

Mining Graphlet Counts in Online Social Networks

XIAOWEI CHEN and JOHN C. S. LUI, The Chinese University of Hong Kong

Counting subgraphs is a fundamental analysis task for online social networks (OSNs). Given the sheer size and restricted access of OSN, 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 article, 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 improved estimators, which increases the accuracy of the estimation with 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 algorithms. 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 20,000 sampled nodes. Detailed comparison with the state-of-the-art methods demonstrates that our algorithm is 2–10 times more accurate.

CCS Concepts: • **General and reference** → **Estimation**; • **Mathematics of computing** → **Markov-chain Monte Carlo methods**; • **Networks** → **Online social networks**; • **Theory of computation** → **Graph algorithms analysis**; **Random walks and Markov chains**;

Additional Key Words and Phrases: Graphlet count, online social networks, random walk, Markov chain Monte Carlo

ACM Reference format:

Xiaowei Chen and John C. S. Lui. 2018. Mining Graphlet Counts in Online Social Networks. *ACM Trans. Knowl. Discov. Data.* 12, 4, Article 41 (April 2018), 38 pages.
<https://doi.org/10.1145/3182392>

1 INTRODUCTION

Analyzing properties of online social networks (OSNs) has attracted extensive attention because of their increasing popularity, significant importance, and diverse applications [37]. 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” [23]. 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 [12, 24,

The work by John C.S. Lui is supported in part by GRF 14208816 and Huawei’s Grant. A preliminary version of the article appears in the proceedings of IEEE ICDM’16 as [8].

Authors’ addresses: X. Chen and J. C. S. Lui, Ho Sin-Hang Engineering Building, The Chinese University of Hong Kong, Shatin N.T. Hong Kong, The People’s Republic of China; emails: {xwchen, cslui}@cse.cuhk.edu.hk.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 ACM 1556-4681/2018/04-ART41 \$15.00

<https://doi.org/10.1145/3182392>

[34, 46, 55]. Recently, Ugander et al. [51] 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 [4], anomaly and event detection [34, 55], and nodes classification [51].

However, it is a challenging task to compute graphlet counts for OSNs. First, 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 [4]. It is natural to resort to “efficient-to-compute estimations” of the graphlet counts in these cases. Second, most OSNs have restrictions on the access to nodes and edges of the underlying networks. On one hand, networks with massive size are usually stored in local or remote databases, which restricts the random access to nodes and edges. For owners of the OSNs, the network data need accessing via Application Programming Interface (API) of the databases. On the other hand, for researchers in academic community, the networks are usually unknown beforehand [17, 37] and they 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 for both of owners of OSNs and academic researchers. To address these challenges, graph sampling via crawling has been widely applied for OSN measurement [17, 26, 27, 33, 56]. 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 [17, 31, 33], 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 structures. To estimate graphlet counts, we need to examine the local structures 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 improved estimators that make 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 real-world social networks. 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-art random walk-based methods [7, 52] that are 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 methods 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.

In this article, we have several extensions as compared with our previous conference paper [8]. First of all, we propose a new improved estimator, which is designed for an important class of access models. Second, we devote a new section to explain our implementation and its intricacy in details. Third, this article provides all the proofs of the theoretical results and adds more illustrative examples and details to make our presentation clearer. Finally, we extend the experiments by adding evaluation on the new improved estimator and giving additional comparison with another state-of-the-art method recently proposed in [7].

Paper organization. The rest of this article is organized as follows. In Section 2, we discuss the related works of our article. The notations and background are provided in Section 3. We describe the algorithm framework and improved estimators in Sections 4 and 6, respectively. The analytical bound of our estimators is presented in Section 5. After showing the performance of our estimators and comparing with the state-of-the-art methods, we conclude in Section 9. The supplementary materials are provided in the Appendix.

2 RELATED WORK

Previous works on subgraph counting include exact counting methods and estimation methods. In the following, we give a brief review on these methods.

2.1 Exact Counting

The exact counting of graphlets has extensive computation cost since the number of possible k -node graphlets grows exponentially with k in $O(|V|^k)$, where V is the set of nodes in the graph. Shervashidze et al. [48] show that for graphs with degree bound Δ , the exact number of all graphlets of size k can be determined in time $O(|V| \cdot \Delta^k)$. Counting 3-, 4-, 5-node graphlets attracts more attention than the general k -node graphlet counting. Alon et al. presented the most time-efficient algorithm to compute the triangles via the matrix multiplication [5]. Despite its fast running time of $O(|E|^{1.41})$, the algorithm is not practical due to its high space complexity of $O(|V|^2)$ associating with the matrix computation. A more practical algorithm “edge-iterator” in [45] counts triangles with fast running time of $O(|E|^{1.5})$ and a reasonable space requirement. The state-of-the-art memory-based method for 3-, 4-node graphlet counting is proposed in [4]. This method’s core idea is to count only a few graphlet types for each edge in parallel, then derive the exact counts for other graphlet types by combining these counts with combinatorial equations. The time complexity of this method is $O(|E| \cdot \Delta)$, $O(|E| \cdot \Delta \cdot T_{max})$ and $O(|E| \cdot \Delta \cdot S_{max})$ for counting triangles, 4-node cliques and 4-node cycles, respectively. Here, Δ is the degree bound, T_{max} is the maximum number of triangles incident to an edges, and S_{max} is the maximum number of 4-node stars incident to an edge. Hočevar and Demšar proposed a combinatorial graphlet counting method [21], which leverages orbits and a system of linear equations. The equations connect the counts of orbits for graphlets up to 5-nodes and allow computing all orbit counts by enumerating only one. The algorithm in [21] is the state-of-the-art 5-node graphlet exact counting method. Recently, [41] proposed an efficient 4-, 5-node graphlet counting method named “Escape,” which is built on cutting a graphlet pattern into smaller ones, and uses counts of smaller patterns to get larger counts. The single thread algorithm *Escape* has smaller running time in counting 4-node graphlets than the method proposed by Ahmed et al. when both of the methods are restricted to use a single core. There are also multiple close works in the area of subgraph enumeration, e.g., [2, 28, 30, 47].

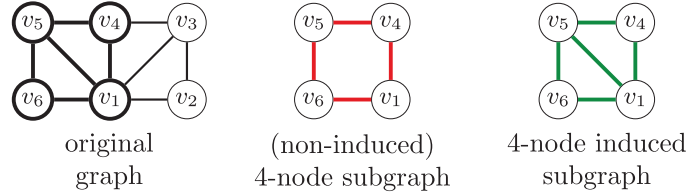


Fig. 1. An example of subgraph and induced subgraph.

2.2 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, and (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 and are not recommended to be adopted for other settings. The sampling methods with full access assumption include the wedge sampling [46], the three-path sampling [23], Moss [54], GRAFT [42], and so on. For streaming graphs, works on graphlet counts estimation include the methods using independent edge sampling [3, 14, 15, 34, 53] and reservoir sampling [12, 24] cannot be easily extended to estimate the graphlet counts.

3 PRELIMINARIES

3.1 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 Figure 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.

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 $u, v \in V$ [13].

