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, *Fellow, IEEE*, Don Towsley, *Fellow, IEEE*, Jing Tao, and Xiaohong Guan, *Fellow, IEEE*

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 graphlet counts for a large graph or a large set of graphs due to the combinatorial nature of the problem. Despite recent efforts in counting 3-node 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 a fast sampling method and unbiased estimators of graphlet counts, but also derive simple yet exact formulas for the variances of the estimators which are 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.

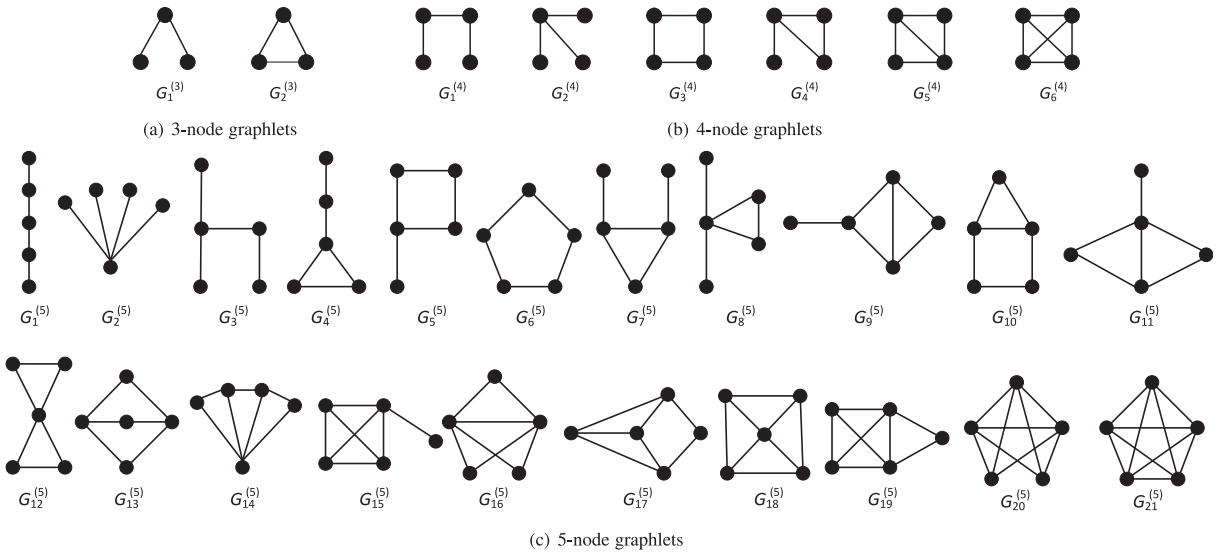
Index Terms—Graphlet kernel, subgraph sampling, graph mining

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], [4], [5], [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 an 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

- *P. Wang is with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xianning West Road, Xi'an, Shaanxi 710049, China, and the Shenzhen Research Institute, Xi'an Jiaotong University, Shenzhen 518057, China. E-mail: phwang@mail.xjtu.edu.cn.*
- *J. Zhao and J.C.S. Lui are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong, Mainland China. E-mail: jzzhao@sei.xjtu.edu.cn, cslui@cse.cuhk.edu.hk.*
- *X. Zhang is with the King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia. E-mail: xiangliang.zhang@kaust.edu.sa.*
- *Z. Li is with the Huawei Noah's Ark Lab, Shatin, Hong Kong. E-mail: li.zhenguo@huawei.com.*
- *J. Cheng is with the Tencent Cloud Security Lab, Shenzhen 518057, China. E-mail: geoffcheng@tencent.com.*
- *D. Towsley is with the Department of Computer Science, University of Massachusetts, Amherst, MA 01003. E-mail: towsley@cs.umass.edu.*
- *J. Tao is with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xianning West Road, Xi'an, Shaanxi 710049, China. E-mail: jtao@mail.xjtu.edu.cn.*
- *X. Guan is with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xianning West Road, Xi'an, Shaanxi 710049, China, and the Center for Intelligent and Networked Systems, Tsinghua National Lab for Information Science and Technology, Tsinghua University, Beijing, China. E-mail: xhguan@mail.xjtu.edu.cn.*

Manuscript received 14 Jan. 2017; revised 13 Aug. 2017; accepted 14 Sept. 2017. Date of publication 26 Sept. 2017; date of current version 5 Dec. 2017. (Corresponding author: Junzhou Zhao.)
Recommended for acceptance by K. Yi.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2017.2756836



(a) 3-node graphlets (b) 4-node graphlets

(c) 5-node graphlets

Fig. 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, $\frac{1}{6}d_v(d_v-1)(d_v-2)$ 4-node CISes, and $\frac{1}{24}d_v(d_v-1)(d_v-2)(d_v-3)$ 5-node CISes.

the node classes (e.g., the field of researchers). Similarly, they represent each online discussion thread on OSN Reddit[1] as a graph where nodes correspond to users and there exists 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 the existing malicious code to generate new malware variants, (b) use repackaging techniques to inject a small piece of malicious code into popular mobile Apps such as Angry Bird. Meanwhile, they can easily avoid traditional detectors based on pure syntax. Recently, [16], [17], [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.

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, 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 should satisfy that it can stop as soon as possible when 1) it achieves the required accuracy or 2) the sampling budget runs out, and then returns 1) graphlet count estimates and 2) estimation errors.

Despite recent progress in counting triangles [21], [22], [23], [24], [25], [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] propose 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 is scalable and computationally efficient.
- To validate our method, we perform an in-depth analysis to demonstrate the accuracy of our method.

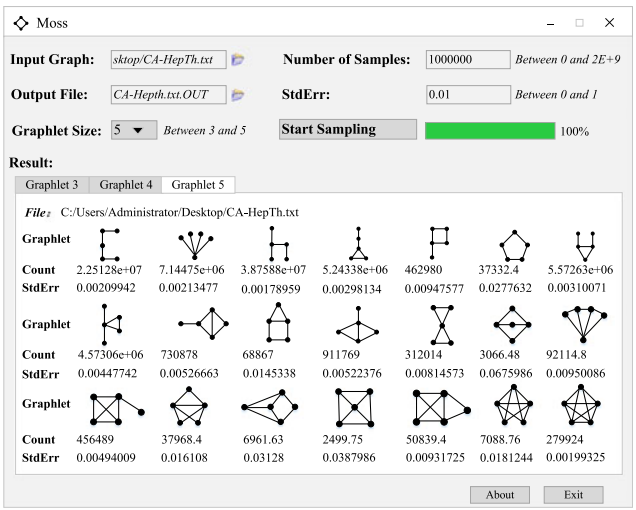


Fig. 2. The graphical user interface of our system.

1. http://www.reddit.com

TABLE 1
Table of Notation

$G = (V, E)$	G 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
$G^{(5)}(s)$	5-node graphlet ID of CIS s
$C^{(5)}$	the set of all 5-node CISes in G
$C_1^{(5)}, \ldots, C_{21}^{(5)}$	$C_i^{(5)}$ is the set of 5-node CISes in G isomorphic to graphlet $G_i^{(5)}$, $1 \le i \le 21$
$\eta_1, \ldots, \eta_{21}$	$\eta_i = \|C_i^{(5)}\|$, $1 \le i \le 21$
K_1, K_2	sampling budgets of T-5 and Path-5
$K = K_1 + K_2$	sampling budgets of MOSS-5
$\phi_i^{(1)}, 1 \le i \le 21$	the number of subgraphs in a CIS $s \in C_i^{(5)}$ that are isomorphic to $G_3^{(5)}$
$\phi_i^{(2)}, 1 \le i \le 21$	the number of subgraphs in a CIS $s \in C_i^{(5)}$ that are isomorphic to $G_1^{(5)}$
$\phi_i^{(3)}, 1 \le i \le 21$	the number of subgraphs in a CIS $s \in C_i^{(5)}$ that are isomorphic to $G_2^{(5)}$

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.[2]

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 the related work. Concluding remarks then follow.

2 Problem Formulation

Let $G = (V, E)$ be an undirected graph, where V and E are the node set and edge set respectively. To define graphlet counts of G, let us first introduce some notation. A subgraph G' of G is a graph of which node set and edge set are both subsets of V and E respectively. 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., $V' \subset V$, $E' = \{(u, v) : u, v \in V', (u, v) \in E\}$. *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 $C^{(5)}$ the set of 5-node CISes in G, and $C_i^{(5)}$ the set of 5-node CISes in G isomorphic[3] to graphlet $G_i^{(5)}$. The graphlet count of $G_i^{(5)}$ is defined as $\eta_i = |C_i^{(5)}|$, $1 \le i \le 21$. As we 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.

2. http://nskeylab.xjtu.edu.cn/dataset/phwang/code/mosscode.zip

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 $f : V_1 \to V_2$ 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.

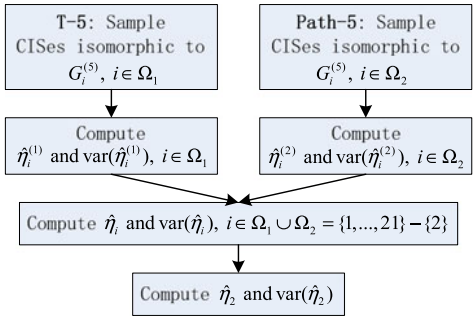


Fig. 3. Overview of MOSS-5. MOSS-5 smartly combines two T-5 and Path-5 sampling methods, which are two novel sampling methods proposed in this paper.

