2016 IEEE 16th International Conference on Data Mining

Mining Graphlet Counts in Online Social Networks

Xiaowei Chen, John C.S. Lui
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Email: {xwchen, cslui}@cse.cuhk.edu.hk

***Abstract*—Counting subgraphs is a fundamental analysis task for online social networks (OSNs). Given the sheer size and restricted access of online social network data, efficient computation of subgraph counts is highly challenging. Although a number of algorithms have been proposed to estimate the relative counts of subgraphs in OSNs with restricted access, there are only few works which try to solve a more general problem, i.e., counting subgraph frequencies. In this paper, we propose an efficient random walk-based framework to estimate the subgraph counts. Our framework generates samples by leveraging consecutive steps of the random walk as well as by observing neighbors of visited nodes. Using the importance sampling technique, we derive unbiased estimators of the subgraph counts. To make better use of the degree information of visited nodes, we also design an improved estimator, which increases the efficiency of the estimate at no additional cost. We conduct extensive experimental evaluation on real-world OSNs to confirm our theoretical claims. The experiment results show that our estimators are unbiased, accurate, efficient and better than the state-of-the-art algorithm. For the Weibo graph with more than 58 million nodes, our method produces estimate of triangle count with an error less than 5% using only 20 thousands sampled nodes. Detailed comparison with the state-of-the-art method demonstrates that our algorithm is 4 to 5 times more accurate.**

***Keywords*—Graphlet counting; Random walk; Online social networks.**

I. Introduction

Analyzing properties of online social networks (OSNs) has attracted extensive attention because of their increasing popularity, significant importance and diverse applications [1]. In this work, we focus on counting the number of subgraphs in OSNs. Subgraphs whose counts are desired are also referred as *"graphlets"*, *"motifs"* or *"pattern subgraphs"* [2]. Counting the number of graphlets in OSNs is a fundamental analysis task. For example, computing the triadic tendencies (e.g., clustering coefficient) has a long history in the social network analysis and modeling [3]–[7]. Recently, Ugander et al. [8] analyzed the 4-node graphlet counts for social networks and studied what properties are merely indicated by graph theory and what are actual social features of the real-world graphs. There are also numerous applications of graphlet counts in social science, e.g., large-scale graph comparison [9], anomaly and event detection [4], [5], nodes classification [8].

However, it is a challenging task to compute graphlet counts for OSNs. Firstly, the complete networks are usually too large, which renders the exact computation impractical. In fact, counting graphlets even on a moderately sized OSN has prohibitive computation cost, e.g., the computation of 4-node graphlets cannot finish within a week for a Twitter graph with 21.3M nodes in our datasets using the state-of-the-art exact counting algorithm in [9]. Secondly, for most OSNs like Facebook and Twitter, the underlying network topology is unknown beforehand [1], [10], and researchers can only have limited access via APIs provided by the OSNs' operators. Such restricted access makes the retrieval of entire topology prohibitively expensive due to extremely high query cost. To address the challenges, graph sampling via crawling has been widely applied for OSN measurement [10]–[14]. In particular, random walk-based methods are popular due to its simple implementation and capability to remove bias of samples.

Our goal is to design an efficient random walk-based sampling algorithm to estimate the graphlet counts in the OSNs. Different from some nodal properties, e.g., degree distribution, which has been extensively studied with random walk-based methods [10], [14], [15], single node samples generated by the random walk are not sufficient for the estimation of graphlet counts since single node carries no information about local structure. To estimate graphlet counts, we need to examine the local structure of networks during the random walk.

Summary of Contributions. In this work, we design an efficient random walk-based algorithm to estimate $3, 4, 5$-node graphlet counts. It is important to point out that our algorithm can be easily extended to graphlets with larger size. We summarize the contributions as follows.

- **Novel Algorithm.** Our algorithm provides provably unbiased estimate. The main idea is to consider the *consecutive steps* of the random walk and examine the neighbors of visited nodes. To further improve the efficiency, we also propose an improved estimator which makes better use of degree information of visited nodes. To the best of our knowledge, our algorithm is the first to estimate all $3, 4, 5$-node graphlet counts of OSNs via the random walk.
- **Analytical Bound.** We provide an analytical bound on the sample size to guarantee that the estimate is within $(1 \pm \epsilon)$ relative to the true counts with probability of at least $1 - \delta$. The bound depends on the parameter ϵ and the confidence level δ as well as some parameters of the graphs. The analytical bound guarantees the theoretical convergence of our estimator and sheds light on what parameters of the graphs affect the performance of our algorithm.
- **Extensive Experiments.** We validate our algorithm on six OSNs. The experiments show that our estimators are unbiased and accurate. Furthermore, our estimators converge to the ground truth rapidly. Compared with the state-of-the-

2374-8486/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDM.2016.99

art random walk-based method [16] which is only capable of computing the relative counts of graphlets, our algorithm not only solves the more general problem, i.e., graphlet counting, but also significantly outperforms the state-of-the-art method in estimating relative counts of graphlets.

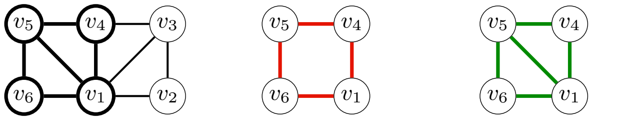
- **Excellent Empirical Accuracy.** The experiment results demonstrate that our algorithm is practical, e.g., with only 20K visited nodes, the average relative error of estimated triangle counts is within 5% for all the tested graphs.

II. Preliminaries

A. Notations and Definitions

Our input network is modeled as an undirected, unweighted and connected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. We assume G has neither self-loops nor multi-edges. For a node $v \in V$, $\mathcal{N}(v)$ represents the set of neighbors of v and $d_v = |\mathcal{N}(v)|$ is the degree of v.

Subgraph. A k-node subgraph $G_k = (V_k, E_k)$ of G satisfies $V_k \subseteq V$, $E_k \subseteq E$ and $|V_k| = k$. An "*induced subgraph*" ensures that all edges connecting nodes in V_k are also present in E_k, i.e., $E_k = \{(u,v)|u,v \in V_k \wedge (u,v) \in E\}$. We distinguish between subgraph and induced subgraph. In general, if we do not say "induced" subgraph, we mean a "normal" subgraph which just has a subset of edges of the original graph. Consider examples in Fig. 1. The edge set $\{(v_1,v_6),(v_5,v_6),(v_1,v_4),(v_4,v_5)\}$ forms a (non-induced) 4-node subgraph while the node set $\{v_1,v_4,v_5,v_6\}$ induces a 4-node induced subgraph.



original graph

(non-induced) 4-node subgraph

4-node induced subgraph

Fig. 1: An example of subgraph and induced subgraph.

Isomorphic. Two graph $G = (V, E)$ and $G' = (V', E')$ are isomorphic if there exists a bijection $\varphi : V \rightarrow V'$ with $(u,v) \in E \Leftrightarrow (\varphi(u), \varphi(v)) \in E'$ for all $u, v \in V$ [17].

Graphlet. Graphlets are defined as *non-isomorphic, connected, induced* subgraphs in large graphs. Let $\mathcal{G}^k$ denote a family of k-node graphlets, i.e., $\mathcal{G}^k = \{g_1^k, \cdots, g_m^k\}$. To illustrate, in Section III, Table I depicts $\mathcal{G}^3$ and $\mathcal{G}^4$ while Table II depicts $\mathcal{G}^5$. The second rows of these two tables show all possible 3, 4, 5-node graphlets. We can see that $|\mathcal{G}^3| = 2$, $|\mathcal{G}^4| = 6$, and $|\mathcal{G}^5| = 21$.

Problem Definition. Given a family of k-node graphlets $\mathcal{G}^k = \{g_1^k, \cdots, g_m^k\}$, let C_i^k denote the number of *induced* subgraphs that are isomorphic to the graphlet $g_i^k \in \mathcal{G}^k$ in the input graph G. Our goal is to compute $\{C_1^k, \cdots, C_m^k\}$ efficiently.

We refer to $\{C_1^k, \cdots, C_m^k\}$ as the *graphlet counts*. The computation of graphlet counts is usually restricted to graphlets of no more than 5 nodes [2], [9], [11], [18]–[20] due to the extremely high computation cost. In this work, our aim is to efficiently compute C_i^k, for $k = 3, 4, 5$.

B. Random Walk on Graphs

Access Model. In this work, we assume the topology of the input graph G is not readily available and we can only obtain it with restricted access, i.e., the graph data can only be accessed by calling APIs provided by operators of OSNs. While APIs have various design specifications across different OSNs, most of them support queries by taking node IDs as input. Some basic information collected when querying a node u is the set of friends $\mathcal{N}(u)$, and other attributes of u.