Graphlet. Graphlets are defined as *non-isomorphic, connected, and induced* subgraphs in graphs. Let \mathcal{G}^k denote the family of k -node graphlets, i.e., $\mathcal{G}^k = \{g_1^k, \dots, g_{m^k}^k\}$. Here, m^k denotes the number of all distinct k -node graphlets. To illustrate, in Section 4, Table 2 depicts \mathcal{G}^3 and \mathcal{G}^4 , while Table 3 depicts \mathcal{G}^5 . The second row of the Tables 2 and 3 show all the 3-, 4-, 5-node graphlets. We can see that $m^k = \{2, 6, 21\}$ for $k = 3, 4, 5$, respectively.

Problem definition: Given the family of k -node graphlets $\mathcal{G}^k = \{g_1^k, \dots, g_{m^k}^k\}$, 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 $\{C_1^k, \dots, C_{m^k}^k\}$ efficiently.

We refer to $\{C_1^k, \dots, C_{m^k}^k\}$ as the *graphlet counts*. The computation of graphlet counts is usually restricted to graphlets of no more than 5 nodes [4, 6, 21, 23, 26, 54] due to the extremely high computation cost. Besides, various applications, e.g., [48, 51] focus on graphlets with less than 6 nodes since graphlets with up to 5 nodes have the best cost–benefit tradeoff [6]. In this work, our aim is to efficiently compute C_i^k , for $k = 3, 4, 5$.

3.2 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 (e.g., user name and privacy settings) [17].

Random walk: Random walk-based methods fit in with the restricted access setting 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 P , where

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

Let $\pi(v)$ be the steady-state probability of node v . It is easy to show that $\pi(v) = d_v / (2|E|)$, $v \in V$ [35]. Note that these steady state probabilities (a.k.a. stationary distribution) are important for removal of the random walk sampling bias.

Theoretical guarantee: The mathematical foundations of the random walk root at theories for finite Markov chain. In the following, we review the Strong Law of Large Numbers (SLLN, a.k.a. ergodic theorem) for the Markov chain [25, 36, 39], 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 [43]. 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 3.1. *Suppose $\{X_n\}$ is a finite, irreducible Markov chain with stationary distribution $\boldsymbol{\pi}$. As $n \rightarrow \infty$, we have*

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

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.

4 ALGORITHMIC FRAMEWORK

Our algorithm generates the subgraph samples through *consecutive steps of the random walk*. Furthermore, we leverage the neighbors of the nodes visited along the random walk. We correct

Table 1. Summary of Notations

Notation	Meaning
$G = (V, E)$	Underlying graph with node set V and edge set E
$G_k = (V_k, E_k)$	k -node subgraph with node set V_k and edge set E_k
$\mathcal{N}(v)$	The set of neighbors of node v
\mathcal{G}_i^k	i -th type k -node graphlet
C_i^k	Count of the graphlet \mathcal{G}_i^k
\hat{C}_i^k	Estimated count of the graphlet \mathcal{G}_i^k
$\mathcal{M}^{(l)}$	State space of the expanded Markov chain ¹
$X = (v_1, \dots, v_l)$	State in $\mathcal{M}^{(l)}$, v_1, \dots, v_l are the l nodes contained in state X
π_M	Stationary probability of the expanded Markov chain
$V(X)$	Set of nodes in the state X
$\mathcal{B}(G_k)$	Set of states which can find subgraph $G_k = (V_k, E_k)$, i.e., $\mathcal{B}(G_k) \triangleq \{X X \in \mathcal{M}^{(k-1)}, V(X) = k - 1, V(X) \in V_k\}$
$\mathcal{A}(G_k)$	The set of states whose node set is the same as $G_k = (V_k, E_k)$, i.e., $\mathcal{A}(G_k) \triangleq \{X X \in \mathcal{M}^{(k)}, V(X) = V_k\}$
$\mathcal{A}(X)$	The set of states in $\mathcal{M}^{(l)}$ whose node set is the same as state X
β_i^k	The cardinality of the set $\mathcal{B}(G_k)$ where G_k is isomorphic to \mathcal{G}_i^k
α_i^k	The cardinality of the set $\mathcal{A}(G_k)$ where G_k is isomorphic to \mathcal{G}_i^k

the sampling bias via *importance sampling* [40]. For ease of presentation, we summarize the main notations in Table 1.

4.1 Basic Idea

We first describe the high level idea of our algorithm framework. 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 nodes in V_k are *available*, i.e., we have obtained the $\mathcal{N}(v)$ for all $v \in V_k$ by querying 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, i.e., one has not obtained $\mathcal{N}(v)$ yet. 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 Figure 2(a). Assume, we already obtain the neighbors of nodes 4 and 7. According to the definition, 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 since only $\mathcal{N}(4)$ is obtained. In other words, we cannot determine whether 5 and 8 are connected with only $\mathcal{N}(4)$.

Since our goal is to compute the graphlet counts, only graphlets, i.e., connected subgraphs, are considered in this work. We now discuss how to obtain the k -node connected subgraph samples. 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 *visible* k -node subgraph samples. 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 Section 3.2. If the random walk fails to visit $k - 1$ distinct nodes with $k - 1$ steps, we just discard such consecutive $k - 1$ steps and continue the random walk.

¹The expanded Markov chain with state space $\mathcal{M}^{(l)}$ remembers l consecutive steps of the corresponding random walk.

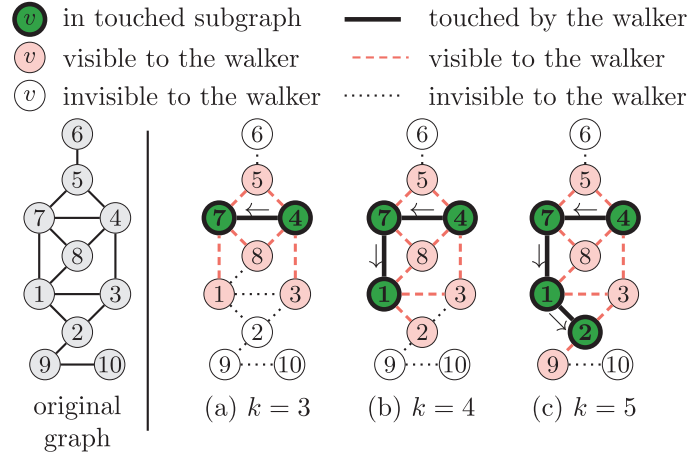


Fig. 2. Illustration of the basic idea. (a) When $k = 3$, assume we visit the nodes 4 and 7 sequentially. Then, we obtain the touched subgraph $\{4, 7\}$, and we can observe 2 visible triangles (\triangle) and 2 visible wedges (\blacktriangleright). (b) When $k = 4$, suppose we walk for three steps and 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 (\triangle), 1 chordal-cycle (\blacktriangleright), and 1 tailed-triangle (\blacktriangleright). (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 \blacktriangleright , 1 \blacktriangleright , 1 \blacktriangleright , and 1 \blacktriangleright visible to the walker; here the 4-node line $\{4, 7, 1, 2\}$ is the 4-node touched subgraph.

Our algorithm explores the neighborhood of the node set V_{k-1} to get k -node graphlet samples, which is motivated by the fact that the random walker needs to know the exact set of neighbors when it randomly jumps to next node, and most APIs support the function to return a set of neighbors when querying a node. Since API calls are expensive, it is reasonable to leverage the neighborhood information obtained during 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 $\mathcal{N}(V_{k-1}) = \{\cup_{v \in V_{k-1}} \mathcal{N}(v)\} \setminus V_{k-1}$. 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 the random walker when the nodes in V_{k-1} are visited during the random walk. We use these k -node visible subgraphs as the obtained k -node subgraph samples. Note that we do not need to query 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 then 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 Figure 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 [40] to remove the bias. We explain the detailed derivation of unbiased estimator in following subsections.