3 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 at least one subgraph isomorphic to graphlet $G_3^{(5)}$; 2) except CISes in $C_2^{(5)} \cup C_3^{(5)} \cup C_8^{(5)}$, 5-node CISes include at least one subgraph isomorphic to graphlet $G_1^{(5)}$. Let $\Omega_1 = \{1, \ldots, 21\} - \{1, 2, 6\}$ and $\Omega_2 = \{1, \ldots, 21\} - \{2, 3, 8\}$. 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 η_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 η_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 $\hat{\eta}_k$ of η_k, $k \in \Omega_1 \cup \Omega_2 = \{1, \ldots, 21\} - \{2\}$ and provide an unbiased estimator $\hat{\eta}_2$ of η_2. To bound the error of $\hat{\eta}_k$, $k \in \{1, \ldots, 21\}$, we also derive the variance of each $\hat{\eta}_k$.

3.1 The T-5 Sampling Method

Denote

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

We assign a weight $\Gamma_v^{(1)}$ to each node $v \in V$. Define $\Gamma^{(1)} = \sum_{v \in V} \Gamma_v^{(1)}$ and $\rho_v^{(1)} = \frac{\Gamma_v^{(1)}}{\Gamma^{(1)}}$. 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

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

TABLE 2
Values of $\phi_i^{(1)}$, $\phi_i^{(2)}$, and $\phi_i^{(3)}$

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$\phi_i^{(1)}$	0	0	1	1	2	0	2	2	4	4	5	4	6	10	9	12	10	20	20	36	60
$\phi_i^{(2)}$	1	0	0	2	2	5	1	0	4	7	2	4	6	10	6	6	14	24	18	36	60
$\phi_i^{(3)}$	0	1	0	0	0	0	0	1	0	0	1	1	0	1	1	2	0	1	2	3	5

Step 3. Sample a node w from $N_v - \{u\}$ at random;
Step 4. Sample a node r from $N_v - \{u, w\}$ at random;
Step 5. Sample a node t from $N_u - \{v\}$ 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 N_v uniformly. In contrast, sampling v and u according to specific distributions $\rho^{(1)}$ and $\sigma^{(v)}$ 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_1 times to obtain K_1 CISes $s_1^{(1)}, \ldots, s_{K_1}^{(1)}$. The pseudo-code of T-5 is shown in Algorithm 1. In Algorithm 1, function $\mathrm{WeightRandomVertex}(V, \rho^{(1)})$ returns a node sampled from V according to the distribution $\rho^{(1)} = \{\rho_v^{(1)} : v \in V\}$, function $\mathrm{RandomVertex}(X)$ returns a node sampled from X at random, and function $\mathrm{CIS}(\{v, u, w, r, t\})$ returns the CIS with the node set $\{v, u, w, r, t\}$ in G.

Algorithm 1. The Pseudo-Code of T-5

input: $G = (V, E)$ and K_1.
output: $\hat{\eta}_i^{(1)}$.
for $i \in \Omega_1$ **do**
  $\hat{\eta}_i^{(1)} \leftarrow 0$;
end
for $k \in [1, K_1]$ **do**
  $v \leftarrow \mathrm{WeightRandomVertex}(V, \rho^{(1)})$;
  $u \leftarrow \mathrm{WeightRandomVertex}(N_v, \sigma^{(v)})$;
  $w \leftarrow \mathrm{RandomVertex}(N_v - \{u\})$;
  $r \leftarrow \mathrm{RandomVertex}(N_v - \{u, w\})$;
  $t \leftarrow \mathrm{RandomVertex}(N_u - \{v\})$;
  $s_k^{(1)} \leftarrow \mathrm{CIS}(\{v, u, w, r, t\})$;
  if $t \neq w$ and $t \neq r$ **thenz**
    $i \leftarrow G^{(5)}(s_k^{(1)})$;
    $\hat{\eta}_i^{(1)} \leftarrow \hat{\eta}_i^{(1)} + \frac{1}{K_1 p_i^{(1)}}$;
  end
end

For a CIS s isomorphic to graphlet $G_i^{(5)}$, $1 \leq i \leq 21$, denote $\phi_i^{(1)}$ as the number of subgraphs in s that are isomorphic to graphlet $G_3^{(5)}$. In other word, s contains $\phi_i^{(1)}$ subgraphs that are isomorphic to $G_3^{(5)}$. 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 η_i.

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

$$p_i^{(1)} = \frac{2\phi_i^{(1)}}{\Gamma^{(1)}}, \qquad 1 \leq i \leq 21.$$

Proof. As shown in Fig. 4, we can easily find that there exist two ways to sample a subgraph isomorphic to graphlet $G_3^{(5)}$ by T-5. Each happens with probability

$$\rho_v^{(1)} \times \sigma_u^{(v)} \times \frac{1}{d_v - 1} \times \frac{1}{d_v - 2} \times \frac{1}{d_u - 1} = \frac{1}{\Gamma^{(1)}}.$$

For a 5-node CIS $s \in C_i^{(5)}$, s has $\phi_i^{(1)}$ different subgraphs isomorphic to graphlet $G_3^{(5)}$, $1 \leq i \leq 21$. Therefore, the probability of T-5 sampling s is $\frac{2\phi_i^{(1)}}{\Gamma^{(1)}}$. □

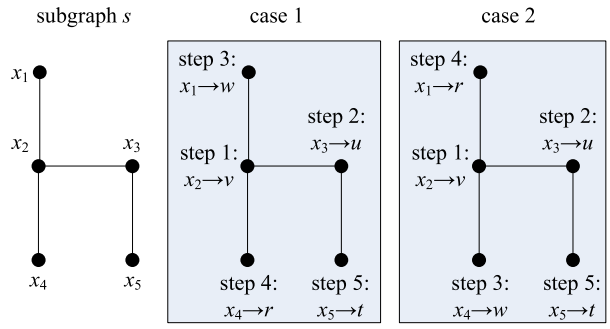


Fig. 4. The ways of T-5 sampling a subgraph s isomorphic to graphlet $G_3^{(5)}$, where v, u, w, r, and t are the variables in Algorithm 1, i.e., the nodes sampled at the 1^{st}, 2^{nd}, 3^{rd}, 4^{th}, and 5^{th} steps, respectively.

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

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

It is easy to obtain the expectation of $m_i^{(1)}$ as

$$\mathbb{E}(m_i^{(1)}) = K_1 p_i^{(1)} \eta_i.$$

For $i \in \Omega_1$, $p_i^{(1)}$ is larger than zero and thus we estimate η_i as

$$\hat{\eta}_i^{(1)} = \frac{m_i^{(1)}}{K_1 p_i^{(1)}}.$$

Theorem 2. *For $i \in \Omega_1$, $\hat{\eta}_i^{(1)}$ is an unbiased estimator of η_i, i.e., $\mathbb{E}(\hat{\eta}_i^{(1)}) = \eta_i^{(1)}$, and the variance of $\hat{\eta}_i^{(1)}$ is*

$$\mathrm{Var}(\hat{\eta}_i^{(1)}) = \frac{\eta_i}{K_1}\left(\frac{1}{p_i^{(1)}} - \eta_i\right). \tag{1}$$

We estimate $\mathrm{Var}(\hat{\eta}_i^{(1)})$ by replacing η_i with $\hat{\eta}_i^{(1)}$ in (1). We also compute the covariance of $\hat{\eta}_i^{(1)}$ and $\hat{\eta}_j^{(1)}$ as follows,

$$\mathrm{Cov}(\hat{\eta}_i^{(1)}, \hat{\eta}_j^{(1)}) = -\frac{\eta_i \eta_j}{K_1}, \quad i \neq j, \quad i, j \in \Omega_1,$$

which is used to compute the variance of the estimate of η_2 given in Section 3.3.

Proof. For $i \in \Omega_1$ and $1 \leq k \leq K_1$, we have

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

Since $s_1^{(1)}, \ldots, s_{K_1}^{(1)}$ 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

$$\mathbb{E}(m_i^{(1)}) = K_1 p_i^{(1)} \eta_i, \mathrm{Var}(m_i^{(1)}) = K_1 p_i^{(1)} \eta_i (1 - p_i^{(1)} \eta_i).$$

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

$$\mathbb{E}(\hat{\eta}_i^{(1)}) = \mathbb{E}\left(\frac{m_i^{(1)}}{K_1 p_i^{(1)}}\right) = \frac{\mathbb{E}(m_i^{(1)})}{K_1 p_i^{(1)}} = \eta_i,$$
$$\mathrm{Var}(\hat{\eta}_i) = \mathrm{Var}\left(\frac{m_i^{(1)}}{K_1 p_i^{(1)}}\right) = \frac{\eta_i}{K_1}\left(\frac{1}{p_i^{(1)}} - \eta_i\right).$$

□

For $i \neq j$ and $i, j \in \Omega_1$, the covariance of $\hat{\eta}_i^{(1)}$ and $\hat{\eta}_j^{(1)}$ is

$$\begin{aligned}
&\mathrm{Cov}(\hat{\eta}_i^{(1)}, \hat{\eta}_j^{(1)}) \\
&= \mathrm{Cov}\left(\frac{m_i^{(1)}}{K_1 p_i^{(1)}}, \frac{m_j^{(1)}}{K_1 p_j^{(1)}}\right) \\
&= \frac{\mathrm{Cov}(\sum_{k=1}^{K_1} \mathbf{1}(G^{(5)}(s_k^{(1)}) = i), \sum_{l=1}^{K_1} \mathbf{1}(G^{(5)}(s_l^{(1)}) = j))}{K_1^2 p_i^{(1)} p_j^{(1)}} \\
&= \frac{\sum_{k=1}^{K_1} \sum_{l=1}^{K_1} \mathrm{Cov}(\mathbf{1}(G^{(5)}(s_k^{(1)}) = i), \mathbf{1}(G^{(5)}(s_l^{(1)}) = j))}{K_1^2 p_i^{(1)} p_j^{(1)}}.
\end{aligned}$$