Random Walk. Random walk-based methods fit in with the restricted access naturally. Simple random walk (SRW) on a graph is defined as follows. We start from an initial node v_0 in the graph and extract its information, and then randomly select one of v_0's neighbors (with equal probability), say v_1, and then we transit to and explore v_1. We repeat this process until some stopping criteria, e.g., stop after making a pre-defined number of transitions. In fact, SRW on G can be modeled as a *finite, time reversible* Markov chain with state space V and transition matrix $\mathbf{P}$ where

$$\mathrm{P}(u,v) = \begin{cases} \frac{1}{d_u} & \text{if } (u,v) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\pi(v)$ be the steady state probability of node v. It is easy to show that $\pi(v) = d_v/(2|E|), v \in V$ [21]. Note that these steady state probabilities (a.k.a. stationary distribution) are important to remove the random walk sampling bias.

Theoretical Guarantee. In the following, we review the Strong Law of Large Numbers (SLLN) for the Markov chain, which serves as the basis for graph sampling via random walk over a graph G, or more generally, the Markov Chain Monte Carlo (MCMC) samplers.

Suppose the Markov chain with the state space $\mathcal{M}$ has the stationary distribution $\boldsymbol{\pi}$, and the function $f : \mathcal{M} \rightarrow \mathbb{R}$ is an integrable function with respect to $\boldsymbol{\pi}$. Then the expectation of f w.r.t. $\boldsymbol{\pi}$ which is given by $\mu \triangleq \mathbb{E}_{\boldsymbol{\pi}}[f] \triangleq \sum_{X \in \mathcal{M}} \pi(X) f(X)$ exists. Let $\{X_t\}_{t=1}^n$ represent the sequence of visited states of the Markov chain. We define the sample average $\hat{\mu}_n \triangleq \frac{1}{n}\sum_{t=1}^n f(X_t)$ as the *estimator* for the expectation μ.

Theorem 1. *Suppose $\{X_n\}$ is a finite, irreducible Markov chain with stationary distribution $\boldsymbol{\pi}$. As $n \rightarrow \infty$, we have*

$$\hat{\mu}_n \rightarrow \mu \textit{ almost surely (a.s.)}$$

for any initial distribution and any function with $\mathbb{E}_{\boldsymbol{\pi}}[|f|] < \infty$.

The above theorem guarantees the convergence of the sample mean to the expectation. The estimator $\hat{\mu}_n$ is an unbiased estimator of μ according to the SLLN. Later on, we use the SLLN to prove the unbiasedness of our proposed estimator.

III. Algorithmic Framework

Our algorithm generates the subgraph samples through *consecutive steps of the random walk*. Furthermore, we leverage the neighbors of the nodes collected along the random walk. We eliminate the sampling bias via *importance sampling* [22].

A. Basic Idea

We first describe the high level idea. For clarity, we introduce the concepts of *touched subgraph* and *visible subgraph*. A k-node subgraph $G_k = (V_k, E_k)$ is defined as a *touched subgraph* if neighbor sets of all nodes in V_k are *available*, i.e., we obtain the $\mathcal{N}(v)$ for all $v \in V_k$ by quering node v through APIs. The subgraph $G_k = (V_k, E_k)$ is defined as a *visible subgraph* if there is *one and only one* node $v \in V_k$ whose $\mathcal{N}(v)$ is not available. Such subgraph is *"visible"* because we can infer all edges between nodes in V_k so as to determine the graphlet type of the visible subgraph. To illustrate, consider the graph in Fig. 2. Assume we already obtain the neighbors of nodes 4 and 7, then the subgraph induced by $\{4, 7\}$ is a touched subgraph while the subgraph induced by $\{4, 7, 8\}$ is a visible subgraph (a triangle). The subgraph induced by $\{5, 4, 8\}$ is not visible because we cannot determine whether 5 and 8 are connected with only $\mathcal{N}(4)$ and $\mathcal{N}(7)$.

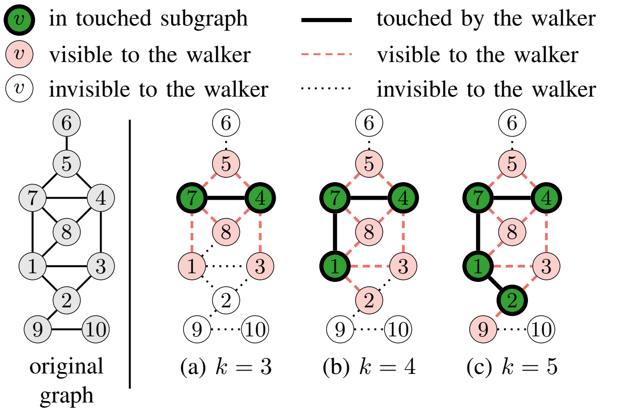
Our idea is to generate $(k-1)$-node *touched* subgraphs first. Then using these $(k-1)$-node touched subgraphs together with the neighborhood nodes, we can obtain many k-node *visible* subgraph samples. Since our goal is graphlet counts, only connected subgraphs are considered here. We generate touched subgraphs through *consecutive* steps of the random walk. Formally, we consider each $k-1$ consecutive steps of the random walk which visits $k-1$ *distinct* nodes as a $(k-1)$-node subgraph. These $(k-1)$-node subgraphs are touched subgraphs according to the access model in Subsection II-B. If the random walk fails to visit $k-1$ distinct nodes with $k-1$ steps, we just continue the random walk.

Suppose we have obtained a $(k-1)$-node touched subgraph $G_{k-1} = (V_{k-1}, E_{k-1})$. Define the neighborhood of V_{k-1} as $\mathcal{N}(V_{k-1}) = \left\{\cup_{v \in V_{k-1}} \mathcal{N}(v)\right\} \setminus V_{k-1}$. The key observation is that the k-node subgraphs induced by $V_{k-1} \cup \{v\}, \forall v \in \mathcal{N}(V_{k-1})$ are visible to us. We use these k-node visible subgraphs as the obtained k-node subgraph samples. Note that we do not need to explore any node in $\mathcal{N}(V_k)$ for extra neighborhood information to determine the graphlet types of these k-node visible subgraph samples. It is sufficient to get $(k-1)$-node touched subgraphs first so to get the k-node subgraph samples. Specifically, we get $2, 3, 4$-node touched subgraph first for $3, 4, 5$-node graphlet counts estimation. Refer to Fig. 2 for the illustration of the basic idea.

Each k-node induced subgraph is visible to the walker with unequal probability. We need to compute the "*visible probability*" of the subgraphs, and then use the importance sampling technique [22] to remove the bias. We explain the detailed derivation of unbiased estimator in next subsections.

B. Mathematical Description

Now, we translate the basic idea to formal mathematical description and define the MCMC sampler. Our proposed algorithm considers the $l = k-1$ consecutive steps of the random walk as a touched subgraph. Accordingly, we define a Markov chain that remembers l steps of the random walk as the *expanded Markov chain*. The state space $\mathcal{M}^{(l)}$ of the expanded Markov chain is defined as the set of all possible consecutive l steps of the random walk, where l represents how many consecutive steps we take into consideration. The state $X \in \mathcal{M}^{(l)}$ can be written as $X = (v_1, \cdots, v_l)$ where $(v_i, v_{i+1}) \in E, 1 \le i \le l-1$. Each time the random walk proceeds to next node, the expanded Markov chain transits to the next state. For example, suppose the expanded Markov chain is at state $X_t = (v_1, \cdots, v_l)$, for the random walker, it is at node v_l. If the walker randomly chooses a neighbor v_{l+1} of v_l and moves to it, then the expanded Markov chain transits to the state $X_{t+1} = (v_2, \cdots, v_{l+1})$. For any two states $X_i = (v_{i_1}, \cdots, v_{i_l})$ and $X_j = (v_{j_1}, \cdots, v_{j_l})$ in $\mathcal{M}^{(l)}$, the transition matrix $\mathbf{P}_M$ of the expanded Markov chain is



original graph (a) $k = 3$ (b) $k = 4$ (c) $k = 5$

Fig. 2: Illustration of the basic idea. (a) When $k = 3$, assume we visit the touched subgraph ④–⑦ induced by $\{4, 7\}$, then we can observe 2 visible triangles (△) and 2 visible wedges (∧). (b) When $k = 4$, suppose we walk for three steps and visit nodes $\{4, 7, 1\}$ sequentially, then the wedge ④–⑦–① is a 3-node touched subgraph; we can observe 1 line (⊓), 1 cycle (□), 1 chordal-cycle (⊠), and 1 tailed-triangle (⊠). (c) When $k = 5$, we need to walk for four steps to get the touched subgraphs. Assume we visit nodes $\{4, 7, 1, 2\}$ sequentially, then there are 1 ⊠, 1 ⊠, 1 ⊠, and 1 ⊠ visible to the walker; here the 4-node line ④–⑦–①–② is the 4-node touched subgraph.