4.2 Mathematical Description

Now, we translate the basic idea to the 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 $X = (v_1, \dots, v_l)$, where $(v_i, v_{i+1}) \in E, 1 \leq i \leq l-1$. 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 $X_t = (v_1, \dots, v_l)$, 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 $X_{t+1} = (v_2, \dots, v_{l+1})$.

Example. Consider Figure 2(c). In this case, we have $k = 5$ and $l = k - 1 = 4$. Suppose, the walker already visits nodes 4 and 7 and is currently at node 1. Then, we say that the walker is at state $(4, 7, 1)$. If the random walker proceeds to node 2 in the next step, then the expanded Markov chain transits to state $(7, 1, 2)$.

For any two states $X_i = (v_{i_1}, \dots, v_{i_l})$ and $X_j = (v_{j_1}, \dots, v_{j_l})$ in $\mathcal{M}^{(l)}$, the transition matrix \mathbf{P}_M of the expanded Markov chain is

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

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 [18]. Let π_M denote the stationary distribution of the expanded Markov chain. For the state $X = (v_1, \dots, v_l) \in \mathcal{M}^{(l)}$, we have

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

We now define the function $f_i^k : \mathcal{M}^{(l)} \rightarrow \mathbb{R}$. Let $V(X) \triangleq \{v_1, \dots, v_l\}$ denote the set of nodes in $X = (v_1, \dots, v_l)$, 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 $\mathcal{S}(X) \triangleq \{G_k(V_k) | V_k = V(X) \cup \{v\}, v \in \mathcal{N}(V(X))\}$. Here, $G_k(V_k)$ denotes the subgraph induced by V_k . Formally, the real-valued function $f_i^k(X)$ can be written as follows:

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

Example. (a) *Example of $\mathcal{S}(X)$:* Figure 3 gives an example of $\mathcal{S}(X)$, where there are two-tailed triangles (g_4^4, \mathfrak{N}), one clique (g_6^4, \mathfrak{M}), and one cycle (g_5^4, \mathfrak{N}). (b) *Example of $f_i^k(X)$:* Refer to Figure 3. We have $f_4^4(X) = 2$, $f_6^4(X) = 1$ and $f_5^4(X) = 1$.

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 function f_i^k .

4.3 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

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

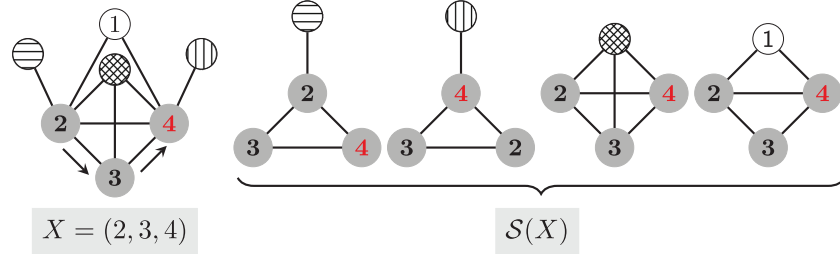


Fig. 3. Example of $\mathcal{S}(X)$. Suppose nodes 2, 3, 4 are visited sequentially and the random walker is at node 4. The currently visited state is $X = (2, 3, 4)$.

We first define the number of states that can 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 that can find subgraph $G_k = (V_k, E_k)$ as

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

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

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

Finally, the re-weight function is

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

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 can be found by β_i^k distinct states. 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 (\bowtie). 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 [40] and SLLN, we have the following theorem.

THEOREM 4.1. *The average of the function $w_i^k(X)f_i^k(X)$ is*

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

which is an asymptotic unbiased estimator of C_i^k , i.e., count of graphlet g_i^k , when $\beta_i^k \neq 0$.

Refer to Appendix A.1 for details of the proof. Algorithm 1 demonstrates the sampling procedure.

ALGORITHM 1: Unbiased Estimate of k -Node GraphletCounts**Input:** sampling budget n , graphlet size k , input graph G **Output:** unbiased estimate of C_i^k for graphlet g_i^k with $\beta_i^k \neq 0$

```

1:  $\hat{C}_i^k \leftarrow 0, \forall 1 \leq i \leq |\mathcal{G}^k|$ 
2:  $X = (v_1, \dots, v_{k-1}) \leftarrow$  initial  $k - 1$  random walk steps
3: random walk step counter  $t \leftarrow 0$ 
4: while  $t < n$  do
5:   for  $i \in \{1, \dots, |\mathcal{G}^k|\}$  do
6:     if  $\beta_i^k \neq 0$  then
7:        $\hat{C}_i^k \leftarrow \hat{C}_i^k + w_i^k(X) f_i^k(X) / n$ 
8:        $v_{t+k} \leftarrow$  uniformly choose a neighbor of  $v_{t+k-1}$ 
9:        $X \leftarrow (v_{t+2}, \dots, v_{t+k})$ 
10:       $t \leftarrow t + 1$ 
11: return  $[\hat{C}_1^k, \dots, \hat{C}_{|\mathcal{G}^k|}^k]$ 

```

4.3.1 *Computation of β_i^k .* The remaining task is to compute β_i^k , which is part of the re-weight function in Equation (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.,

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

Intuitively, $|\mathcal{A}(G_{k-1})|$ equals to the number of ways to walk through V_{k-1} during the random walk. Theoretically, it is twice of the number of *Hamilton paths*² in G_{k-1} (each Hamilton path is counted for both directions). Hence, 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} . 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, 48, 55]. Algorithm 2 in Appendix B shows the computation of α_j^{k-1} .

Let \mathcal{H}_{k-1} denote the set of $(k-1)$ -node connected subgraphs in G_k . The relationship between $\mathcal{B}(G_k)$ and $\mathcal{A}(G_{k-1}), \forall G_{k-1} \in \mathcal{H}_{k-1}$ is as follows.

$$\mathcal{B}(G_k) = \bigcup_{G_{k-1} \in \mathcal{H}_{k-1}} \mathcal{A}(G_{k-1})$$

Let t_j denote the count of $(k-1)$ -node graphlet g_j^{k-1} in G_k . Here, G_k is isomorphic to g_i^k . It is easy to verify that

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

The computation procedure of β_i^k is illustrated in Algorithm 3 Appendix B. Tables 2 and 3 list the values of α_i^k and β_i^k for all the 3-, 4-, 5-node graphlets.

Example. For a triangle G_3 induced by $\{u, v, w\}$, $\mathcal{B}(G_3) = \{(u, v), (v, u), (v, w), (w, v), (u, w), (w, u)\}$. 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 four triangles in the 4-node clique (g_6^4 , \mathfrak{K}_4). Based on Equation (4), we have $\beta_6^4 = 4 \times 6 = 24$.

²A path in G contains each vertex of G is a Hamilton path.

Table 2. Coefficient α_i^k and β_i^k for 3, 4-Node Graphlets

Graphlets	\mathcal{G}_1^3	\mathcal{G}_2^3	\mathcal{G}_1^4	\mathcal{G}_2^4	\mathcal{G}_3^4	\mathcal{G}_4^4	\mathcal{G}_5^4	\mathcal{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

Table 3. 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

4.3.2 Special Graphlets. We like to point out that some special graphlets, such as line, cycle, and clique are of more interest for applications in social networks. The β_i^k for any k -node line is always 4. Similarly, for k -node cycles, we have $|\mathcal{B}(\text{cycle})| = 2k$. The k -node clique has $|\mathcal{B}(\text{clique})| = k!$. For the special graphlets, we have just listed, the computation of β_i^k is easier and can be computed in constant time.