Each sampled 5-node CIS is obtained independently, so we have

$$\mathrm{Cov}(\mathbf{1}(G^{(5)}(s_k^{(1)}) = i), \mathbf{1}(G^{(5)}(s_l^{(1)}) = j)) = 0, \quad k \neq l.$$

In addition, we have $P(G^{(5)}(s_k^{(1)}) = i \wedge G^{(5)}(s_k^{(1)}) = j) = 0$ when $i \neq j$. Therefore, we obtain

$$\begin{aligned}
&\mathrm{Cov}(\mathbf{1}(G^{(5)}(s_k^{(1)}) = i), \mathbf{1}(G^{(5)}(s_k^{(1)}) = j)) \\
&= \mathbb{E}(\mathbf{1}(G^{(5)}(s_k^{(1)}) = i)\mathbf{1}(G^{(5)}(s_k^{(1)}) = j)) \\
&\quad - \mathbb{E}(\mathbf{1}(G^{(5)}(s_k^{(1)}) = i))\mathbb{E}(\mathbf{1}(G^{(5)}(s_k^{(1)}) = j)) \\
&= 0 - p_i^{(1)} \eta_i p_j^{(1)} \eta_j \\
&= -p_i^{(1)} p_j^{(1)} \eta_i \eta_j.
\end{aligned}$$

Now, we easily have

$$\begin{aligned}
&\mathrm{Cov}(\hat{\eta}_i^{(1)}, \hat{\eta}_j^{(1)}) \\
&= \frac{\sum_{k=1}^{K_1} \mathrm{Cov}(\mathbf{1}(G^{(5)}(s_k^{(1)}) = i), \mathbf{1}(G^{(5)}(s_k^{(1)}) = j))}{K_1^2 p_i^{(1)} p_j^{(1)}} \\
&= -\frac{\eta_i \eta_j}{K_1}.
\end{aligned}$$

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 $\rho^{(2)} = \{\rho_v^{(2)} : v \in V\}$;

Step 2. Sample a node u from N_v according to the distribution $\tau^{(v)} = \{\tau_u^{(v)} : u \in N_v\}$, where we define

$$\tau_u^{(v)} = \frac{(d_u - 1)(\sum_{y \in N_v - \{u\}} (d_y - 1))}{\Gamma_v^{(2)}}, \quad u \in N_v;$$

Step 3. Sample a node w from $N_v - \{u\}$ according to the distribution $\mu^{(v,u)} = \{\mu_w^{(v,u)} : w \in N_v - \{u\}\}$, where we define

$$\mu_w^{(v,u)} = \frac{d_w - 1}{\sum_{y \in N_v - \{u\}} (d_y - 1)}, \quad w \in N_v - \{u\};$$

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, and 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**
  $v \leftarrow \mathrm{WeightRandomVertex}(V, \rho^{(2)})$;
  $u \leftarrow \mathrm{WeightRandomVertex}(N_v, \tau^{(v)})$;
  $w \leftarrow \mathrm{WeightRandomVertex}(N_v - \{u\}, \mu^{(v,u)})$;
  $r \leftarrow \mathrm{RandomVertex}(N_u - \{v\})$;
  $t \leftarrow \mathrm{RandomVertex}(N_w - \{v\})$;
  $s_k^{(2)} \leftarrow \mathrm{CIS}(\{v, u, w, r, t\})$;
  if $t \neq u$ and $r \neq w$ and $t \neq r$ **then**
    $i \leftarrow G^{(5)}(s_k^{(2)})$;
    $\hat{\eta}_i^{(2)} \leftarrow \hat{\eta}_i^{(2)} + \frac{1}{K_2 p_i^{(2)}}$;
  end
end

We run the above procedure K_2 times to obtain K_2 CISes $s_1^{(2)}, \ldots, s_{K_2}^{(2)}$. For a CIS s isomorphic to graphlet $G_i^{(5)}$, $1 \leq i \leq 21$, let $\phi_i^{(2)}$ denote the number of subgraphs in s that are isomorphic to $G_1^{(5)}$. In other word, s contains $\phi_i^{(2)}$ subgraphs that are isomorphic to $G_1^{(5)}$. The value of $\phi_i^{(2)}$ 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*

$$p_i^{(2)} = \frac{2\phi_i^{(2)}}{\Gamma^{(2)}}, \qquad 1 \leq i \leq 21.$$

Proof. As shown in Fig. 5, we can see that there exist two ways to sample a subgraph isomorphic to graphlet $G_1^{(5)}$ by Path-5. Each one happens with probability

$$\rho_v^{(2)} \times \tau_u^{(v)} \times \mu_w^{(v,u)} \times \frac{1}{d_u - 1} \times \frac{1}{d_w - 1} = \frac{1}{\Gamma^{(2)}}.$$

For a 5-node CIS $s \in C_i^{(5)}$, s has $\phi_i^{(2)}$ subgraphs isomorphic to graphlet $G_1^{(5)}$, $1 \leq i \leq 21$. Thus, the probability of Path-5 sampling s is $\frac{2\phi_i^{(2)}}{\Gamma^{(2)}}$. □

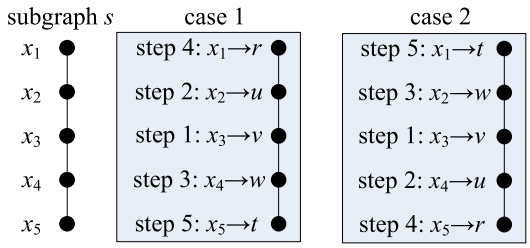


Fig. 5. The ways of Path-5 sampling a subgraph s isomorphic to graphlet $G_1^{(5)}$, where v, u, w, r, and t are the variables in Algorithm 2, i.e., the nodes sampled at the 1^{st}, 2^{nd}, 3^{rd}, 4^{th}, and 5^{th} steps, respectively.

Denote

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

Then, we have

$$\mathbb{E}(m_i^{(2)}) = K_2 p_i^{(2)} \eta_i.$$

For $i \in \Omega_2$, $p_i^{(2)}$ is larger than zero and we then estimate η_i as

$$\hat{\eta}_i^{(2)} = \frac{m_i^{(2)}}{K_2 p_i^{(2)}}.$$

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

$$\mathrm{Var}(\hat{\eta}_i^{(2)}) = \frac{\eta_i}{K_2}\left(\frac{1}{p_i^{(2)}} - \eta_i\right). \tag{2}$$

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

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

which is used to compute the variance of the estimate of η_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 η_i as $\hat{\eta}_i^{(1)}$ and $\hat{\eta}_i^{(2)}$ for $i \in \Omega_1 - \Omega_2$ and $i \in \Omega_2 - \Omega_1$ respectively. When $i \in \Omega_1 \cap \Omega_2$, according to [29], we estimate η_i based on its two unbiased estimates $\hat{\eta}_i^{(1)}$ and $\hat{\eta}_i^{(2)}$. Formally, let

$$\lambda_i^{(1)} = \frac{\mathrm{Var}(\hat{\eta}_i^{(2)})}{\mathrm{Var}(\hat{\eta}_i^{(1)}) + \mathrm{Var}(\hat{\eta}_i^{(2)})}, \lambda_i^{(2)} = \frac{\mathrm{Var}(\hat{\eta}_i^{(1)})}{\mathrm{Var}(\hat{\eta}_i^{(1)}) + \mathrm{Var}(\hat{\eta}_i^{(2)})}.$$

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

$$\hat{\eta}_i = \begin{cases} \lambda_i^{(1)} \hat{\eta}_i^{(1)} + \lambda_i^{(2)} \hat{\eta}_i^{(2)}, & i \in \Omega_1 \cap \Omega_2, \\ \hat{\eta}_i^{(1)}, & i \in \Omega_1 - \Omega_2, \\ \hat{\eta}_i^{(2)}, & i \in \Omega_2 - \Omega_1. \end{cases} \tag{3}$$

Note that $\Omega_1 \cup \Omega_2 = \{1, 2, \ldots, 21\} - \{2\}$. Next, we discuss our method for estimating η_2. For a CIS s isomorphic to graphlet $G_i^{(5)}$, $1 \leq i \leq 21$, let $\phi_i^{(3)}$ denote the number of subgraphs in s that are isomorphic to graphlet $G_2^{(5)}$. In other word, s contains $\phi_i^{(3)}$ subgraphs that are isomorphic to $G_2^{(5)}$. The value of $\phi_i^{(3)}$ is given in Table 2. Let

$$\Lambda_4 = \sum_{v \in V} \binom{d_v}{4}.$$

Then, the number of all 5-node subgraphs (not necessarily induced) in G isomorphic to graphlet $G_2^{(5)}$ is Λ_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 η_2 as

$$\hat{\eta}_2 = \Lambda_4 - \sum_{i \in \Omega_3^*} \phi_i^{(3)} \hat{\eta}_i.$$

Theorem 5. *$\hat{\eta}_i$ is an unbiased estimator of η_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*

$$\mathrm{Var}(\hat{\eta}_i) = \begin{cases} \frac{\mathrm{Var}(\hat{\eta}_i^{(1)})\mathrm{Var}(\hat{\eta}_i^{(2)})}{\mathrm{Var}(\hat{\eta}_i^{(1)}) + \mathrm{Var}(\hat{\eta}_i^{(2)})}, & i \in \Omega_1 \cap \Omega_2, \\ \mathrm{Var}(\hat{\eta}_i^{(1)}), & i \in \Omega_1 - \Omega_2, \\ \mathrm{Var}(\hat{\eta}_i^{(2)}), & i \in \Omega_2 - \Omega_1. \end{cases} \tag{4}$$