$$\mathrm{P}_M(X_i, X_j) = \begin{cases} \frac{1}{d_{v_{i_l}}} & \text{if } (v_{i_2}, \cdots, v_{i_l}) = (v_{j_1}, \cdots, v_{j_{l-1}}), \\ 0 & \text{otherwise.} \end{cases}$$

Note that we define the expanded Markov chain only for the convenience of deriving the unbiased estimator, since it describes the same process as the random walk. It is easy to verify that the expanded Markov chain is irreducible and there exists a unique stationary distribution [23]. Let $\boldsymbol{\pi}_M$ denote the stationary distribution of the expanded Markov chain. For the state $X = (v_1, \cdots, v_l) \in \mathcal{M}^{(l)}$, we have

$$\pi_M(X) = \begin{cases} d_{v_1}/2|E| & l = 1, \\ 1/2|E| & l = 2, \\ \frac{1}{2|E|}\frac{1}{d_{v_2}} \cdots \frac{1}{d_{v_{l-1}}} & l \ge 3. \end{cases}$$

We now define the function $f_i^k : \mathcal{M}^{(l)} \to \mathbb{R}$. Let $V(X) \triangleq \{v_1, \cdots, v_l\}$ denote the set of nodes in $X = (v_1, \cdots, v_l)$, where l equals to $k-1$ in our algorithm. If $|V(X)| < l$, then $f_i^k(X) = 0$. Otherwise, $f_i^k(X)$ equals to the number of subgraphs induced by $V(X) \cup \{v\}$ $(\forall v \in \mathcal{N}(V(X)))$ that are isomorphic to g_i^k. Let $\mathcal{S} \triangleq \{G_k(V_k) | V_k = V(X) \cup \{v\}, v \in \mathcal{N}(V(X))\}$. Here $G_k(V_k)$ denotes the subgraph induced by V_k. Formally, the function $f_i^k(X)$ can be written as:

$$f_i^k(X) = \begin{cases} 0 & \text{if } |V(X)| < l, \\ |\{G_k | G_k \in \mathcal{S} \wedge G_k \text{ isomorphic to } g_i^k\}| & \text{if } |V(X)| = l. \end{cases}$$

Example. Refer to Fig. 2. When $k = 3$, the function $f_1^3((4,7)) = 2, f_2^3((4,7)) = 2$ since we observe 2 wedges (g_1^3) and 2 triangles (g_2^3) containing subgraph induced by $\{4, 7\}$.

The function $f_i^k(X)$ indicates how many k-node visible subgraphs we can observe through the $(k-1)$-node touched subgraph. In next subsection, we derive an unbiased estimator of the graphlet counts using the stationary distribution of the expanded Markov chain and the real-valued function f_i^k.

C. Derivation of the Unbiased Estimator

To derive the unbiased estimator of C_i^k, we need to remove the bias of the k-node visible subgraphs. In particular, our goal is to design an appropriate re-weight function $w_i^k(X)$ such that

$$\frac{1}{n} \sum_{t=1}^{n} w_i^k(X_t) f_i^k(X_t) \to C_i^k \ a.s.$$

We first compute the number of states that find the subgraph G_k. If state $X \in \mathcal{M}^{(k-1)}$ is a $(k-1)$-node subgraph of G_k, we say that G_k is found by X. *One important note is that a subgraph G_k may be found by several states.* Recall that $V(X)$ denotes the set of nodes in the state X. Define the set of states which can find subgraph $G_k = (V_k, E_k)$ as

$$\mathcal{B}(G_k) \triangleq \left\{ X | X \in \mathcal{M}^{(k-1)}, |V(X)| = k-1, V(X) \subset V_k \right\}.$$

Note that the size of $\mathcal{B}(G_k)$ only depends on the graphlet type of G_k. Hence, we define $\beta_i^k = |\mathcal{B}(G_k)|$ for any subgraph G_k isomorphic to the graphlet g_i^k. *Since each subgraph G_k isomorphic to g_i^k is found by β_i^k states*, we have

$$\sum_{X \in \mathcal{M}^{(k-1)}} f_i^k(X) = \beta_i^k C_i^k. \tag{1}$$

Finally, the re-weight function is

$$w_i^k(X) \triangleq \frac{1}{\beta_i^k} \cdot \frac{1}{\pi_M(X)}, \ \beta_i^k \neq 0. \tag{2}$$

The reciprocal of the re-weight function $w_i^k(X)$ is the nominal "visible probability". The re-weight function consists of two parts. The first part $1/\beta_i^k$ is due to that each k-node subgraph isomorphic to g_i^k is found β_i^k times. The second part is due to the non-uniform sampling of states in $\mathcal{M}^{(k-1)}$. The condition $\beta_i^k \neq 0$ is satisfied for most graphlets, e.g., when $k = 3, 4, 5$, the only graphlet with $\beta_i^k = 0$ is g_3^5 ( ). In fact, our algorithm can be applied to any k-node graphlets with $\beta_i^k \neq 0$. For graphlets with $\beta_i^k = 0$ (i.e., $k = 5, i = 3$), we will discuss the detailed estimation method in next subsection. Combining the importance sampling [22] and SLLN, we have the following theorem.

TABLE I: Coefficient α_i^k and β_i^k for $3, 4$-node graphlets.

Graphlets	g_1^3	g_2^3	g_1^4	g_2^4	g_3^4	g_4^4	g_5^4	g_6^4
Shape								
$\alpha_i^k(k=3,4)$	2	6	2	**0**	8	4	12	24
$\beta_i^k(k=3,4)$	4	6	4	6	8	10	16	24

Theorem 2. *The average of the function $w_i^k(X) f_i^k(X)$ is*

$$\hat{C}_i^k \triangleq \frac{1}{n} \sum_{t=1}^{n} w_i^k(X_t) f_i^k(X_t), \tag{3}$$

which is an asymptotic unbiased estimator of C_i^k for the graphlet g_i^k with $\beta_i^k \neq 0$.

We have the detailed proof in the technical report [24]. Algorithm 1 demonstrates the sampling procedure.

Algorithm 1 Unbiased Estimate of k-node Graphlet Counts

```
Input: sample budget $n$, graphlet size $k$, input graph $G$
Output: unbiased estimate of $C_i^k$ for graphlet $g_i^k$ with $\beta_i^k \neq 0$
$\hat{C}_i^k \leftarrow 0, \ \forall 1 \leq i \leq |\mathcal{G}^k|$
$X = (v_1, \cdots, v_{k-1}) \leftarrow$ initial $k-1$ steps
random walk step $t \leftarrow 0$
while $t < n$ do
    for $i \in \{1, \cdots, |\mathcal{G}^k|\}$ do
        if $\beta_i^k \neq 0$ then
            $\hat{C}_i^k \leftarrow \hat{C}_i^k + w_i^k(X) f_i^k(X)/n$
    $v_{t+k} \leftarrow$ uniformly choose a neighbor of $v_{t+k-1}$
    $X \leftarrow (v_{t+2}, \cdots, v_{t+k})$
    $t \leftarrow t + 1$
return $\left[\hat{C}_1^k, \cdots, \hat{C}_{|\mathcal{G}^k|}^k\right]$
```

Computation of β_i^k. The remaining task is to compute β_i^k, which is part of the re-weight function in Eq. (2). Define $\mathcal{A}(G_{k-1})$ as the set of states whose node set is the same as the connected subgraph $G_{k-1} = (V_{k-1}, E_{k-1})$, i.e.,

$$\mathcal{A}(G_{k-1}) \triangleq \left\{ X | X \in \mathcal{M}^{(k-1)}, V(X) = V_{k-1} \right\}.$$

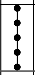
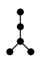
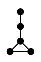
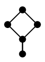
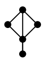
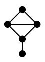
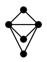
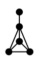
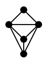
The size of $\mathcal{A}(G_{k-1})$ only depends on the graphlet type of G_{k-1}. We define $\alpha_j^{k-1} = |\mathcal{A}(G_{k-1})|$ for any G_{k-1} isomorphic to g_j^{k-1}. Let t_j denote the count of g_j^{k-1} in G_k. Here G_k is isomorphic to g_i^k. It is easy to verify that

$$\beta_i^k = \sum_{j=1}^{|\mathcal{G}^{k-1}|} t_j \cdot \alpha_j^{k-1}. \tag{4}$$

Example. For a triangle G_3 induced by the set $\{u, v, w\}$, we have $\mathcal{B}(G_3) = \{(u,v), (v,u), (v,w), (w,v), (u,w), (w,u)\}$. Hence $\beta_2^3 = |\mathcal{B}(G_3)| = 6$. The set $\mathcal{A}(G_3) = \{(u,v,w), (w,v,u), (v,u,w), (w,u,v), (u,w,v), (v,w,u)\}$. So we have $\alpha_2^3 = |\mathcal{A}(G_3)| = 6$. There are 4 triangles in the 4-node clique (g_6^4,). Based on Eq. (4), we have $\beta_6^4 = 4 \times 6 = 24$.