4.3.3 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 [11] (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

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

to estimate the number of edges. Here, π is the stationary distribution of the SRW. Assume the sequence of visited node is v_1, \dots, v_{n+k-2} and the corresponding sampled states are X_1, \dots, X_n . Here, $X_i \triangleq (v_i, \dots, v_{i+k-2})$. According to Equation (5), the number of edges can be estimated as

$$|V| \frac{n+k-2}{2 \sum_{t=1}^{n+k-2} 1/d_{v_t}} \rightarrow |E| \text{ a.s.} \quad (6)$$

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

$$\hat{C}_i^k \triangleq |V| \left(\frac{n+k-2}{n} \right) \left(\frac{\sum_{t=1}^n \tilde{w}_i^k(X_t) f_i^k(X_t)}{\sum_{t=1}^{n+k-2} 1/d_{v_t}} \right). \quad (7)$$

The unbiasedness of Equation (7) can be proved by combining Theorem 4.1 and Equation (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) [37]. The LCCs

are enough to represent the properties of the whole graphs. Besides, we can use the state-of-the-art algorithm [26] with high accuracy to estimate number of nodes in the LCCs.

4.4 Estimators for 3-, 4-, 5-Node Graphlets

We now discuss the detailed estimators for 3-, 4-, 5-node graphlet counts in this subsection. Additionally, we propose a novel method to estimate the count of 5-node star ($\mathcal{G}_3^5, \mathfrak{X}$).

4.4.1 Estimator for 3-Node Graphlets. For 3-node graphlets, each edge $(u, v) \in E$ corresponds to two states (u, v) and (v, u) in the state space of $\mathcal{M}^{(2)}$. Counting the “common nodes” in $\mathcal{N}(u)$ and $\mathcal{N}(v)$ is the key operation for 3-node graphlet counts estimation. Specifically, for each edge (u, v) , the number of wedges (open triangles) containing edge (u, v) is $d_u + d_v - 2|\mathcal{N}(u) \cap \mathcal{N}(v)| - 2$, and the number of triangles containing edge (u, v) is $|\mathcal{N}(u) \cap \mathcal{N}(v)|$. The re-weight function $w_1^3 = \frac{1}{4} \cdot 2|E| = |E|/2$ and $w_2^3 = \frac{1}{6} \cdot 2|E| = |E|/3$. Suppose, the sequence of nodes visited during the random walk is v_1, \dots, v_{n+1} , the estimator of wedge count is

$$\hat{C}_1^3 = \frac{1}{n} \sum_{t=1}^n \frac{|E|}{2} (d_{v_t} + d_{v_{t+1}} - 2|\mathcal{N}(v_t) \cap \mathcal{N}(v_{t+1})| - 2),$$

and the estimator of triangle count is

$$\hat{C}_2^3 = \frac{1}{n} \sum_{t=1}^n \frac{|E|}{3} |\mathcal{N}(v_t) \cap \mathcal{N}(v_{t+1})|.$$

4.4.2 Estimator for 4-Node Graphlets. To estimate the counts of 4-node graphlets, we consider each consecutive three steps of the random walk. Given the sequence of visited nodes v_1, \dots, v_{n+2} , we have

$$\hat{C}_i^4 = \frac{1}{n} \sum_{t=1}^n 2|E| \frac{d_{v_{t+1}}}{\beta_i^4} f_i^4((v_t, v_{t+1}, v_{t+2})).$$

The value of the function $f_i^4((v_t, v_{t+1}, v_{t+2}))$ is determined by $\mathcal{N}(v_t)$, $\mathcal{N}(v_{t+1})$, and $\mathcal{N}(v_{t+2})$. For example, if $\{v_t, v_{t+1}, v_{t+2}\}$ induces a triangle, then $f_6^4((v_t, v_{t+1}, v_{t+2})) = |\mathcal{N}(v_t) \cap \mathcal{N}(v_{t+1}) \cap \mathcal{N}(v_{t+2})|$. Similar to 3-node graphlets, set intersection is the key operation to compute f_i^4 .

4.4.3 Estimator for 5-Node Graphlets. All 3-, 4-, 5-node graphlet counts can be estimated with Equation (3) except the 5-node star graphlets ($\mathcal{G}_3^5, \mathfrak{X}$) due to $\beta_3^5 = 0$ ($\beta_3^5 = 0$ is because all 4-node graphlets in \mathfrak{X} are \mathfrak{Z} , and we cannot walk through \mathfrak{Z} via SRW, i.e., there is no Hamilton path in \mathfrak{Z}). 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 \mathcal{G}_i^k . N_3^5 denotes the counts of subgraphs isomorphic to \mathcal{G}_3^5 (\mathfrak{X}). There is a simple linear relationship between C_i^k and N_i^k [23]. For example, we have the following:

$$\begin{aligned} N_3^5 &= \sum_{v \in V} \binom{d_v}{4} = \sum_{i=1}^{21} \phi_i^5 \cdot C_i^5 = C_3^5 + C_6^5 + C_9^5 + C_{10}^5 \\ &\quad + 2C_{14}^5 + C_{15}^5 + C_{16}^5 + 2C_{18}^5 + C_{19}^5 + 3C_{20}^5 + 5C_{21}^5, \end{aligned} \quad (8)$$

where ϕ_i^5 denotes the number of \mathcal{G}_3^5 contained in the graphlet \mathcal{G}_i^5 . Given the sequence of nodes v_1, \dots, v_{n+3} visited by the random walk, the unbiased estimator of N_3^5 is

$$\hat{N}_3^5 \triangleq \frac{1}{n+3} \sum_{t=1}^{n+3} 2|E| \binom{d_{v_t}}{4} / d_{v_t} \rightarrow N_3^5 \text{ a.s.}$$

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

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

Leverage the formula (8) and above equation, we have

$$\hat{C}_3^5 = \hat{N}_3^5 - \sum_{i \in \{1, \dots, 21\} \setminus \{3\}} \phi_i^5 \hat{C}_i^5 \rightarrow C_3^5 \text{ a.s.}$$

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

5 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 introduce the mixing time, which quantifies how fast the random walk approaches the stationary distribution.

Definition 5.1 (Mixing time). [38, 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.³

We start by analyzing the estimator in Equation (3), where $|E|$ is known. Define

$$M_i^k = \max_{X \in \mathcal{M}^{(t)}} w_i^k(X) f_i^k(X).$$

Let T be the mixing time that ensures the total variation distance between the distribution after T steps and the stationary distribution of the random walk is within $1/8$, i.e., $T \triangleq \tau(1/8)$. The initial distribution of the random walk is denoted by $\boldsymbol{\varphi}$, and we define $\|\boldsymbol{\varphi}\|_{\boldsymbol{\pi}} \triangleq \sum_{v \in V} \varphi^2(v) / \pi(v)$. The following lemma describes the bound on the sample size to guarantee the estimates are within $1 \pm \epsilon$ of the true value with probability at least $1 - \delta$.