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

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

where $\mathrm{Cov}(\hat{\eta}_i, \hat{\eta}_j) =$

$$\begin{cases} -\sum_{l=1,2} \frac{\lambda_i^{(l)} \lambda_j^{(l)} \eta_i \eta_j}{K_l}, & i, j \in \Omega_1 \cap \Omega_2, \\ -\frac{\lambda_j^{(1)} \eta_i \eta_j}{K_1}, & i \in \Omega_1 - \Omega_2, j \in \Omega_1 \cap \Omega_2, \\ -\frac{\lambda_i^{(1)} \eta_i \eta_j}{K_1}, & i \in \Omega_1 \cap \Omega_2, j \in \Omega_1 - \Omega_2, \\ -\frac{\lambda_i^{(2)} \eta_i \eta_j}{K_2}, & i \in \Omega_1 \cap \Omega_2, j \in \Omega_2 - \Omega_1, \\ -\frac{\lambda_j^{(2)} \eta_i \eta_j}{K_2}, & i \in \Omega_2 - \Omega_1, j \in \Omega_1 \cap \Omega_2, \\ 0, & i \in \Omega_1 - \Omega_2, j \in \Omega_2 - \Omega_1, \\ 0, & i \in \Omega_2 - \Omega_1, j \in \Omega_1 - \Omega_2. \end{cases}$$

Proof. For $i \in \Omega_1 \cup \Omega_2$, Theorems 2 and 4 tell us that $\eta_i^{(1)}$ and $\eta_i^{(2)}$ 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

$$\mathbb{E}(\hat{\eta}_2) = \Lambda_4 - \sum_{i \in \Omega_3^*} \phi_i^{(3)} \mathbb{E}(\hat{\eta}_i) = \Lambda_4 - \sum_{i \in \Omega_3^*} \phi_i^{(3)} \eta_i = \eta_2.$$

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 $i, j \in \Omega_1 \cup \Omega_2$, we have $\mathrm{Cov}(\hat{\eta}_i^{(1)}, \hat{\eta}_j^{(2)}) = 0$

because $\hat{\eta}_i^{(1)}$ and $\hat{\eta}_j^{(2)}$ are independent; (II) from Theorem 2, we have $\mathrm{Cov}(\hat{\eta}_i^{(1)}, \hat{\eta}_j^{(1)}) = -\frac{\eta_i \eta_j}{K_1}$, $i \neq j$ and $i, j \in \Omega_1$; (III) from Theorem 4, we have $\mathrm{Cov}(\hat{\eta}_i^{(2)}, \hat{\eta}_j^{(2)}) = -\frac{\eta_i \eta_j}{K_2}$, $i \neq j$ and $i, j \in \Omega_2$. Based on these three equations and eq. (3), we compute $\hat{\eta}_i$ and $\hat{\eta}_j$ as follows:

- When $i \in \Omega_1 - \Omega_2$ and $j \in \Omega_2 - \Omega_1$, we have $\mathrm{Cov}(\hat{\eta}_i, \hat{\eta}_j) = \mathrm{Cov}(\hat{\eta}_i^{(1)}, \hat{\eta}_j^{(2)}) = 0$.
- When $i \in \Omega_2 - \Omega_1$ and $j \in \Omega_1 - \Omega_2$, we have $\mathrm{Cov}(\hat{\eta}_i, \hat{\eta}_j) = \mathrm{Cov}(\hat{\eta}_i^{(2)}, \hat{\eta}_j^{(1)}) = 0$.
- When $i \in \Omega_1 - \Omega_2$ and $j \in \Omega_1 \cap \Omega_2$, we have

$$\mathrm{Cov}(\hat{\eta}_i, \hat{\eta}_j) = \mathrm{Cov}(\hat{\eta}_i^{(1)}, \lambda_j^{(1)} \hat{\eta}_j^{(1)} + \lambda_j^{(2)} \hat{\eta}_j^{(2)}) = -\frac{\lambda_j^{(1)} \eta_i \eta_j}{K_1}.$$

- When $i \in \Omega_1 \cap \Omega_2$ and $j \in \Omega_1 - \Omega_2$, we have

$$\mathrm{Cov}(\hat{\eta}_i, \hat{\eta}_j) = \mathrm{Cov}(\lambda_i^{(1)} \hat{\eta}_i^{(1)} + \lambda_i^{(2)} \hat{\eta}_i^{(2)}, \hat{\eta}_j^{(1)}) = -\frac{\lambda_i^{(1)} \eta_i \eta_j}{K_1}.$$

- When $i \in \Omega_1 \cap \Omega_2$ and $j \in \Omega_2 - \Omega_1$, we have

$$\mathrm{Cov}(\hat{\eta}_i, \hat{\eta}_j) = \mathrm{Cov}(\lambda_i^{(1)} \hat{\eta}_i^{(1)} + \lambda_i^{(2)} \hat{\eta}_i^{(2)}, \hat{\eta}_j^{(2)}) = -\frac{\lambda_i^{(2)} \eta_i \eta_j}{K_2}.$$

- When $i \in \Omega_2 - \Omega_1$ and $j \in \Omega_1 \cap \Omega_2$, we have

$$\mathrm{Cov}(\hat{\eta}_i, \hat{\eta}_j) = \mathrm{Cov}(\hat{\eta}_i^{(2)}, \lambda_j^{(1)} \hat{\eta}_j^{(1)} + \lambda_j^{(2)} \hat{\eta}_j^{(2)}) = -\frac{\lambda_j^{(2)} \eta_i \eta_j}{K_2}.$$

- When $i, j \in \Omega_1 \cap \Omega_2$ and $i \neq j$, we have

$$\begin{aligned}\mathrm{Cov}(\hat{\eta}_i, \hat{\eta}_j) &= \mathrm{Cov}(\lambda_i^{(1)} \hat{\eta}_i^{(1)} + \lambda_i^{(2)} \hat{\eta}_i^{(2)}, \lambda_j^{(1)} \hat{\eta}_j^{(1)} + \lambda_j^{(2)} \hat{\eta}_j^{(2)}) \\ &= -\eta_i \eta_j \left(\frac{\lambda_i^{(1)} \lambda_j^{(1)}}{K_1} + \frac{\lambda_i^{(2)} \lambda_j^{(2)}}{K_2} \right).\end{aligned}$$

Finally, we have $\mathrm{Var}(\hat{\eta}_2) = \mathrm{Var}(\Lambda_4 - \sum_{i \in \Omega_3^*} \phi_i^{(3)} \hat{\eta}_i) = \sum_{i \in \Omega_3^*} (\phi_i^{(3)})^2 \mathrm{Var}(\hat{\eta}_i) + \sum_{i,j \in \Omega_3^*, i \neq j} \phi_i^{(3)} \phi_j^{(3)} \mathrm{Cov}(\hat{\eta}_i, \hat{\eta}_j)$. □

3.4 Implementation and Complexities

In this subsection, we introduce our methods of implementing the functions in Algorithms 1 and 2. We also analyze their time and space complexities.

Function $G^{(5)}(s)$. Let A be the adjacent matrix of s. We compute a bit string by concatenating all elements above the main diagonal of A, that is,

$$str = A_{1,2} || \ldots || A_{1,5} || A_{2,3} || \ldots || A_{2,5} || A_{3,4} || \ldots || A_{3,5} || A_{4,5}.$$

Then, we compute $G^{(5)}(s)$ by looking up key str in hash table HT. Therefore, the average computational complexity of function $G^{(5)}(s)$ is $O(1)$. Hash table HT is generated in advance as: For each graphlet $G_j^{(5)}$, $1 \leq j \leq 13$, we compute a key str for each permutation of nodes in $G_j^{(k)}$, and then store key str and its value j in hash table HT, i.e., $HT[str] = j$. There exist $5! = 120$ different permutations for 5 nodes. Therefore, it is memory and computationally efficient to generate HT in advance.

Functions $\Gamma_v^{(1)}$ *and* $\Gamma_v^{(2)}$. For each node v, we store its degree d_v and use a list to store its neighbors' degrees. Clearly, it requires $O(d_v)$ operations to compute $\Gamma_v^{(1)}$ and $\Gamma_v^{(2)}$. Therefore, the space and time complexities of processing all nodes are both $O(|E|)$.

$\mathrm{WeightRandomVertex}(V, \rho^{(1)})$. We use a list $V[1, \ldots, |V|]$ to store the nodes in V. We store an array $ACC\_\Gamma^{(1)}[1, \ldots, |V|]$ in memory, where $ACC\_\Gamma^{(1)}[i]$ is defined as $ACC\_\Gamma^{(1)}[i] = \sum_{j=1}^{i} \Gamma_{V[j]}^{(1)}$, $1 \leq i \leq |V|$. Note that $ACC\_\Gamma^{(1)}[|V|] = \Gamma^{(1)}$. Let $ACC\_\Gamma^{(1)}[0] = 0$. Then, $\mathrm{WeightRandomVertex}(V, \rho^{(1)})$ 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 $ACC\_\Gamma^{(1)}[i-1] < rnd \leq ACC\_\Gamma^{(1)}[i]$, which is solved by binary search; *step 3*) return $V[i]$. Thus, the space and time complexities of $\mathrm{WeightRandomVertex}(V, \rho^{(1)})$ are $O(|V|)$ and $O(\log |V|)$ respectively.