Intuitively, $|\mathcal{A}(G_{k-1})|$ equals to the number of ways to walk through V_k during the random walk. Theoretically, it is twice of the number of *Hamilton paths* in G_{k-1} (each Hamilton path

TABLE II: Coefficient α_i^5 and β_i^5 for 5-node graphlets.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Shape																					
α_i^5	2	**0**	**0**	2	4	**0**	10	4	4	8	8	12	14	12	12	20	28	36	48	72	120
β_i^5	4	4	0	10	8	8	10	12	20	16	20	24	20	36	36	34	36	56	56	84	120

is counted for both directions). Counting Hamilton path is an NP-complete problem. We need to enumerate all Hamilton paths in the graphlets to compute α_j^{k-1}. Fortunately, the computation of α_j^{k-1} is not a big concern since the computation of graphlet counts is usually restricted to $k \leq 5$ in various applications [4], [9], [25]. The detailed computation of α_i^k and β_i^k can be found in the technical report [24]. Table I and II list the values of α_i^k and β_i^k for all the $3, 4, 5$-node graphlets.

Practical Issues. Under the restricted access, the exact number of nodes and edges is usually unknown. Very often, one can obtain the approximated number of users in OSNs [26] (e.g., from financial reports or the Internet). However, the number of edges is not available in most cases. Since our primary goal is the graphlet counts, it is essential to know the number of edges. To address this problem, we use the following fact

$$\mathbb{E}_{\boldsymbol{\pi}}\left[\frac{1}{d_v}\right] = \sum_{v\in V}\frac{d_v}{2|E|}\frac{1}{d_v} = \frac{|V|}{2|E|} \qquad \textbf{(5)}$$

where $\boldsymbol{\pi}$ is the stationary distribution of the simple random walk. Assume the sequence of visited node is $v_1, \cdots, v_{n+k-2}$ and the corresponding sampled states are $X_1, \cdots, X_n$. Here $X_i \triangleq (v_i, \cdots, v_{i+k-2})$. According to Eq. (5), the number of edges can be estimated with $|V|(n+k-2)/(2\sum_{t=1}^{n+k-2} 1/d_{v_t})$. Define $\tilde{\pi}_M(X) = 2|E| \cdot \pi_M(X)$ and $\tilde{w}_i^k(X) = 1/(\beta_i^k \cdot \tilde{\pi}_M(X))$. The graphlet counts can be estimated with

$$\hat{C}_i^k \triangleq |V|\left(\frac{n+k-2}{n}\right)\left(\frac{\sum_{t=1}^{n}\tilde{w}_i^k(X_t)f_i^k(X_t)}{\sum_{t=1}^{n+k-2}1/d_{v_t}}\right). \qquad \textbf{(6)}$$

The unbiasedness of Eq. (6) can be proved by combining Theorem 2 and Eq. (5). Note that both $\tilde{w}_i^k(X)$ and $f_i^k(X)$ can be computed with the local neighborhood information and no knowledge of $2|E|$ is required.

Another practical issue is that OSNs are not necessary connected. The random walk can only crawl over nodes in the same connected components. However, it is not a big concern for OSNs since most nodes ($> 90\%$) of OSNs are in the largest connected components (LCCs) [1]. The LCCs are enough to represent the properties of the whole graphs. Besides, we can use the state-of-the-art algorithm [11] with high accuracy to estimate number of nodes in the LCCs.

D. Estimator for 5-node Graphlets

All $3, 4, 5$-node graphlet counts can be estimated with Eq. (3) except the 5-node star graphlets (g_3^5, ✕) due to $\beta_3^5 = 0$ ($\beta_3^5 = 0$ is because all 4-node graphlets in ✕ are ⸬ and we cannot walk through ⸬ via simple random walk). However, we can use the relationship between induced subgraphs and non-induced subgraphs to solve this problem. Let N_i^k denote the sum of induced and non-induced subgraphs that are isomorphic to graphlet g_i^k. N_3^5 denotes the counts of subgraphs isomorphic to g_3^5 (✕). There is a simple linear relationship between C_i^k and N_i^k. We have the following:

$$\begin{aligned} N_3^5 &= \sum_{v\in V}\binom{d_v}{4} = \sum_{i=1}^{21}\phi_i^5\cdot C_i^5 = C_3^5 + C_6^5 + C_9^5 + C_{10}^5 \\ &+2C_{14}^5 + C_{15}^5 + C_{16}^5 + 2C_{18}^5 + C_{19}^5 + 3C_{20}^5 + 5C_{21}^5, \end{aligned} \qquad \textbf{(7)}$$

where ϕ_i^5 denotes the number of g_3^5 contained in the graphlet g_i^5. Given the sequence of nodes $v_1, \cdots, v_{n+3}$ visited by the random walk, the unbiased estimator of N_3^5 is

$$\hat{N}_3^5 \triangleq \frac{1}{n+3}\sum_{t=1}^{n+3} 2|E|\binom{d_{v_t}}{4}/d_{v_t}.$$

Graphlet counts except C_3^5 can be estimated with Eq. (3), i.e.,

$$\hat{C}_i^5 = \frac{1}{n}\sum_{t=1}^{n} 2|E|\frac{d_{v_{t+1}}d_{v_{t+2}}}{\beta_i^5}f_i^5\left((v_t, v_{t+1}, v_{t+2}, v_{t+3})\right),\ i \neq 3.$$

We leverage the formula (7) to estimate C_3^5, i.e.,

$$\hat{C}_3^5 = \hat{N}_3^5 - \sum_{i\in\{1,\cdots,21\}\backslash\{3\}}\phi_i^5\hat{C}_i^5.$$

Applying the linearity of the expectation, one can easily prove that $\mathbb{E}_{\boldsymbol{\pi}}[\hat{C}_3^5] = \mathbb{E}_{\boldsymbol{\pi}}[\hat{N}_3^5] - \sum_{i\in\{1,\cdots,21\}\backslash\{3\}}\phi_i^5\mathbb{E}[\hat{C}_i^5]$, which is $N_3^5 - \sum_{i\in\{1,\cdots,21\}\backslash\{3\}}\phi_i^5 C_i^5 = C_3^5$.

IV. Analytical Bound

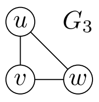
We also provide an analytical bound on the needed sample size to guarantee our estimators are within $(1 \pm \epsilon)$ accuracy with a high probability at least $(1-\delta)$. The bound is expressed in terms of the accuracy parameter ϵ and the confidence level δ, as well as some parameters of the graph. Since our analytical bound depends on the mixing time of the random walk, we first introduce the mixing time, which quantifies how fast the random walk approaches the stationary distribution.

Definition 1 (Mixing time)**.** *[27, Definition 1] The mixing time (parameterized by ξ) of a Markov chain is defined as*

$$\tau(\xi) \triangleq \max_i \min\{t : |\boldsymbol{\pi} - \boldsymbol{\pi}_0^{(i)}\mathbf{P}^t|_1 < \xi\},$$

where $\boldsymbol{\pi}$ is the stationary distribution, $\boldsymbol{\pi}_0^{(i)}$ is the initial distribution concentrated at node v_i, $\mathbf{P}^t$ is the transition matrix after t steps, and $|\cdot|_1$ is the total variation distance.

We start by analyzing the estimator in Eq. (3) where $|E|$ is known. Define $M_i^k = \max_{X\in\mathcal{M}^{(l)}} w_i^k(X)f_i^k(X)$. Let T be the mixing time that ensures the total variation distance between



$$X_a = (u,v,w), \pi_M(X_a) = \frac{1}{2|E|}\frac{1}{d_v}$$
$$X_b = (u,w,v), \pi_M(X_b) = \frac{1}{2|E|}\frac{1}{d_w}$$
$$X_c = (v,u,w), \pi_M(X_c) = \frac{1}{2|E|}\frac{1}{d_u}$$
$$P(G_3)/|\mathcal{A}(G_3)| = \frac{1}{6|E|}\left(1/d_u + 1/d_v + 1/d_w\right)$$

Fig. 3: Example of the observation. $X_a, X_b, X_c \in \mathcal{A}(G_3)$. **If the degree** d_u, d_v, d_w **are unequal to each other, then the steady state probabilities** $\pi_M(X_a), \pi_M(X_b), \pi_M(X_c)$ **are different from each other.**

the distribution after T steps and the stationary distribution of the random walk is within $1/8$, i.e., $T = \tau(1/8)$. The initial distribution of the random walk is denoted by $\boldsymbol{\varphi}$ and $\|\boldsymbol{\varphi}\|_{\boldsymbol{\pi}} \triangleq \sum_{v\in V} \varphi^2(v)/\pi(v)$.