LEMMA 5.1. *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 Equation (7) 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 at least $1 - \delta$, it requires that

$$n \geq B_2 \triangleq \zeta \max \left\{ \frac{M_i^k}{C_i^k}, \frac{2|E|}{|V|} \right\} \frac{\log(2\|\boldsymbol{\varphi}\|_{\boldsymbol{\pi}} / \delta)}{\epsilon^2} (9T).$$

³The total variation distance between two distribution \mathbf{d}_1 and \mathbf{d}_2 on a countable space \mathcal{S} is given by $\|\mathbf{d}_1 - \mathbf{d}_2\|_1 \triangleq \frac{1}{2} \sum_{x \in \mathcal{S}} |d_1(x) - d_2(x)|$.

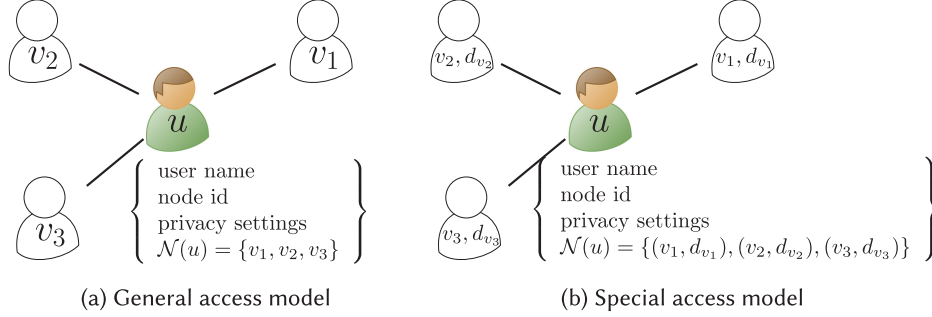


Fig. 4. Illustration of two different access models. (a) General access model. When querying a node u , we obtain a set of the user IDs for the neighbors of node u as well as some other attributes of node u . (b) Special access model. Some OSNs like Twitter and Weibo support the query that takes node u as input and return a set of friend objects of node u . The object usually contains the detailed information of a node, e.g., friend count (i.e., degree), location, and registration time.

This can be proved by finding the steps that guarantees both $\frac{1}{n} \sum_{t=1}^n \tilde{w}(X_t) f_i^k(X_t)$ and $\frac{1}{n} \sum_{t=1}^n \frac{1}{d_{v_t}}$ are within $(1 \pm \epsilon/3)$ of their expected value with probability at least $1 - \delta/2$. The fundamental concentration inequality we use in our proof is the Chernoff–Hoeffding bound for Markov chain [9], which is a concentration inequality based on the mixing time. The detailed proof on bound B_1 and B_2 can be found in Appendix A.2.

Remark. 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 $\Theta(\log^2 n)$ [1, 26, 38], which indicates good performance of our estimators in social networks. 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 $M_2^3/C_2^3 = (2|E| \cdot \max_{X \in \mathcal{M}^{(2)}} f_2^3(X))/C_2^3 = 2\Delta/(C_2^3/|E|)$, where Δ is the local maximum triangle count and $C_2^3/|E|$ is the average triangle count of each edge. The initial distribution contributes a little to the bound B_1 . According to the definition of $\|\varphi\|_{\pi}$, it is a good practice to start the random walk from nodes with high degree.

6 IMPROVED ESTIMATORS

In this section, two novel methods based on different access models are proposed to improve the efficiency of the estimators.⁴ The first method is based on the general access model we describe in Section 3.2. The second method is based on another important access model that can give degree information for friends of visited nodes. We call such access model as special access model. The difference of these two models are illustrated in Figure 4. The core idea of the improvement methods is to make better use of the degree information of observed nodes during the random walks.

6.1 Improvement for General Access Model

We first design an improved estimator for the general access model. 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

⁴We define an estimator with smaller variance as a more efficient estimator.

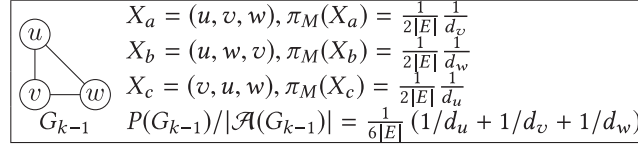


Fig. 5. Example of the observation ($k = 4$). $X_a, X_b, X_c \in \mathcal{A}(G_{k-1})$. If the degree d_u, d_v, d_w are unequal to each other, then the steady state probabilities $\pi_M(X_a), \pi_M(X_b), \pi_M(X_c)$ are different from each other.

sampling probability of the subgraph G_{k-1} instead of using the stationary distribution of the state X . Meanwhile, the sample space becomes the set of all $(k-1)$ -node connected induced subgraphs. 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 $\pi_M(X_a) \neq \pi_M(X_b)$.

Figure 5 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 probabilities 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 corresponding to the same subgraph 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.

– *Real-valued function:* 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.,

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

The reason we define $F_i^k(G_{k-1})$ as the summation of $f_i^k(X), \forall X \in \mathcal{A}(G_{k-1})$ is due to the fact that the summation of $F_i^k(G_{k-1})$ over all $(k-1)$ -node subgraphs in G is the same as the summation of $f_i^k(X)$ over all states in $\mathcal{M}^{(k-1)}$, i.e.,

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

– *Sampling probability of subgraph:* Following the definition of stationary distribution of the Markov chain, we define the nominal sampling probability of the subgraph $G_{k-1} = (V_{k-1}, E_{k-1})$ during the random walk process as

$$P(G_{k-1}) \triangleq \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n \mathbb{1}\{V(X_t) = V_{k-1}\} = \sum_{X_a \in \mathcal{A}(G_{k-1})} \pi_M(X_a).$$

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), here $\{X_t\}_{t=1}^n$ is a sequence of the states. According to the spirit of importance sampling, we have the following (not formally proved):

$$\frac{1}{n} \sum_{t=1}^n \frac{1}{\beta_i^k} \frac{F_i^k(G_{k-1}^t)}{P(G_{k-1}^t)} \rightarrow C_i^k \text{ a.s.} \quad (9)$$

Similar to the definition of $\mathcal{A}(G_{k-1})$ in Section 4.3, let $\mathcal{A}(X)$ denote the set of states in $\mathcal{M}^{(k-1)}$ that have the same node set as X . Rewrite the Equation (9) as follows:

$$\frac{1}{n} \sum_{t=1}^n \frac{1}{\beta_i^k} \frac{|\mathcal{A}(X_t)|}{\sum_{X_a \in \mathcal{A}(X_t)} \pi_M(X_a)} f_i^k(X_t) \rightarrow C_i^k \text{ a.s.}$$

Based on above discussion, we define a new re-weight function for the state X 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 following theorem is a formal description of Equation (9).

THEOREM 6.1. *The average of the function $W_i^k(X) f_i^k(X)$*

$$\hat{C}_i^k \triangleq \frac{1}{n} \sum_{t=1}^n W_i^k(X_t) f_i^k(X_t) \quad (10)$$

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

PROOF. Refer to Appendix A.3 for detailed proof. \square

6.1.1 Properties of Re-weight Function $W_i^k(X)$. The re-weight function $W_i^k(X)$ 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 the function $f_i^k(X)$ by $\sum_{X_a \in \mathcal{A}(X)} \pi_M(X_a) / |\mathcal{A}(X)| = P(G_{k-1}) / |\mathcal{A}(G_{k-1})|$ (i.e., the average of $\pi_M(X)$, $\forall X \in \mathcal{A}(G_{k-1})$) instead of $\pi_M(X)$ to remove the bias caused by unequal stationary probabilities.