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

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

$\mathrm{WeightRandomVertex}(N_v, \tau^{(v)})$. It is achieved similarly to that for $\mathrm{WeightRandomVertex}(N_v, \sigma^{(v)})$.

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

$\mathrm{RandomVertex}(N_v - \{u\})$. Function $\mathrm{RandomVertex}(N_v - \{u\})$ 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 $\mathrm{RandomVertex}(N_v - \{u\})$ is $O(1)$.

$\mathrm{RandomVertex}(N_v - \{u, w\})$. 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 $\mathrm{RandomVertex}(N_v - \{u, w\})$ is $O(1)$.

In summary, the space and time complexities of T-5 sampling K_1 CISes are $O(|V| + |E|)$ and $O(|E| + K_1 \log |V|)$ 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.

TABLE 3
Undirected Graph Datasets Used in Our Experiments. Where "max-degree" Represents the Maximum Number of Edges Incident to a Node in the Undirected Graph

Graph	nodes	edges	max-degree
com-Orkut [30]	3,072,441	117,185,803	33,313
LiveJournal [31]	5,189,809	48,688,097	15,017
Pokec [32]	1,632,803	22,301,964	14,854
Flickr [31]	1,715,255	15,555,041	27,236
Xiami [33]	1,753,620	16,018,571	19,727
Wiki-Talk [34]	2,394,385	4,659,565	100,029
Web-Google [35]	875,713	4,322,051	6,332
YouTube [31]	1,138,499	2,990,443	28,754
ca-HepPh [36]	12,008	118,490	491

TABLE 4
Computational Time (Seconds) of MOSS-5

Graph	sampling budget K		
	100,000	1,000,000	10,000,000
com-Orkut	69.1 s	80.4 s	193.8 s
LiveJournal	31.5 s	40.4 s	128.7 s
Pokec	20.7 s	30.2 s	125.2 s
Flickr	10.4 s	19.6 s	111.6 s
Xiami	10.4 s	18.6 s	99.3 s
Wiki-Talk	4.6 s	12.9 s	95.7 s
Web-Google	4.1 s	10.9 s	79.3 s
YouTube	2.5 s	9.4 s	78.6 s
ca-HepPh	0.53 s	4.8 s	46.3 s

3.5 Parameter Settings

From Theorem 5, we can see that the variance of $\hat{\eta}_i$ 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 $\hat{\eta}_i$, $i \in \Omega_2 - \Omega_1$. 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 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 ranged from 0.1 to 117 million edges taken from the Stanford Network Analysis Platform (SNAP),[4] which are summarized in Table 3. We use the state-of-the-art method ESCAPE [28] to exactly compute 5-node graphlet counts $\eta_1, \ldots, \eta_{21}$ for all these graphs. Fig. 6 shows the real values of $\eta_1, \ldots, \eta_{21}$.

4.2 Metric

We study the Normalized Root Mean Square Error (NRMSE) to measure the relative error of the graphlet count estimate $\hat{\eta}_i$ with respect to its true value η_i, $i = 1, \ldots, 21$. It is defined as

$$\mathrm{NRMSE}(\hat{\eta}_i) = \frac{\sqrt{\mathrm{MSE}(\hat{\eta}_i)}}{\eta_i}, \qquad i = 1, , 21,$$

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

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

We can see that $\mathrm{MSE}(\hat{\eta}_i)$ 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 a good estimation performance. When $\hat{\eta}_i$ is an unbiased estimator of η_i, we have $\mathrm{MSE}(\hat{\eta}_i) = \mathrm{Var}(\hat{\eta}_i)$ and then $\mathrm{NRMSE}(\hat{\eta}_i)$ is equivalent to the normalized standard error of $\hat{\eta}_i$, i.e., $\mathrm{NRMSE}(\hat{\eta}_i) = \sqrt{\mathrm{Var}(\hat{\eta}_i)}/\eta_i$. In addition, we define $\frac{|\hat{\eta}_i - \eta_i|}{\eta_i}$ as the relative error of $\hat{\eta}_i$, and also study the CCDF (complementary cumulative distribution function) of $\frac{|\hat{\eta}_i - \eta_i|}{\eta_i}$, that is,

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

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 Opeteron (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 running time. This is because the time cost of computing $\Gamma_v^{(1)}$ and $\Gamma_v^{(2)}$ 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

Fig. 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 $K = 100,000$, and all 5-node graphlets' NRMSEs are smaller than 0.1 when $K = 10,000,000$. 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 $\frac{1}{\sqrt{10}}$. Fig. 8 shows the CCDF of relative error $\frac{|\hat{\eta}_i - \eta_i|}{\eta_i}$, $1 \le i \le 21$, given by MOSS-5 with sampling budget $K = 1,000,000$. We can see that more than 99 percent of estimates $\hat{\eta}_i$ obtained by 1,000 independent runs have a relative error smaller than $3\mathrm{NRMSE}(\hat{\eta}_i)$. From Figs. 6, 7 and 8, we observe that MOSS-5 exhibits smaller estimation errors for graphlets with large graphlet counts (i.e., frequent graphplets) than graphlets with small graphlet counts (i.e., rare graphlets).