Lemma 1. *There is a constant value ζ, such that if $n \geq B_1 \triangleq \zeta \frac{M_i^k}{C_i^k} \frac{\log(\|\boldsymbol{\varphi}\|_{\boldsymbol{\pi}}/\delta)}{\epsilon^2} T$, we have*

$$\Pr\left[|\hat{C}_i^k - C_i^k| \leq \epsilon C_i^k\right] \geq 1 - \delta.$$

Moreover, the estimator in Eq. (6) assumes the number of edges is unknown, which is a common case for OSN analysis. To guarantee $\hat{C}_i^k$ is within $(1 \pm \epsilon) C_i^k$ with probability of at least $1-\delta$, it requires that $n \geq B_2 \triangleq \zeta \frac{M_i^k}{C_i^k} \frac{\log(2\|\boldsymbol{\varphi}\|_{\boldsymbol{\pi}}/\delta)}{\epsilon^2}(9T)$. The detailed proof of B_1 and B_2 is in our technical report [24].

Remarks. In addition to ϵ and δ, the sample size bound B_1 and B_2 also depend on the parameters of the graphs. One parameter is the mixing time of the random walk. The smaller the mixing time, the smaller the required sample size. For social network with small world properties, the mixing time is $\Theta(\log^2 n)$ [11], [27], [28]. Another parameter is M_i^k/C_i^k, which describes the ratio between the local maximum graphlet counts and the average graphlet counts. For example, let $\Delta \triangleq \max_{(u,v)\in E} |\mathcal{N}(u) \cap \mathcal{N}(v)|$ denote the maximum number of triangles sharing the same edge. Then we have $M_2^3/C_2^3 = (2 \cdot \max_{X\in\mathcal{M}^{(2)}} f_2^3(X))/(C_i^k/|E|) = 2\Delta/(C_2^3/|E|)$.

V. Improved Estimator

We now design an improved estimator. The core idea is to view each state $X \in \mathcal{M}^{(k-1)}$ containing $k-1$ distinct nodes as a $(k-1)$-node subgraph G_{k-1} and compute the *sampling probability* of the subgraph G_{k-1} instead of using the stationary distribution of the state X. The new estimator benefits from the better use of the degree information of visited nodes. The idea is inspired by the following observation.

Observation 1. *For $X_a, X_b \in \mathcal{A}(G_{k-1})$, it is possible to have $\pi_M(X_a) \neq \pi_M(X_b)$.*

Fig. 3 gives an example of the observation. Recall that $\mathcal{A}(G_{k-1})$ is the set of states in $\mathcal{M}^{(k-1)}$ whose node set is the same as subgraph G_{k-1}. Once the state X is visited, we can enumerate all the states in $\mathcal{A}(G_{k-1})$, here G_{k-1} contains the same node set as X. However, the stationary distribution of these states differ even though they correspond to the same subgraph G_{k-1}. This motivates us to design a new re-weight function that considers states in $\mathcal{A}(G_{k-1})$ as a whole.

Our approach is to ignore the order of nodes in the states and view each state as a subgraph. We derive the improved unbiased estimator by defining the real-valued function on the subgraph and computing the sampling probability of the subgraph. The function defined for the subgraph G_{k-1} simply takes the sum over $f_i^k(X), \forall X \in \mathcal{A}(G_{k-1})$, i.e.,

$$\begin{aligned} F_i^k(G_{k-1}) &\triangleq \sum_{X_a\in\mathcal{A}(G_{k-1})} f_i^k(X_a) \\ &= |\mathcal{A}(G_{k-1})| f_i^k(X), \forall X \in \mathcal{A}(G_{k-1}). \end{aligned}$$

Following the definition of stationary distribution of the Markov chain, we define the nominal sampling probability of the subgraph G_{k-1} as

$$P(G_{k-1}) \triangleq \lim_{n\to\infty} \frac{1}{n}\sum_{t=1}^{n} \Pr[X_t = G_{k-1}] = \sum_{X_a\in\mathcal{A}(G_{k-1})} \pi_M(X_a).$$

Here $\{X_t\}_{t=1}^n$ is a sequence of the states. Suppose $\{X_t\}_{t=1}^n$ corresponds to the sequence of subgraphs $\{G_{k-1}^t\}_{t=1}^n$ (the node set of X_t induces the subgraph G_{k-1}^t). We need to show the following:

$$\frac{1}{n}\sum_{t=1}^{n} \frac{1}{\beta_i^k} \frac{F_i^k(G_{k-1}^t)}{P(G_{k-1}^t)} \to C_i^k \; a.s. \tag{8}$$

We define a new re-weight function for the state X based on above discussion. Similar to the definition of $\mathcal{A}(G_{k-1})$ in Section III-C, let $\mathcal{A}(X)$ denote the set of states in $\mathcal{M}^{(k-1)}$ that have the same node set as X. We define a new re-weight function as follows

$$W_i^k(X) \triangleq \frac{1}{\beta_i^k} \frac{|\mathcal{A}(X)|}{\sum_{X_a\in\mathcal{A}(X)} \pi_M(X_a)}.$$

The re-weight function W_i^k ignores the order of nodes in X. Let G_{k-1} denote the subgraph induced by the nodes in the state X. Compared with $w_i^k(X)$, $W_i^k(X)$ divides $\sum_{X_a\in\mathcal{A}(X)} \pi_M(X_a)/|\mathcal{A}(X)| = P(G_{k-1})/|\mathcal{A}(G_{k-1})|$ instead of $\pi_M(X)$ to remove the bias.The following theorem is a formal description of Eq. (8).

Theorem 3. *The average of the function $W_i^k(X) f_i^k(X)$*

$$\hat{C}_i^k \triangleq \frac{1}{n}\sum_{t=1}^{n} W_i^k(X_t) f_i^k(X_t) \tag{9}$$

is an asymptotic unbiased estimator of C_i^k for the graphlet g_i^k with $\beta_i^k \neq 0$.

Refer to the technical report [24] for the proof. We call Eq. (3) as the *basic estimator* and Eq. (9) as the *improved estimator*. Similar to Eq. (6), replacing $\widetilde{w}_i^k(X_t)$ with $W_i^k(X_t)/(2|E|)$ in Eq. (9), we can get the improved estimator for unknown $|E|$.

VI. Experimental Evaluation

A. Experimental setup

We test the performance of our proposed algorithm on various social networks. Table III lists the datasets used in our experiments. For all the datasets, we remove the directions, self-loops and multi-edges. We report the number of nodes

TABLE III: Summary of the datasets.

Name	Nodes	Edges	Description
Epinion [30]	76K	406K	Trust network from the online social network Epinion.
Slashdot [30]	77K	469K	Friend/foe links between the users of Slashdot social network.
Pokec [31]	1.6M	22.3M	Friendship network from the Slovak social network Pokec.
Flickr [31]	2.2M	22.7M	Social network of Flickr users and their friendship connections.
Twitter [32]	21.3M	265M	Graph about who follows whom on Twitter.
Weibo [32]	58.7M	261M	A micro-blogging service with millions of users in China.

and edges in the largest connected components (LCCs) of the graphs in the table. In fact, all the graphs are connected except Flickr, whose LCC contains 94% of the nodes. Exact counts of $3, 4$-node graphlets are computed with the state-of-the-art algorithm proposed in [9]. For 5-node graphlets, we obtain the ground truth with the method in [20]. Note that we ran the experiments on a Linux machine with 3.7GHz Intel Xeon processor. All the algorithms are implemented in C++. The source code is available [29].

B. Performance analysis

Error Metrics. To evaluate the performance of our proposed algorithm, we consider the following metrics. These error metrics provide a comprehensive picture of the error distribution.