6.1.2 Properties of Sampling Probability $P(G_{k-1})$. Recall that G_{k-1} denotes the subgraph induced by the nodes in the state X . We list the detailed comparison between $P(G_{k-1}) / |\mathcal{A}(G_{k-1})|$ and $\pi_M(X)$ when $k = 4, 5$ in the Table 9 in Appendix C. Note that $\pi_M(X)$ equals to $P(G_{k-1}) / |\mathcal{A}(G_{k-1})|$ when G_{k-1} is isomorphic to graphlet g_i^{k-1} with $\alpha_i^{k-1} = 2$. As a result, the improved estimator remains the same for the graphlet whose all the $(k-1)$ -node connected subgraphs are isomorphic to graphlets with $\alpha_j^{k-1} = 0$ or 2 (i.e., there is none or only one Hamilton path in the subgraph). For example, the $\pi_M(X)$ equals to $P(G_{k-1}) / |\mathcal{A}(G_{k-1})|$ when G_{k-1} is isomorphic to wedge $(\bullet-\bullet)$. Consequently the improvement method does not apply for the 4-node graphlets g_1^4 ($\bullet-\bullet-\bullet$), g_2^4 ($\bullet-\bullet-\bullet$), and g_3^4 ($\bullet-\bullet-\bullet$), since all the 3-node connected subgraphs in these graphlets are isomorphic to $\bullet-\bullet$.

6.1.3 Properties of the Improved Estimator. The intuition behind the improved estimator is that we combine the states corresponding to the same subgraph together and make better use of degree information of nodes in the subgraph. Similar to the definition $M_i^k \triangleq \max_{X \in \mathcal{M}^{(k-1)}} w_i^k(X) f_i^k(X)$ in Section 5, we define $O_i^k \triangleq \max_{X \in \mathcal{M}^{(k-1)}} W_i^k(X) f_i^k(X)$. With almost identical proof of bound B_1 and B_2 , we conclude that to guarantee the estimation is within $(1 \pm \epsilon) C_i^k$ with probability at least $1 - \delta$, the improved estimator requires the sample size at least $B_3 \triangleq \zeta \frac{O_i^k}{C_i^k} \frac{\log(\|\boldsymbol{\varphi}\|_{\pi}/\delta)}{\epsilon^2} T$ and $B_4 \triangleq \zeta \max\left\{\frac{O_i^k}{C_i^k}, \frac{2|E|}{|V|}\right\} \frac{\log(2\|\boldsymbol{\varphi}\|_{\pi}/\delta)}{\epsilon^2} (9T)$ for the situation where $|E|$ is known and $|E|$ is unknown, respectively. Define $X^* \triangleq \operatorname{argmax}_{X \in \mathcal{M}^{(k-1)}} w_i^k(X) f_i^k(X)$. Recall that all state $X \in \mathcal{A}(X^*)$ has $f_i^k(X) = f_i^k(X^*)$. Hence, we have $w_i^k(X^*) = \max_{X \in \mathcal{A}(X^*)} w_i^k(X)$ (i.e., $\pi_M(X^*) = \min_{X \in \mathcal{A}(X^*)} \pi_M(X)$), otherwise we can find some state $X \in \mathcal{A}(X^*)$ such that $w_i^k(X) f_i^k(X)$ is larger. Based on above discussion, we have $W_i^k(X^*) / w_i^k(X^*) = \frac{|\mathcal{A}(X^*)| \cdot \min_{X \in \mathcal{A}(X^*)} \pi_M(X)}{\sum_{X \in \mathcal{A}(X^*)} \pi_M(X)} \leq 1$, which indicates $O_i^k \leq M_i^k$. So the bound of the sample size for the improved estimator is smaller. Besides, we have $\operatorname{Var}_{\pi_M}[W_i^k(X)$

$f_i^k(X)] \leq \text{Var}_{\pi_M}[w_i^k(X)f_i^k(X)]$ (proof can be found in Lemma A.4 in Appendix A.3). Hence, the variance of the new estimator has the potential to be smaller, i.e., the new estimator is more efficient. In general, we do not have a deterministic conclusion that given the same samples, the improved estimator has higher accuracy. We leave it as an open question. Some preliminary discussion is already provided in Appendix A.3.

6.2 Improvement for the Special Access Model

Now, we design an improved estimator for the special access model, which is illustrated in Figure 4(b). The method in this subsection assumes the APIs provided by the OSNs support the function to return a set of neighbors' IDs as well as the degree information of these neighbors when querying a node u . Several OSNs support this kind of APIs, e.g., the GET friends/list⁵ and GET followers/list⁶ of Twitter, GET friendships/friends,⁷ and GET friendships/followers⁸ of Weibo. To leverage the degree information of neighbors, the *inclusion probability* is computed for each subgraph *observed* by current visited state.

Recall that a subgraph can be observed by the state X if and only if the subgraph contains all the nodes in X and one node in the neighborhood of X . The observed subgraphs of state X are formally defined in Section 4.2 as $\mathcal{S}(X) \triangleq \{G_k(V_k) | V_k = V(X) \cup \{v\}, v \in \mathcal{N}(V(X))\}$. The main idea of the improvement method is to define an *indicator function* and the *inclusion probability* for each subgraph $G_k \in \mathcal{S}(X)$.

– *Indicator function*: We define an indicator function for the k -node subgraph G_k as

$$h_i^k(G_k) = \mathbb{1}\{G_k \text{ is isomorphic to } g_i^k\}.$$

It is trivial to verify that $\sum_{X \in \mathcal{M}^{(k-1)}} (\sum_{G_k \in \mathcal{S}(X)} h_i^k(X)) = \sum_{X \in \mathcal{M}^{(k-1)}} f_i^k(X) = \beta_i^k C_i^k$.

– *Inclusion probability*: All the states in $\mathcal{B}(G_k)$ can observe the subgraph G_k . The *inclusion probability*

$$\mathcal{P}(G_k) \triangleq \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n \mathbb{1}\{X_t \in \mathcal{B}(G_k)\} = \sum_{X \in \mathcal{B}(G_k)} \pi_M(X)$$

is defined to measure the probability that G_k is observed during the random walk.

Using the principle of the importance sampling, we have the following theorem.

THEOREM 6.2. *The average of the function $\sum_{G_k \in \mathcal{S}(X)} h_i^k(G_k)/\mathcal{P}(G_k)$*

$$\hat{C}_i^k = \frac{1}{n} \sum_{t=1}^n \left(\sum_{G_k \in \mathcal{S}(X_t)} \frac{h_i^k(G_k)}{\mathcal{P}(G_k)} \right) \quad (11)$$

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

PROOF. The detailed proof is present in Appendix A.4. □

Example. Table 4 shows an example of computing $\mathcal{P}(G_k)$, where G_k is one of the k -node subgraphs in $\mathcal{S}(X)$ presented in Figure 3. Refer to Figure 3 for detailed illustration of X and $\mathcal{S}(X)$. Computing $\mathcal{P}(G_k)$ in the example leverages the degrees of visited nodes 2, 3, 4, and the degree of the neighborhood node 1. Meanwhile, the nominal probability $\beta_i^k \pi_M(X)$ in Equation (3) only

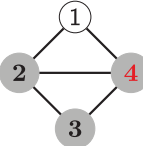
⁵<https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/api-reference/get-friends-list>.