4. www.snap.stanford.edu

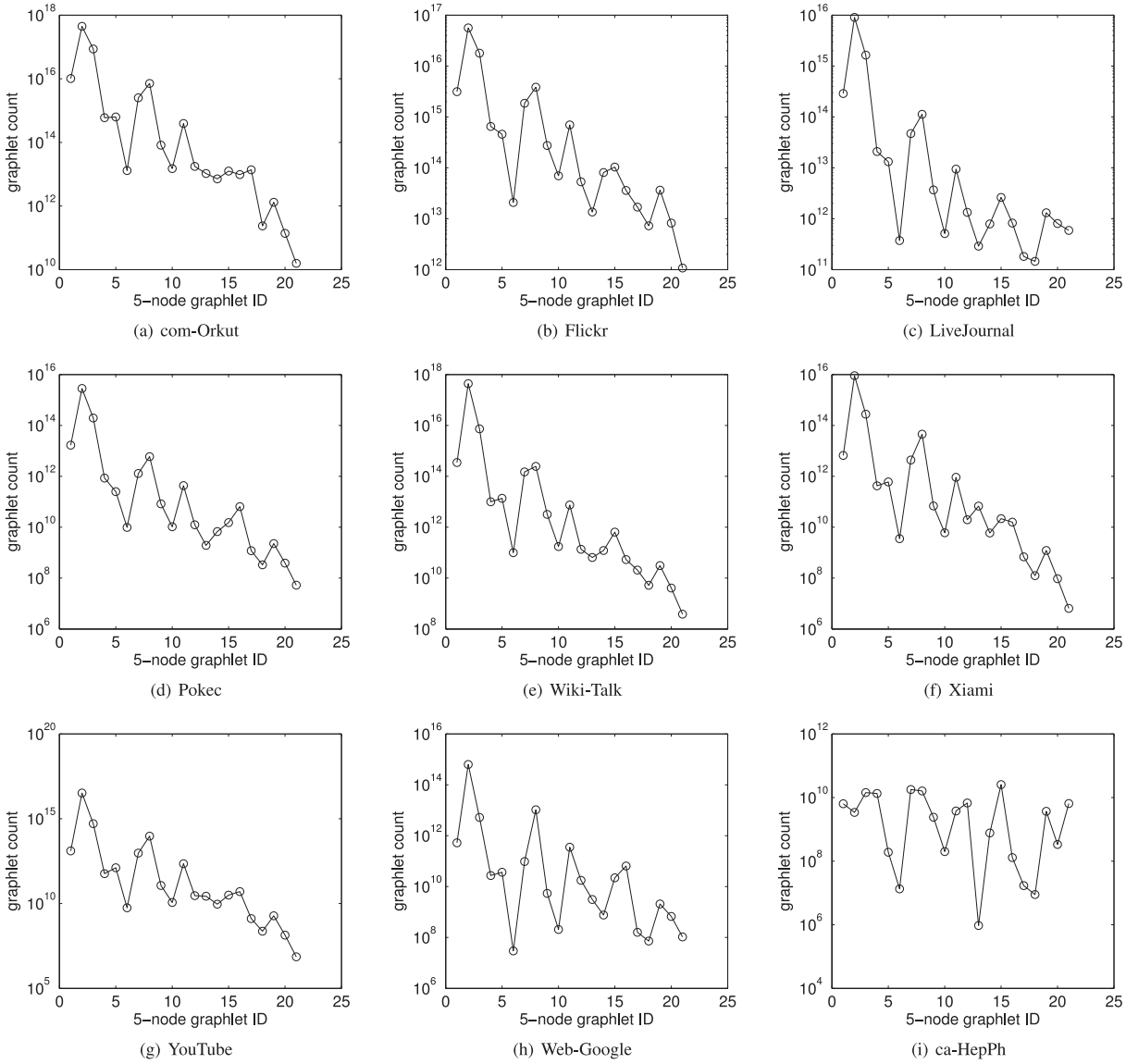


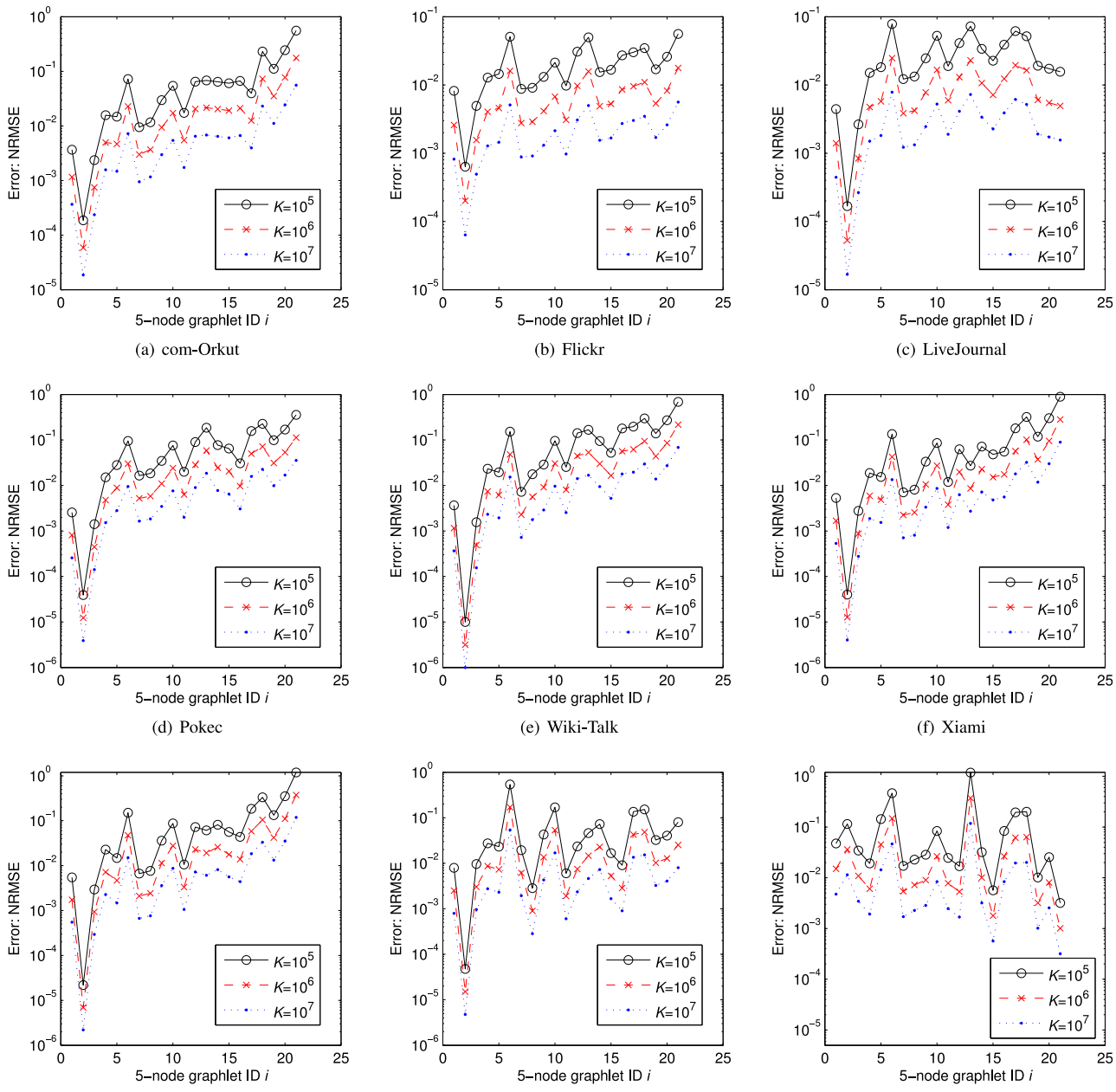
Fig. 6. Real values of 5-node graphlet counts.

4.5 Comparison with Prior Art

MOSS versus 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 Flickr, 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 for 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 $\frac{1}{21}\sum_{i=1}^{21} \mathrm{NRMSE}(\hat{\eta}_i)$ when $\max_{i=1,\ldots,21}\mathrm{NRMSE}(\hat{\eta}_i) = 0.1$. We can see that the average NRMSE varies from 0.01 to 0.04 for all graphs studied in this paper.

TABLE 5
Computational Time (Seconds) and Accuracy of MOSS-5 in Comparison with State-of-the-Art Exact Counting Method ESCAPE

Graph	ESCAPE	MOSS-5, $\max_{i=1,\ldots,21}\mathrm{NRMSE}(\hat{\eta}_i) = 0.1$	
	(time)	time	$\frac{1}{21}\sum_{i=1}^{21} \mathrm{NRMSE}(\hat{\eta}_i)$
Flickr	189,450 s	10 s	0.039
com-Orkut	116,029 s	103 s	0.015
LiveJournal	82,445 s	31 s	0.037
Pokec	3,696 s	31 s	0.024
Wiki-Talk	1,877 s	47 s	0.018
Xiami	518 s	82 s	0.013
Web-Google	112 s	25 s	0.013
YouTube	193 s	96 s	0.011
ca-HepPh	589 s	64 s	0.011



(a) com-Orkut (b) Flickr (c) LiveJournal

(d) Pokec (e) Wiki-Talk (f) Xiami

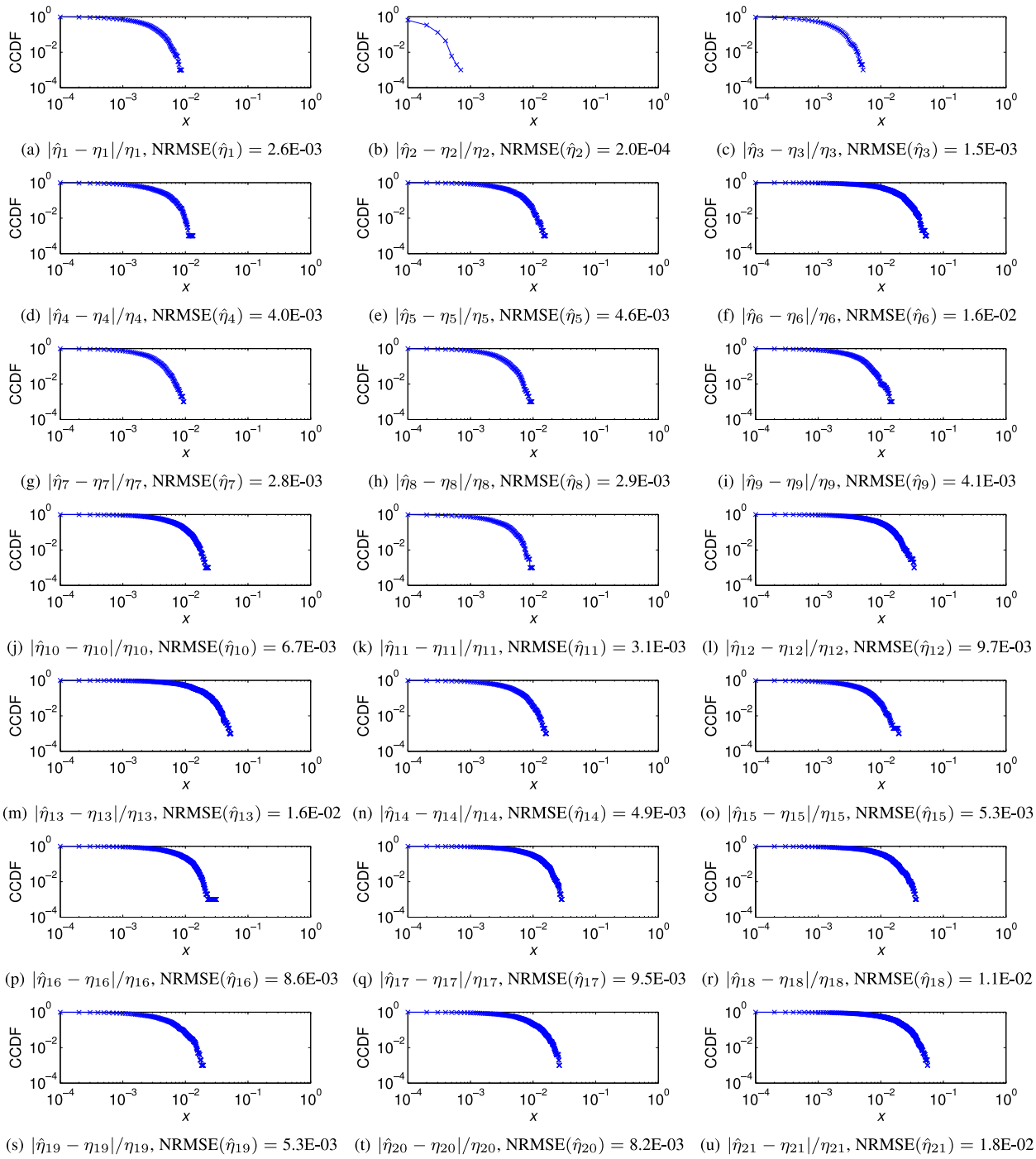
(g) YouTube (h) Web-Google (i) ca-HepPh

Fig. 7. NRMSEs of estimates of 5-node graphlet counts η_i, $1 \leq i \leq 21$, given by MOSS-5 with sampling budget $K = 10^5, 10^6, 10^7$.