- *Error of average estimate*: we consider the relative error $\frac{|E[\hat{C}_i^k]-C_i^k|}{C_i^k}$ as a measure of the unbiasedness of the estimators. Here $E[\hat{C}_i^k]$ is the mean estimate value across 1000 independent runs.
- *Confidence bound*: we construct a $[5\%, 95\%]$-confidence interval for the estimate z, which is defined as the interval $[\text{LB}, \text{UB}]$ such that $\Pr[z \leq \text{LB}] = 0.05$ and $\Pr[z \geq \text{UB}] = 0.95$. To estimate the confidence interval, we run the simulations for 1000 times, and use the 5^{th} and 95^{th} percentile as the estimated LB and UB respectively.
- *Mean of relative error* (MRE): we compute the average of $|\hat{C}_i^k - C_i^k|/C_i^k$ over 1000 independent runs. This measures how close our estimate is to the ground truth.
- *Normalized root mean square error* (NRMSE): for an estimator $\hat{C}_i^k$, the NRMSE is define as $$\text{NRMSE}(\hat{C}_i^k) = \frac{\sqrt{\mathbb{E}[(\hat{C}_i^k - C_i^k)^2]}}{C_i^k} = \frac{\sqrt{\text{Var}[\hat{C}_i^k] + (\mathbb{E}[\hat{C}_i^k] - C_i^k)^2}}{C_i^k}.$$ NRMSE is a combination of the variance and bias. When the estimator is unbiased, the NRMSE equals to $\sqrt{\text{Var}[\hat{C}_i^k]}/C_i^k$.

TABLE IV: Accuracy of the proposed estimator when the sample size equals to 20K, i.e., we perform the random walk for 20K steps. For 4, 5-node graphlets, we report the results for the improved estimator.

3-node triangle (g_2^3)

	C_2^3	$E[\hat{C}_2^3]$	$\frac{\|E[\hat{C}_2^3]-C_2^3\|}{C_2^3}$	LB	UB	MRE	NRMSE
Epinion	1.6M	1.6M	0.0007	1.57M	1.68M	0.015	0.019
Slashdot	552K	551K	0.0020	519K	585K	0.029	0.036
Pokec	32.6M	32.6M	0.0001	31.4M	33.7M	0.017	0.021
Flickr	837.6M	838.4M	0.0009	766.3M	915.5M	0.043	0.054
Twitter	17.3B	17.3B	0.0013	16.3B	18.4B	0.029	0.036
Weibo	213.0M	212.9M	0.0003	192.6M	235.1M	0.048	0.061

4-node cycle (g_3^4)

	C_3^4	$E[\hat{C}_3^4]$	$\frac{\|E[\hat{C}_3^4]-C_3^4\|}{C_3^4}$	LB	UB	MRE	NRMSE
Epinion	71.5M	71.5M	0.0005	67.8M	75.3M	0.026	0.033
Slashdot	27.2M	27.1M	0.0012	25.5M	28.9M	0.030	0.037
Pokec	358.3M	357.2M	0.0031	331.0M	387.9M	0.039	0.049
Flickr	219.0B	218.4B	0.0029	198.7B	240.0B	0.046	0.057

4-node chordal-cycle (g_5^4)

	C_5^4	$E[\hat{C}_5^4]$	$\frac{\|E[\hat{C}_5^4]-C_5^4\|}{C_5^4}$	LB	UB	MRE	NRMSE
Epinion	77.7M	77.6M	0.0015	72.0M	83.7M	0.037	0.046
Slashdot	16.8M	16.9M	0.0014	14.6M	19.2M	0.065	0.082
Pokec	466.8M	465.7M	0.0024	398.8M	597.6M	0.101	0.150
Flickr	372.9B	371.3B	0.0043	328.1B	417.5B	0.060	0.074

4-node clique (g_6^4)

	C_6^4	$E[\hat{C}_6^4]$	$\frac{\|E[\hat{C}_6^4]-C_6^4\|}{C_6^4}$	LB	UB	MRE	NRMSE
Epinion	5.8M	5.8M	0.0023	5.2M	6.5M	0.055	0.070
Slashdot	2.0M	2.0M	0.0015	1.6M	2.5M	0.112	0.140
Pokec	42.9M	42.8M	0.0030	37.9M	48.2M	0.058	0.073
Flickr	40.2B	39.9B	0.0062	33.8B	46.5B	0.078	0.097

5-node center-square (g_{19}^5)

	C_{19}^5	$E[\hat{C}_{19}^5]$	$\frac{\|E[\hat{C}_{19}^5]-C_{19}^5\|}{C_{19}^5}$	LB	UB	MRE	NRMSE
Epinion	205.6M	205.1M	0.0026	163.6M	251.6M	0.100	0.127
Slashdot	31.1M	31.1M	0.0007	21.3M	42.7M	0.163	0.208

5-node semi-clique (g_{20}^5)

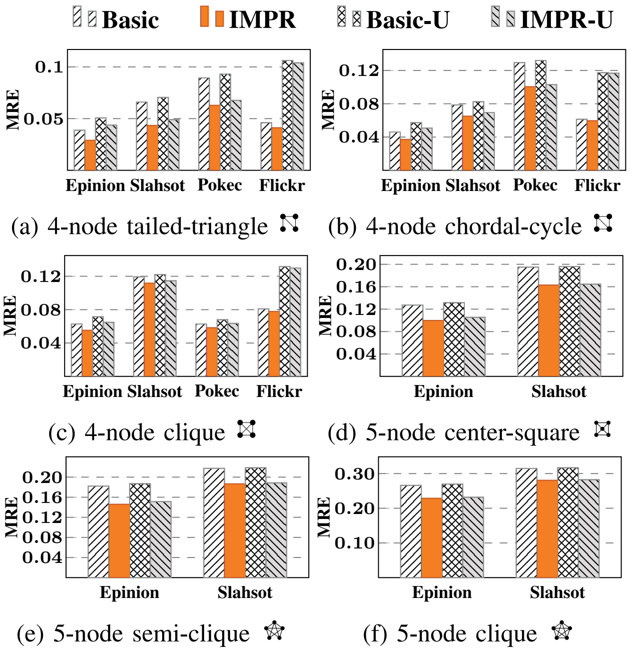
	C_{20}^5	$E[\hat{C}_{20}^5]$	$\frac{\|E[\hat{C}_{20}^5]-C_{20}^5\|}{C_{20}^5}$	LB	UB	MRE	NRMSE
Epinion	158.2M	158.5M	0.0018	116.5M	213.3M	0.146	0.186
Slashdot	54.8M	54.0M	0.0143	34.9M	77.0M	0.187	0.239

5-node clique (g_{21}^5)

	C_{21}^5	$E[\hat{C}_{21}^5]$	$\frac{\|E[\hat{C}_{21}^5]-C_{21}^5\|}{C_{21}^5}$	LB	UB	MRE	NRMSE
Epinion	17.4M	17.4M	0.0038	10.1M	26.5M	0.229	0.290
Slashdot	10.7M	10.4M	0.0264	4.8M	17.2M	0.281	0.356

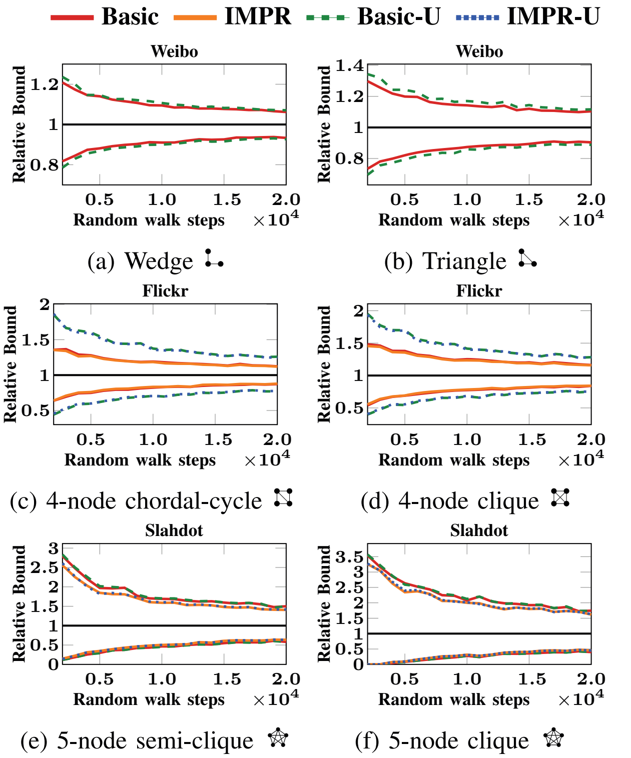
Accuracy. We demonstrate the accuracy of our proposed estimators in Table IV. Only the accuracy of estimators for graphlets $g_2^3, g_3^4, g_5^4, g_6^4, g_{19}^5, g_{20}^5, g_{21}^5$ is reported since their counts are the *smallest* among $3, 4, 5$-node graphlets respectively and they were observed to have the *lowest* accuracy. We apply the improved estimator for $4, 5$-node graphlets (for 3-node graphlets, the basic estimator and the improved estimator are the same). Due to the space limitation, we only show the results when the exact number of edges is known. The extremely high computation cost of the exact enumeration algorithms makes it difficult to obtain the $4, 5$-node graphlets counts for all graphs. Hence we only show the results of $4, 5$-node graphlets for the graphs whose ground truth can be obtained. The sample size equals to 20K. The findings are summarized as follows.