⁶<https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/api-reference/get-followers-list>.

⁷<http://open.weibo.com/wiki/2/friendships/friends>.

⁸<http://open.weibo.com/wiki/2/friendships/followers>.

Table 4. Example of the Improved Estimator for Special Access Model ($k = 4$)

G_k	$\mathcal{B}(G_k) = \bigcup_{G_{k-1} \in \mathcal{H}_{k-1}} \mathcal{A}(G_{k-1})$				$\beta_i^k \pi_M(X)$	$\beta_i^k \frac{\sum_{X_a \in \mathcal{A}(X)} \pi_M(X_a)}{ \mathcal{A}(X) }$	$\mathcal{P}(G_k)$
	$G_{k-1} \in \mathcal{H}_{k-1}$				$\frac{16}{2 E } \frac{1}{d_3}$	$\frac{16}{6 E } \left(\frac{1}{d_2} + \frac{1}{d_3} + \frac{1}{d_4} \right)$	$\frac{1}{2 E } \left(\frac{2}{d_1} + \frac{6}{d_2} + \frac{2}{d_3} + \frac{6}{d_4} \right)$
	$\mathcal{A}(G_{k-1})$						
$X = (2, 3, 4)$	(1, 2, 3) (3, 2, 1)	(1, 4, 3) (3, 4, 1)	(2, 1, 4) (4, 1, 2) (1, 4, 2) (2, 4, 1)	(2, 3, 4) (4, 3, 2) (3, 2, 4) (4, 2, 3) (2, 4, 3) (3, 4, 2)	Eq. (3)	Eq.(10)	Eq. (11)
$k = 4$							

uses the degree of node 3 and the sampling probability in Equation (10) only uses the degrees of node 2, 3, 4. We conclude that the improved estimator in Equation (11) removes the bias of each subgraph G_k in $S(X)$ with more degree information.

Remark. When the number of edges is unknown, we just need to replace the exact $|E|$ in Equation (10) and Equation (11) with the estimated $|E|$ in Equation (6) to get unbiased estimates of graphlet counts.

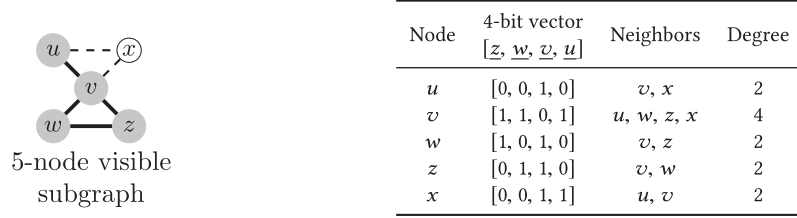
7 IMPLEMENTATION DETAILS

In this section, we discuss the detailed implementation of our estimators. A straightforward implementation is to iterate through each node $v \in \mathcal{N}(V(X))$ and determine the graphlet type of the subgraph induced by $V(X) \cup \{v\}$. However, for the two estimators designed for the general access model, we have more efficient implementation to compute the most time-consuming operation in both estimators, i.e., the computation of $f_i^k(X)$. In the following, we present the more efficient implementation that can leverage previous computation results and reduce the running time.

7.1 Representing Visible Subgraphs

First, we discuss how to represent the k -node visible subgraphs. Assume, we visit a $(k-1)$ -node touched subgraph induced by the node set $V_{k-1} = \{v_1, \dots, v_{k-1}\}$. We allocate $k-1$ bits for each node $u \in \bigcup_{v \in V_{k-1}} \mathcal{N}(v)$. The least significant bit (LSB) to the most significant bit (MSB) of the bit vector indicate whether the node u is adjacent to nodes $\{v_1, \dots, v_{k-1}\}$, respectively. These bit vectors are the compressed adjacent matrix of the k -node visible subgraphs. The degrees of nodes in the visible subgraph can be viewed as *degree signature* of the subgraph, which can be used to determine the graphlet types of the subgraphs [6].

Example. Figure 6(a) shows an example on representing 5-node visible subgraph induced by the node set $\{u, v, w, z, x\}$. The subgraph induced by $\{u, v, w, z\}$ is a touched subgraph. The MSB to the LSB of the 4-bit vectors indicates whether the node is adjacent to nodes z, w, v, u , respectively. The 4-bit vectors of u, v, w, z reveal the adjacent relationship between them. Besides, the 4-bit vector of node x is $[0, 0, 1, 1]$, which means node x is adjacent to node u and v . Hence, we know all the edges between nodes u, v, w, z, x with the 4-bit vectors. The degree signature of the visible



(a) Example on representing 5-node visible subgraphs.

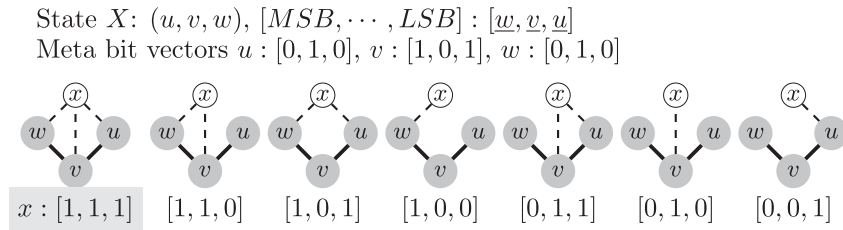
(b) Example of computing $f_i^k(X)$.

Fig. 6. Illustration of implementation details.

subgraph induced by $\{u, v, w, z, x\}$ is $\{2, 4, 2, 2, 2\}$, which is sufficient for us to know that the subgraph is isomorphic to $\mathfrak{X}(\mathcal{G}_{10}^5)$.

7.2 Computation of $f_i^k(X)$

The bit vectors for nodes in V_{k-1} are the “meta bit vectors,” which can be used to determine the graphlet type of the subgraph induced by V_{k-1} . Each $(k-1)$ -bit vector combined with the “meta bit vectors” corresponds to a k -node graphlet type. For each $(k-1)$ -bit vector B , we maintain the counts of nodes in $\mathcal{N}(V_{k-1})$ which have bit vectors B . These counts are used to compute $f_i^k(X)$.

Example. Figure 6(b) gives an example on computing $f_i^k(X)$ with the counts of different bit vectors B . The MSB to the LSB of the bit vectors indicate whether the node is adjacent to $\{w, v, u\}$, respectively. The bit vector of x combined with the meta bit vectors of w, v, u is sufficient to determine the graphlet type of subgraph induced by $\{x, w, v, u\}$. For example, if the bit vector of node x is $[1, 1, 1]$, which means x is adjacent to w, v , and u , the subgraph induced by $\{x, w, v, u\}$ has degree signature $\{3, 2, 3, 2\}$, i.e., the subgraph is isomorphic to graphlet $\mathfrak{N}(\mathcal{G}_5^4)$. If there are N_1 nodes in $\mathcal{N}(\{u, v, w\})$ having the bit vector $B = [1, 1, 1]$, then we have $f_5^4(X) = N_1$. If the bit vector of node x is $[1, 1, 0]$ or $[0, 1, 1]$, the subgraph induced by $\{x, w, v, u\}$ is isomorphic to graphlet $\mathfrak{N}(\mathcal{G}_4^4)$. Consequently, if there are N_2 nodes in $\mathcal{N}(\{u, v, w\})$ have the bit vector $B = [1, 1, 0]$ or $B = [0, 1, 1]$, then $f_4^4(X) = N_2$.