MOSS versus Sampling Methods Guise [37] and Graft [38]. Most previous work focuses on estimating 5-node motif concentrations, i.e., $\omega_i = \frac{\eta_i}{\sum_{j=1}^{21} \eta_j}$, $i = 1, \ldots, 21$. 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 percent. 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 percent 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], [42], [43], [44], [45]. Recently, a considerable attention has been given to



(a) $|\hat{\eta}_1 - \eta_1|/\eta_1$, NRMSE$(\hat{\eta}_1) = 2.6\text{E-}03$ (b) $|\hat{\eta}_2 - \eta_2|/\eta_2$, NRMSE$(\hat{\eta}_2) = 2.0\text{E-}04$ (c) $|\hat{\eta}_3 - \eta_3|/\eta_3$, NRMSE$(\hat{\eta}_3) = 1.5\text{E-}03$

(d) $|\hat{\eta}_4 - \eta_4|/\eta_4$, NRMSE$(\hat{\eta}_4) = 4.0\text{E-}03$ (e) $|\hat{\eta}_5 - \eta_5|/\eta_5$, NRMSE$(\hat{\eta}_5) = 4.6\text{E-}03$ (f) $|\hat{\eta}_6 - \eta_6|/\eta_6$, NRMSE$(\hat{\eta}_6) = 1.6\text{E-}02$

(g) $|\hat{\eta}_7 - \eta_7|/\eta_7$, NRMSE$(\hat{\eta}_7) = 2.8\text{E-}03$ (h) $|\hat{\eta}_8 - \eta_8|/\eta_8$, NRMSE$(\hat{\eta}_8) = 2.9\text{E-}03$ (i) $|\hat{\eta}_9 - \eta_9|/\eta_9$, NRMSE$(\hat{\eta}_9) = 4.1\text{E-}03$

(j) $|\hat{\eta}_{10} - \eta_{10}|/\eta_{10}$, NRMSE$(\hat{\eta}_{10}) = 6.7\text{E-}03$ (k) $|\hat{\eta}_{11} - \eta_{11}|/\eta_{11}$, NRMSE$(\hat{\eta}_{11}) = 3.1\text{E-}03$ (l) $|\hat{\eta}_{12} - \eta_{12}|/\eta_{12}$, NRMSE$(\hat{\eta}_{12}) = 9.7\text{E-}03$

(m) $|\hat{\eta}_{13} - \eta_{13}|/\eta_{13}$, NRMSE$(\hat{\eta}_{13}) = 1.6\text{E-}02$ (n) $|\hat{\eta}_{14} - \eta_{14}|/\eta_{14}$, NRMSE$(\hat{\eta}_{14}) = 4.9\text{E-}03$ (o) $|\hat{\eta}_{15} - \eta_{15}|/\eta_{15}$, NRMSE$(\hat{\eta}_{15}) = 5.3\text{E-}03$

(p) $|\hat{\eta}_{16} - \eta_{16}|/\eta_{16}$, NRMSE$(\hat{\eta}_{16}) = 8.6\text{E-}03$ (q) $|\hat{\eta}_{17} - \eta_{17}|/\eta_{17}$, NRMSE$(\hat{\eta}_{17}) = 9.5\text{E-}03$ (r) $|\hat{\eta}_{18} - \eta_{18}|/\eta_{18}$, NRMSE$(\hat{\eta}_{18}) = 1.1\text{E-}02$

(s) $|\hat{\eta}_{19} - \eta_{19}|/\eta_{19}$, NRMSE$(\hat{\eta}_{19}) = 5.3\text{E-}03$ (t) $|\hat{\eta}_{20} - \eta_{20}|/\eta_{20}$, NRMSE$(\hat{\eta}_{20}) = 8.2\text{E-}03$ (u) $|\hat{\eta}_{21} - \eta_{21}|/\eta_{21}$, NRMSE$(\hat{\eta}_{21}) = 1.8\text{E-}02$

Fig. 8. (Flickr) $\mathrm{CCDF}(|\hat{\eta}_i - \eta_i|/\eta_i, x)$, $1 \le i \le 21$, given by MOSS-5 with sampling budget $K = 1{,}000{,}000$.

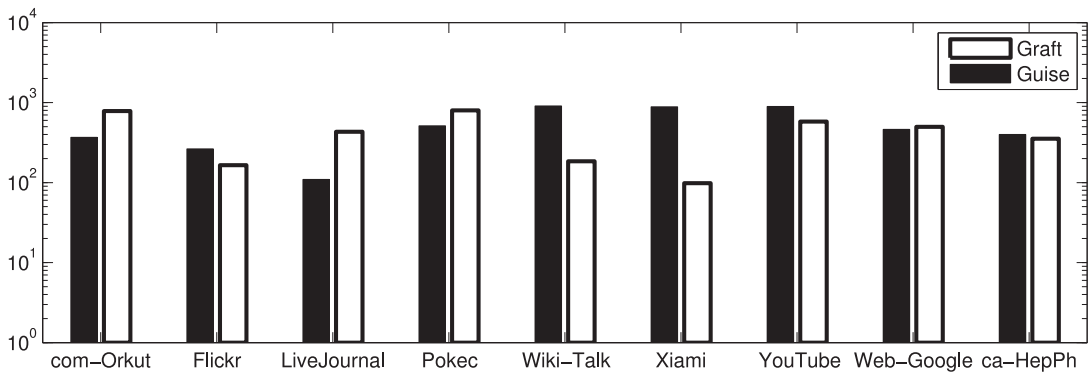


Fig. 9. Runtime of the-state-of-art methods normalized with respect to runtimes of MOSS-5 for estimating 5-node graphlet concentrations.

designing fast algorithms for counting higher order subgraph patterns such as 4- and 5-node graphlets. [28], [44], [45], [46], [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], [38], [39], [48], [49], [50], [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], [22], [23], [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 grahplet counts. When the graph of interest is not available but given a RESampled graph that is obtained by sampling each edge with a fixed probability, Minfer [53] aims to infer the underlying 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 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 [47] 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.

Acknowledgments

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 the 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), the China Postdoctoral Science Foundation (2015M582663), and the Natural Science Basic Research Plan in Shaanxi Province of China (2016JQ6034). Junzhou Zhao is the corresponding author.

References

[1] R. Milo, E. Al, and C. Biology, "Network motifs: Simple building blocks of complex networks," *Sci.*, vol. 298, no. 5549, pp. 824–827, October 2002.

[2] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, "Network motifs in the transcriptional regulation network of escherichia coli," *Nature Genetics*, vol. 31, no. 1, pp. 64–68, May 2002.

[3] H. Chun, Y.-Y. Ahn, H. Kwak, S. Moon, Y. ho Eom, and H. Jeong, "Comparison of online social relations in terms of volume versus interaction: A case study of cyworld," in *Proc. 8th ACM SIGCOMM Conf. Internet Meas.*, Nov. 2008, pp. 57–59.

[4] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The slashdot zoo: Mining a social network with negative edges," in *Proc. 18th Int. Conf. World Wide Web*, Apr. 2009, pp. 741–750.

[5] J. Zhao, J. C. S. Lui, D. Towsley, X. Guan, and Y. Zhou, "Empirical analysis of the evolution of follower network: A case study on douban," in *Proc. NetSciCom*, April 2011, pp. 941–946.

[6] J. Ugander, L. Backstrom, and J. Kleinberg, "Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 1307–1318.

[7] Y. Jin, E. Sharafuddin, and Z.-L. Zhang, "Unveiling core network-wide communication patterns through application traffic activity graph decomposition," in *Proc. 11th Int. Joint Conf. Meas. Modeling Comput. Syst.*, 2009, pp. 49–60.

[8] M. Iliofotou, M. Faloutsos, and M. Mitzenmacher, "Exploiting dynamicity in graph-based traffic analysis: Techniques and applications," in *Proc. 5th Int. Conf. Emerging Netw. Exp. Technol.*, 2009, pp. 241–252.

[9] N. Shervashidze, S. V. N. Vishwanathan, T. Petri, K. Mehlhorn, and K. M. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Proc. 12th Int. Conf. Artif. Intell. Stat.*, 2009, pp. 488–495.

[10] N. Przulj, D. G. Corneil, and I. Jurisica, "Modeling interactome: Scale-free or geometric?" *Bioinf.*, vol. 20, no. 18, pp. 3508–3515, 2004.

[11] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *J. Mol. Biol.*, vol. 330, no. 4, pp. 771–783, 2003.

[12] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *J. Mach. Learn. Res.*, vol. 11, pp. 1201–1242, 2010.

[13] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," in *Proc. Int. Conf. Data Min.*, 2006, pp. 678–689.

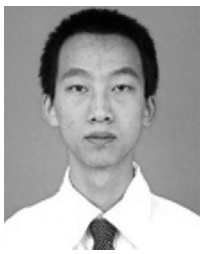
[14] P. Yanardag and S. Vishwanathan, "A submodular framework for graph comparison," in *Proc. NIPS Workshop on Networks*, 2015, pp. 1–7.

[15] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2015, pp. 1365–1374.

[16] K. Chen, et al., "Finding unknown malice in 10 seconds: Mass vetting for new threats at the Google-play scale," in *Proc. 24th USENIX Conf. Secur. Symp.*, 2015, pp. 659–674.