- **Our estimator is unbiased.** The 4th column of the table shows the error of average estimate over 1000 independent runs, which measures the unbiasedness of the estimators.



(a) 4-node tailed-triangle (b) 4-node chordal-cycle (c) 4-node clique (d) 5-node center-square (e) 5-node semi-clique (f) 5-node clique

Fig. 4: Compare the accuracy of the estimators. We compare between four estimators, i.e., the basic estimator when $|E|$ is known (Basic), the improved estimator with known $|E|$, the basic estimator with unknown $|E|$ (Basic-U), the improved estimator with unknown $|E|$ (IMPR-U). The sample size is 20K.



(a) Wedge (b) Triangle (c) 4-node chordal-cycle (d) 4-node clique (e) 5-node semi-clique (f) 5-node clique

Fig. 5: Convergence analysis of the estimators. Basic: the basic estimator when $|E|$ is known, IMPR: the improved estimator with known $|E|$, Basic-U: the basic estimator with unknown $|E|$, IMPR-U: the improved estimator with unknown $|E|$. We show the relative confidence bound of the estimates, i.e., LB/Actual and UB/Actual. Here Actual represents the actual counts.

The error is below 0.7% for all the reported graphlets except the 5-node graphlets of Slashdot. The results verify our claims in Theorem 2 and 3.

- **Our estimator is accurate.** First, we can see that the LB and UB are close to the ground truth. Second, the MRE presented in the table is less than 5% for triangles and 4-node cycle, $3\% \sim 11\%$ for g_5^4 and g_6^4, and $10\% \sim 30\%$ for the 5-node graphlets. These results are enough for many applications, e.g., the computation of graph kernel [25].
- **Our estimator has small variance.** Our estimator is unbiased, hence the NRMSE simply represents the relative variance of our estimator. For the $3, 4$-node graphlets in the table, the NRMSE is around $1.8\% \sim 15\%$. For 5-node graphlets, the NRMSE is below 0.4. Note that the NRMSE for unbiased estimator is an alternative of the confidence bound since the $[5\%, 95\%]$ confidence bound can be written as $\hat{C}_i^k \pm 1.96\sqrt{\text{Var}[\hat{C}_i^k]}$ theoretically.
- **Our estimator is practical.** We only use 20K random walk steps to estimate the graphlet counts. For most OSNs, one can easily crawl 20K users' profile within one day with just one machine [10]. Besides, given the sample size, the accuracy of the estimate does not depend on the size of the graphs, e.g., Slashdot and Twitter have the same MRE and NRMSE for the triangle estimate given 20K sampled nodes. However, the nodes of Twitter is 277 times of Slashdot.

Benefit of the improved estimator. We show the gain of the improved estimator (Eq. 9) in Fig. 4. For fair comparison, we use the same set of 20K samples and then apply the basic and the improved estimator separately. We choose the mean of relative error (MRE) as the accuracy measure. We also show the estimates when the number of edges is unknown. Note that the basic and the improved estimator are the same for 3-node graphlets. For 4-node graphlet, the improved estimator only changes the estimates when the subgraph contains triangle, i.e., g_4^4, g_5^4, g_6^4. From the Fig. 4, we can observe that

- The improved estimator reduces the error for *all the graphs* and *all the graphlet types* presented in the figure. For 4-node graphlets, the improved estimator reduces MRE by $0.001 \sim 0.029$ while for 5-node graphlets, it reduces MRE by $0.027 \sim 0.037$.

Convergence. To show the convergence properties, we choose graphs Weibo, Flickr, and Slashdot for 3, 4, 5-node graphlets respectively since they have the largest number of nodes for each sized graphlets whose ground truth can be obtained. We only present the results of two graphlet types for each sized graphlets. Fig. 5 presents the relative confidence bound, i.e., LB/(True count) and UB/(True count) with increasing sample size. We vary the sample size in increment of 1K. For each choice of sample size, we run 1000 independent simulations. From the figure, we can see that the estimates converge to the ground truth rapidly. The LB and UB are balanced over the ground truth value.

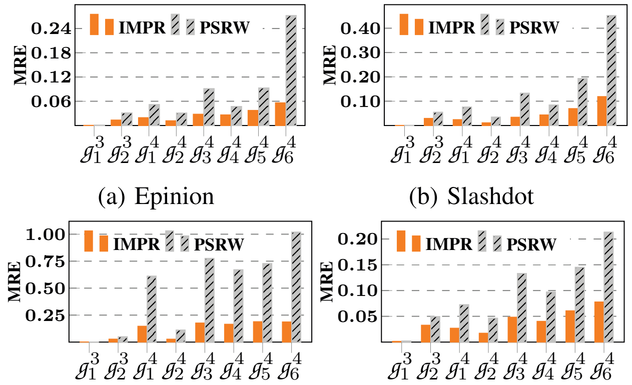
Effect of estimated edges. Fig 4 and 5 also demonstrate the results when $|E|$ is replaced with the estimated edge cardinality. However, we can see that the estimated edge cardinality does not degenerate the performance too much. Except Flickr, the effect is negligible. And the MRE of estimates in Flickr increases less than 0.05 with estimated edge cardinality. Besides, from Fig 5, we can see that the results with estimated edge cardinality approach these with true $|E|$ quickly, which implies the effect of estimated edge cardinality becomes smaller when the sample size increases.

C. Comparison with previous work

We compare our improved estimators with the state-of-the-art methods *PSRW* [16] that are designed for graphs with restricted access. *PSRW* is only capable of estimating relative graphlet counts. The relative graphlet count of g_i^k is defined as $c_i^k \triangleq C_i^k / \sum_{j=1}^{|\mathcal{G}^k|} C_j^k$, which can be computed immediately with the graphlet counts. To estimate the relative counts of k-node subgraphs, *PSRW* performs random walk on a *super graph*. Each node in the super graph is a $(k-1)$-node induced connected subgraph of the original graph. *PSRW* considers two steps of the random walk on the super graph as a k-node subgraph sample. The neighbors of nodes in the super graph can be generated on the fly. Note that *PSRW* cannot be easily extended to estimate graphlet counts. To estimate the relative counts of subgraphs with our method, we define the ratio estimator $\hat{c}_i^k \triangleq \hat{C}_i^k / \sum_{j=1}^{|\mathcal{G}^k|} \hat{C}_j^k$. In fact, the $2|E|$ in the numerator and denominator cancels out in $\hat{c}_i^k$. Hence the estimator $\hat{c}_i^k$ can be computed *without knowing* $|E|$.

Accuracy. The accuracy of our proposed method and *PSRW* is compared in Fig. 6. For both methods, the sample size equals to 20K. For ease of presentation, we measure the accuracy on four graphs Epinion, Slashdot, Pokec and Flickr for all $3,4$-node graphlets. We use the error metric *mean of relative error* (MRE). Our proposed method outperforms *PSRW* significantly. For example, the MRE of our method on estimating the relative count of 4-node clique (g_6^4) is $4 \sim 5$ times smaller than that of *PSRW*. Our estimator shows excellent empirical accuracy for relative counts estimation, e.g., for the graph Flickr, the MRE of the relative counts estimation is below 8% for all $3,4$-node graphlets.

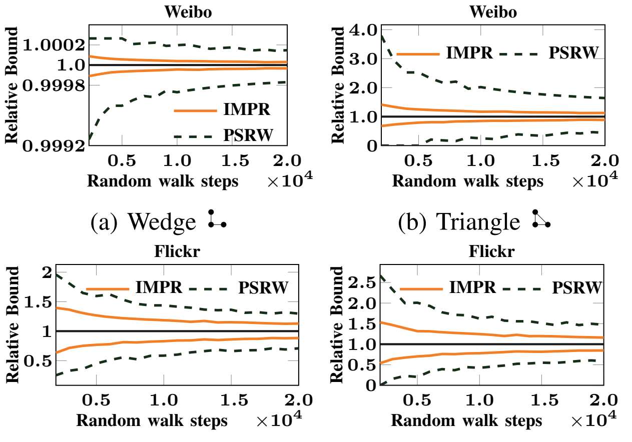
Convergence. Fig. 7 compares the convergence performance of the estimators. Both of our proposed method and *PSRW* converge to the actual relative counts as the sample size increases. However, our proposed estimator has much more tight bound centered around the ground truth. As shown in Fig. 7, the gap between LB and UB of our estimator is no more than half of that produced by *PSRW*. It implies that our proposed method also shows extraordinary performance on the estimation of relative counts.



(a) Epinion (b) Slashdot

(c) Pokec (d) Flickr

Fig. 6: Compare the accuracy of our proposed estimator IMPR and prior state-of-the-art method PSRW. The sample size for both methods is 20K.



