA. He is currently a professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong. His current research interests include communication networks, network/system security, network economics, network sciences, cloud computing, large-scale distributed systems and performance evaluation theory. He serves in the editorial board of IEEE/ACM Transactions on Networking, IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Journal of Performance Evaluation and International Journal of Network Security. He was the chairman of the CSE Department from 2005-2011. He received various departmental teaching awards and the CUHK Vice-Chancellor’s Exemplary Teaching Award. He is also a coreipient of the IFIP WG 7.3 Performance 2005 and IEEE/IFIP NOMS 2006 Best Student Paper Awards. He is an elected member of the IFIP WG 7.3, fellow of the ACM, fellow of the IEEE, and croucher senior research fellow. His personal interests include films and general reading.

Xiaohong Guan received the B.S. and M.S. degrees in automatic control from Tsinghua University, Beijing, China, in 1982 and 1985, respectively, and the Ph.D. degree in electrical engineering from the University of Connecticut, Storrs, US, in 1993. He is currently a professor at the Systems Engineering Institute, Xi’an Jiaotong University, Xi’an, China. From 1993 to 1995, he was a consulting engineer at PG&E. From 1985 to 1988, he was with the Systems Engineering Institute, Xi’an Jiaotong University, Xi’an, China. From January 1999 to February 2000, he was with the Division of Engineering and Applied Science, Harvard University, Cambridge, MA. Since 1995, he has been with the Systems Engineering Institute, Xi’an Jiaotong University, and was appointed Cheung Kong Professor of Systems Engineering in 1999, and dean of the School of Electronic and Information Engineering in 2008. Since 2001 he has been the director of the Center for Intelligent and Networked Systems, Tsinghua University, and served as head of the Department of Automation, 2003-2008. He is an Editor of IEEE Transactions on Power Systems and an Associate Editor of Automata. His research interests include allocation and scheduling of complex networked resources, network security, and sensor networks. He has been elected Fellow of IEEE.