[17] T. Shen, Y. Zhongyang, Z. Xin, B. Mao, and H. Huang, "Detect android malware variants using component based topology graph," in *Proc. IEEE 13th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, 2014, pp. 406–413.
[18] H. Gascon, F. Yamaguchi, D. Arp, and K. Rieck, "Structural detection of android malware using embedded call graphs," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2013, pp. 45–54.
[19] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Math.*, vol. 6, no. 1, pp. 29–123, 2009.
[20] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," in *Proc. 2nd Int. Semantic Web Conf.*, Oct. 2003, pp. 351–368.
[21] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos, "Doulion: Counting triangles in massive graphs with a coin," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2009, pp. 837–846.
[22] A. Pavany, K. Tangwongsan, S. Tirthapuraz, and K.-L. Wu, "Counting and sampling triangles from a graph stream," *Proc. VLDB Endowment*, vol. 6, no. 14, pp. 1870–1881, 2013.
[23] M. Jha, C. Seshadhri, and A. Pinar, "A space efficient streaming algorithm for triangle counting using the birthday paradox," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2013, pp. 589–597.
[24] N. Ahmed, N. Duffield, J. Neville, and R. Kompella, "Graph sample and hold: A framework for big-graph analytics," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2014, pp. 1446–1455.
[25] C. Seshadhri, A. Pinar, and T. G. Kolda, "Wedge sampling for computing clustering coefficients and triangle counts on large graphs," *Stat. Anal. Data Min.*, vol. 7, no. 4, pp. 294–307, 2014.
[26] B. Wu, K. Yi, and Z. Li, "Counting triangles in large graphs by random sampling," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2013–2026, Aug. 2016.
[27] M. Jha, C. Seshadhri, and A. Pinar, "Path sampling: A fast and provable method for estimating 4-vertex subgraph counts," in *Proc. World Wide Web*, 2015, pp. 495–505.
[28] A. Pinar, C. Seshadhr, and V. Visha, "Escape: Efficiently counting all 5-vertex subgraphs," in *Proc. World Wide Web*, 2017, pp. 1431–1440.
[29] F. A. Graybill and R. B. Deal, "Combining unbiased estimators," *Biometrics*, vol. 15, no. 4, pp. 543–550, Dec. 1959.
[30] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proc. IEEE 12th Int. Conf. Data Min.*, Dec. 2012, pp. 745–754.
[31] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proc. 7th ACM SIGCOMM Conf. Int. Meas.*, Oct. 2007, pp. 29–42.
[32] L. Takac and M. Zabovsky, "Data analysis in public social networks." in *Proc. Int. Sci. Conf. Int. Workshop Present Day Trends Innovations*, May 2012, pp. 1–6.
[33] P. Wang, J. Zhao, J. C. S. Lui, D. Towsley, and X. Guan, "Sampling content distributed over graphs," *CoRR*, vol. abs/1311.3882, pp. 1–14, 2013.
[34] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proc. 19th Int. Conf. World Wide Web*, Apr. 2010, pp. 641–650.
[35] "Google programming contest," 2002. [Online]. Available: http://www.google.com/programming-contest/
[36] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, Mar. 2007.
[37] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. A. Hasan, "Guise: Uniform sampling of graphlets for large graph analysis," in *Proc. IEEE 12th Int. Conf. Data Min.*, Dec. 2012, pp. 91–100.
[38] M. Rahman, M. Bhuiyan, and M. A. Hasan, "Graft: An approximate graphlet counting algorithm for large graph analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 2466–2478, Oct. 2014.
[39] P. Wang, J. C. Lui, J. Zhao, B. Ribeiro, D. Towsley, and X. Guan, "Efficiently estimating motif statistics of large networks," *ACM Trans. Knowl. Discovery Data*, 2014, Art. no. 8.
[40] M. A. Hasan and M. J. Zaki, "Output space sampling for graph patterns," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 730–741, Aug. 2009.
[41] S. Chu and J. Cheng, "Triangle listing in massive networks and its applications," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, Aug. 2011, pp. 672–680.
[42] S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," in *Proc. Int. Conf. World Wide Web*, 2011, pp. 607–614.
[43] T. Schank, "Algorithmic aspects of triangle-based network analysis," Ph.D. dissertation, der Fakultatä für Informatik, der Universität Fridericiana zu Karlsruhe, Karlsruhe, Germany, 2007.
[44] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Dimakis, "Beyond triangles: A distributed framework for estimating 3-profiles of large graphs," in *Proc. KDD*, 2015, pp. 229–238.
[45] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Dimakis, "Distributed estimation of graph 4-profiles," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 483–493.
[46] D. Marcus and Y. Shavitt, "Rage–A rapid graphlet enumerator for large networks," *Comput. Netw.*, vol. 56, no. 2, pp. 810–819, 2012.
[47] N. K. Ahmed, J. Neville, R. A. Rossi, and N. G. Duffield, "Efficient counting for large networks," in *Proc. IEEE Int. Conf.Data Min.*, 2015, pp. 1–10.
[48] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon, "Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs," *Bioinf.*, vol. 20, no. 11, pp. 1746–1758, 2004.
[49] S. Wernicke, "Efficient detection of network motifs," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 3, no. 4, pp. 347–359, Oct.-Dec. 2006.
[50] S. Omidi, F. Schreiber, and A. Masoudi-nejad, "Moda: An efficient algorithm for network motif discovery in biological networks," *Genes Genet Syst.*, vol. 84, no. 5, pp. 385–395, 2009.
[51] X. Chen, Y. Li, P. Wang, and J. C. Lui, "A general framework for estimating graphlet statistics via random walk," *Proc. VLDB Endowment*, vol. 10, no. 3, pp. 253–264, 2016.
[52] N. Alon, R. Yuster, and U. Zwick, "Color-coding," *J. ACM*, vol. 42, no. 4, pp. 844–856, Jul. 1995. [Online]. Available: http://doi.acm.org/10.1145/210332.210337
[53] P. Wang, J. C. S. Lui, D. F. Towsley, and J. Zhao, "Minfer: A method of inferring motif statistics from sampled edges," in *Proc. IEEE 32nd Int. Conf. Data Eng.*, 2016, pp. 1050–1061.
[54] X. Chen and J. C. S. Lui, "Mining graphlet counts in online social networks," in *Proc. IEEE 16th Int. Conf. Data Min.*, Dec. 2016, pp. 71–80.
[55] W. Lin, X. Xiao, X. Xie, and X. Li, "Network motif discovery: A GPU approach," in *Proc. IEEE 31st Int. Conf. Data Eng.*, 2015, pp. 831–842.
[56] R. A. Rossi and R. Zhou, "Leveraging multiple gpus and cpus for graphlet counting in large networks," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manag.*, 2016, pp. 1783–1792.

Pinghui Wang received the BS degree in information engineering and the PhD degree in automatic control from Xi'an Jiaotong University, Xi'an, China, in 2006 and 2012, respectively. He was a postdoctoral researcher in the Department of Computer Science and Engineering, The Chinese University of Hong Kong and School of Computer Science, McGill University, Québec, Canada, and was a researcher with Huawei Noah's Ark Lab, Hong Kong. He is currently an associate professor in the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, and is also with the Shenzhen Research Institute at Xi'an Jiaotong University, Shenzhen, China. His research interests include Internet traffic measurement and modeling, traffic classification, abnormal detection, and online social network measurement.

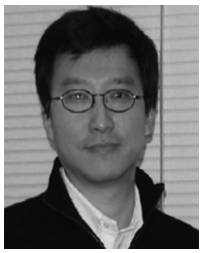


Junzhou Zhao received the BS degree in information engineering and the PhD degree in automatic control from Xi'an Jiaotong University, Xi'an, China, in 2006 and 2012, respectively. He is currently a postdoc fellow in the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong. His research interests include online social network measurement and modeling.

Xiangliang Zhang received the PhD degree in computer science from INRIA-Universite Paris-Sud, France, in 2010. She is currently an assistant professor and directs the Machine Intelligence and Knowledge Engineering (MINE) Laboratory, King Abdullah University of Science and Technology (KAUST), Saudi Arabia. Her main research interests and experiences are in machine learning, data mining, and cloud computing.

Zhenguo Li received the BS and MS degrees from the Department of Mathematics at Peking University, in 2002 and 2005, respectively, and the PhD degree from the Department of Information Engineering at the Chinese University of Hong Kong, in 2008. He is currently the director of AI Theory Lab and principal researcher in Huawei Noah's Ark Lab. He was an associate research scientist in the Department of Electrical Engineering at Columbia University. His research interests include machine learning and artificial intelligence.

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



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, 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 on the editorial boards of the *IEEE/ACM Transactions on Networking*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Parallel and Distributed Systems*, the *Journal of Performance Evaluation*, and the *International Journal of Network Security*. He was the chairman of the CSE Department from 2005-2011. His personal interests include films and general reading. He received various departmental teaching awards and the CUHK Vice-Chancellor's Exemplary Teaching Award. He is also a corecipient 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.

Don Towsley received the BA degree in physics and the PhD degree in computer science from the University of Texas, in 1971 and 1975, respectively. From 1976 to 1985, he was a member of the faculty in the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. He is currently a distinguished professor in the Department of Computer Science, University of Massachusetts. He has held visiting positions at the IBM T.J. Watson Research Center, Yorktown Heights, New York; Laboratoire MASI, Paris, France; INRIA, Sophia-Antipolis, France; AT&T Labs-Research, Florham Park, New Jersey; and Microsoft Research Lab, Cambridge, United Kingdom. His research interests include network measurement and modeling, networks, and performance evaluation. He served as editor-in-chief of the *IEEE/ACM Transactions on Networking* and on the editorial boards of the *Journal of the ACM*, the *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 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. He is a fellow of the IEEE and ACM, member of the ACM and ORSA, and chair of the IFIP Working Group 7.3.

Jing Tao received the BS and MS degrees in automatic control from Xi'an Jiaotong University, Xi'an, China, in 2001 and 2006, respectively. He is currently a teacher with Xi'an Jiaotong University and is on-the-job working toward the PhD degree in 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.

Xiaohong Guan received the BS and MS degrees in automatic control from Tsinghua University, Beijing, China, in 1982 and 1985, respectively, and the PhD degree in electrical engineering from the University of Connecticut, Storrs, in 1993. He is currently a professor in the Systems Engineering Institute, Xi'an Jiaotong University, Xi'an, China. From 1993 to 1995, he was a consulting engineer in 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, Massachusetts. 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 the 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 the *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 is a fellow of the IEEE.