(a) Wedge (b) Triangle

(c) 4-node chordal-cycle (d) 4-node clique

Fig. 7: Compare the convergence performance of the estimators. We show the relative confidence bound of the estimates, i.e., LB/Actual and UB/Actual. Here Actual represents the actual relative counts of subgraphs.

VII. Related Work

Exact counting. The state-of-the-art memory-based method for $3,4$-node graphlet counting is proposed by Ahmed at el. [9]. This method's core idea is to count only a few graphlet types for each edge, then derive the exact counts for other graphlet types by combining these counts with combinatorial equations. Hočevar and Demšar proposed a combinatorial graphlet counting method [20] which leverages orbits and a system of linear equations. The algorithm in [20] is the state-of-the-art 5-node graphlet exact counting method. There are also multiple close works in the area of subgraph enumeration, e.g., [33]–[36].

Sampling methods. Generally speaking, the access assumption of the graphs can be divided into three categories: (i) full access, i.e., graphs could fit in the main memory and random access to graph data is allowed, (ii) restricted access, i.e., full graph topology is not available, but APIs are provided to retrieve information, (iii) streaming access, i.e., edges of graphs appear in streaming. Various sampling methods have been designed for different settings. Usually methods designed for a specific setting have the best performance in that setting. The sampling methods with full access assumption include the wedge sampling [3], the 3-path sampling [2], Moss [18], GRAFT [37], etc. For streaming graphs, works on graphlet counts estimation include the methods using independent edge sampling [5], [38]–[41] and reservoir sampling [6], [7].

Most relevant to our work are the methods that are designed for graphs with restricted access, e.g., [11], [16], [19]. These sampling algorithms focus on estimating the *relative graphlet counts*. In [19], the authors used the Metropolis-Hasting random walk to estimate the relative counts of $3, 4, 5$-node subgraphs. In [11], the authors designed a random-walk based method to estimate the clustering coefficient (the relative count of triangles among all 3-node subgraphs). Wang et al. [16] proposed three random walk-based algorithms to estimate the relative counts of any k-node subgraphs. *Pairwise subgraph random walk (PSRW)* has the best performance among them and is the state-of-the-art method to estimate the relative counts. Note that the methods in [11], [16], [19] cannot be easily extended to estimate the graphlet counts.

VIII. Conclusion

We propose an efficient random walk-based method to estimate the number of $3, 4, 5$-node subgraphs in OSNs. Our algorithm can also be easily extended to graphlets of larger size. Both theoretical analysis and experimental evaluation validate the unbiasedness and convergence of our proposed estimators. Our estimators show excellent empirical accuracy for graphlet counts estimation. Comparison with prior state-of-the-art method also shows the superb performance of our estimators in estimating relative graphlet counts.

Acknowledgment

The work of John C.S. Lui is supported in part by RGC 415013 and Huawei Research Grant.

References

[1] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, “Measurement and analysis of online social networks,” in *IMC*, 2007.

[2] M. Jha, C. Seshadhri, and A. Pinar, “Path sampling: A fast and provable method for estimating 4-vertex subgraph counts,” in *WWW*, 2015.

[3] C. Seshadhri, A. Pinar, and T. G. Kolda, “Triadic measures on graphs: The power of wedge sampling,” in *SDM*, 2013.

[4] J. Zhao, J. C. Lui, D. Towsley, P. Wang, and X. Guan, “Tracking triadic cardinality distributions for burst detection in social activity streams,” in *COSN*, 2015.

[5] Y. Lim and U. Kang, “Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams,” in *KDD*, 2015.

[6] M. Jha, C. Seshadhri, and A. Pinar, “A space-efficient streaming algorithm for estimating transitivity and triangle counts using the birthday paradox,” *TKDD*, 2015.

[7] L. D. Stefani, A. Epasto, M. Riondato, and E. Upfal, “Trièst: Counting local and global triangles in fully-dynamic streams with fixed memory size,” 2016.

[8] J. Ugander, L. Backstrom, and J. Kleinberg, “Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections,” in *WWW*, 2013.

[9] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, “Efficient graphlet counting for large networks,” in *ICDM*, 2015.

[10] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, “Walking in facebook: A case study of unbiased sampling of osns,” in *INFOCOM*, 2010.

[11] L. Katzir and S. J. Hardiman, “Estimating clustering coefficients and size of social networks via random walk,” *TWEB*, 2015.

[12] L. Katzir, E. Liberty, and O. Somekh, “Estimating sizes of social networks via biased sampling,” in *WWW*, 2011.

[13] Z. Zhou, N. Zhang, and G. Das, “Leveraging history for faster sampling of online social networks,” *PVLDB*, 2015.

[14] R.-H. Li, J. Yu, L. Qin, R. Mao, and T. Jin, “On random walk based graph sampling,” in *ICDE*, 2015.

[15] C.-H. Lee, X. Xu, and D. Y. Eun, “Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling,” in *SIGMETRICS*, 2012.

[16] P. Wang, J. C. S. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan, “Efficiently estimating motif statistics of large networks,” *TKDE*, 2014.

[17] R. Diestel, *Graph Theory, 4th Edition*, ser. Graduate texts in mathematics. Springer, 2012.

[18] P. Wang, J. Tao, J. Zhao, and X. Guan, “Moss: A scalable tool for efficiently sampling and counting 4- and 5-node graphlets,” *arXiv:1509.08089*, 2015.

[19] M. Bhuiyan, M. Rahman, and M. Al Hasan, “GUISE: Uniform sampling of graphlets for large graph analysis,” in *ICDM*, 2012.

[20] T. Hočevar and J. Demšar, “A combinatorial approach to graphlet counting,” *Bioinformatics*, 2014.

[21] L. Lovász, “Random walks on graphs: A survey,” in *Combinatorics, Paul Erdős is Eighty*. János Bolyai Mathematical Society, 1996.

[22] A. B. Owen, “Monte carlo theory, methods and examples,” http://statweb.stanford.edu/~owen/mc/, 2013.

[23] H. OLLE, *Finite Markov Chains and Algorithmic Applications*. Cambridge University Press, 2000.

[24] X. Chen and J. Lui, “Mining graphlet counts in online social networks.” [Online]. Available: http://tinyurl.com/GraphletCountOSN

[25] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, “Efficient graphlet kernels for large graph comparison,” in *Artificial Intelligence and Statistics*, 2009.

[26] A. Dasgupta, R. Kumar, and T. Sarlos, “On estimating the average degree,” in *WWW*, 2014.

[27] A. Mohaisen, A. Yun, and Y. Kim, “Measuring the mixing time of social graphs,” in *IMC*, 2010.

[28] L. Addario-Berry and T. Lei, “The mixing time of the newman: Watts small world,” in *SODA*, 2012.

[29] [Online]. Available: http://tinyurl.com/GraphletCountOSNCode

[30] J. Leskovec and A. Krevl, “SNAP Datasets: Stanford large network dataset collection,” http://snap.stanford.edu/data, 2014.

[31] “KONECT Datasets: The koblenz network collection,” http://konect.uni-koblenz.de, 2015.

[32] R. A. Rossi and N. K. Ahmed, “The network data repository with interactive graph analytics and visualization,” in *AAAI*, 2015. [Online]. Available: http://networkrepository.com

[33] F. N. Afrati, D. Fotakis, and J. D. Ullman, “Enumerating subgraph instances using map-reduce,” in *ICDE*, 2013.

[34] Y. Shao, B. Cui, L. Chen, L. Ma, J. Yao, and N. Xu, “Parallel subgraph listing in a large-scale graph,” in *SIGMOD*, 2014.

[35] L. Lai, L. Qin, X. Lin, and L. Chang, “Scalable subgraph enumeration in mapreduce,” *VLDB*, 2015.

[36] H. Kim, J. Lee, S. S. Bhowmick, W.-S. Han, J. Lee, S. Ko, and M. H. Jarrah, “DUALSIM: Parallel subgraph enumeration in a massive graph on a single machine,” in *SIGMOD*, 2016.

[37] M. Rahman, M. A. Bhuiyan, and M. A. Hasan, “Graft: An efficient graphlet counting method for large graph analysis,” *TKDE*, vol. 26, 2014.

[38] P. Wang, J. Lui, and D. Towsley, “Minfer: Inferring motif statistics from sampled edges,” in *ICDE*, 2016.

[39] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Dimakis, “Beyond triangles: A distributed framework for estimating 3-profiles of large graphs,” in *KDD*, 2015.

[40] ——, “Distributed estimation of graph 4-profiles,” in *WWW*, 2016.

[41] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella, “Graph sample and hold: A framework for big-graph analytics,” in *KDD*, 2014.