7.3 Update the Bit Vectors

Assume our current visited state is $X_1 = (v_1, \dots, v_{k-1})$. When the random walker proceeds to v_k (a neighbor of v_{k-1}), we transit to the next state $X_2 = (v_2, \dots, v_k)$. We need to update the bit vectors for the computation of $f_i^k(X)$. The naive implementation is to clear all previous bit vectors and iterate through the neighborhood of the new touched subgraph. However, it is sufficient to simply scan $\mathcal{N}(v_1)$ and $\mathcal{N}(v_k)$. The idea is to clear the position that represents the adjacent relationship

with node v_1 for the bit vectors of nodes in $\mathcal{N}(v_1)$. Then, set the same position in the bit vectors for all nodes in $\mathcal{N}(v_k)$. We maintain the counts of each $(k-1)$ -bit vector B dynamically when we are updating the bit vectors.

Example. Let $k = 4$. Assume node $u \in \mathcal{N}(v_1)$ has bit vector $[0, 1, 1]$ and the LSB indicates node u is adjacent to v_1 . During the clear phase, the bit vector of u becomes $[0, 1, 0]$. We decrease the count of $[0, 1, 1]$ by 1 and increase count of $[0, 1, 0]$ by 1. If node $w \in \mathcal{N}(v_4)$ has bit vector $[1, 1, 0]$ after the clear phase (i.e., node w is adjacent to v_2 and v_3), then during the phase of setting the LSB for bit vectors of nodes in $\mathcal{N}(v_4)$, the bit vector for w becomes $[1, 1, 1]$. We increase count of $[1, 1, 1]$ by 1 and decrease count of $[1, 1, 0]$ by 1.

Remark. When $k \geq 5$, the degree signatures are not enough to determine the graphlet types. To determine the graphlet types efficiently, we adopt a method in [20], whose main idea is creating a look-up table to determine the graphlet types in constant time. To create the look-up table, we represent the graphlets with the lower-triangle of their adjacent matrix, and then place the lower-triangular matrix into a bit vector. The look-up table maps the bit vectors to their corresponding graphlet types. For each graphlet, the look-up table contains all possible bit vectors for such graphlet. Hence, we can determine the graphlet type in $O(k^2)$ time. However, the space complexity of the look-up table grows exponentially as k increases. According to the experimental results in [20], it is practical to create the look-up table when $k \leq 8$.

7.4 Running Time Analysis

The expected size of $\mathcal{N}(V(X))$ is $L_1 \triangleq \sum_{v \in V} (k-1)d_v^2 / (2|E|)^9$. Hence, the expected running time of the naive implementation for computing $f_i^k(X)$ is $\Theta(L_1)$. However, the better implementation can reduce the running time to $\Theta(L_2)$, here $L_2 = \sum_{v \in V} d_v^2 / |E|$. Note that L_2 does not depend on the size of the graphlets, which is especially beneficial for extending our algorithm framework to graphlets with larger size.

7.5 Discussion

Theoretically, our algorithm can be extended to count any k -node graphlets. However, generally speaking, there exists *no* efficient method to compute the number of all k -node graphlets when k is sufficiently large due to the *combinatorial explosion*. We give detailed reasons in the following.

- First, number of all possible k -node graphlets (denoted as $F(k)$) grow exponentially as k increases [19, 49]. The computation of $F(k)$ is described in [19]. We list the sequence of $F(k)$ for $k \leq 19$ in Table 5. As shown in Table 5, the number of distinct 19-node graphlets is $\approx 2.5 \times 10^{34}$, which is extremely large even though $k = 19$ is relatively a small number. It is difficult and impractical to compute and store all k -node graphlet counts since $F(k)$ grows exponentially as k increases.
- Second, we cannot avoid the isomorphism checking when counting the graphlets. To simplify the problem, assume we only compute one specific graphlet H here. For the exact counting, the state-of-the-art algorithm has time complexity $k^{O(k)} \cdot n^{0.174k + o(k)}$ to compute the number of graphlet H with k edges in a graph with n nodes [10]. For the sampling algorithm, we need to check whether the sample is isomorphic to H , which has the time complexity $k^2 k!$ in worst case. In general, the computation of a specific graphlet has exponential time complexity as k increases.

⁹If we visit node v at the time t during the random walk, we need to include $\mathcal{N}(v)$ at time $t, \dots, t+k-2$, i.e., $\mathcal{N}(v)$ is read $k-1$ times.

Table 5. Number of Distinct k -Node Graphlets [49]

k	Number of distinct k -node graphlets
1	1
2	1
3	2
4	6
5	21
6	112
7	853
8	11117
9	261080
10	11716571
11	1006700565
12	164059830476
13	50335907869219
14	29003487462848061
15	31397381142761241960
16	63969560113225176176277
17	245871831682084026519528568
18	1787331725248899088890200576580
19	24636021429399867655322650759681644

– Third, the percentage of some k -node graphlets (e.g., cliques and cycles) among all k -node graphlets decreases quickly as k increases [16, 22], which makes the needed sample size increases quickly for the sampling methods. In the following, we explain the statement in details. Let μ_i^k denote the percentage of the k -node graphlet g_i^k among all k -node graphlets in the graph G , i.e., $\mu_i^k = C_i^k / C^k$. We first discuss the percentage of some special graphlets. Take the percentage of k -node cliques as an example. The 3-, 4-, 5-node cliques in the graph Epinion in our datasets take 2.29×10^{-2} , 2.25×10^{-4} , 1.47×10^{-6} percentage, respectively. We can see that μ_{clique}^k decreases quickly as k increases. More empirical and theoretical analysis on the clique counts can be found in [16, 22]. Then, we discuss the relationship between sample size and μ_i^k . Suppose each graphlet sample is sampled uniformly at random from the set of all k -node graphlets in the graph G . Then, according to the Chernof–Hoeffding bound, to guarantee the estimate of C_i^k within $(1 \pm \epsilon)C_i^k$ with probability at least δ , we need at least $\frac{3}{\epsilon^2 \mu_i^k} \ln \frac{2}{\delta}$ samples, i.e., to guarantee the estimation accuracy, the needed sample size grows almost linearly with $1/\mu_i^k$. Even though smart importance sampling methods can be designed to estimate count of the special graphlets [22], the needed sample size still has the same trend.

In practice, we usually choose small k , say $k \leq 5$. On one hand, the computation cost grows dramatically as k increases, for both of exact counting methods and sampling methods. On the other hand, it is easier to interpret the physical meaning of the k -node graphlets when k is small. For example, the number of triangles can be used to measure the homophily and transitivity of the OSNs. Besides, many applications, such as graph classification [48] based on k -node graphlets counts are proposed when $k \leq 5$.

Table 6. Summary of the Datasets

Name	Nodes	Edges	Description
Epinion [32]	76K	406K	Trust network from the online social network Epinion.
Slashdot [32]	77K	469K	Friend/foe links between the users of Slashdot social network.
Facebook [29]	63K	817K	A small subset of the total Facebook friendship graph.
Pokec [29]	1.6M	22.3M	Friendship network from the Slovak social network Pokec.
Flickr [29]	2.2M	22.7M	Social network of Flickr users and their friendship connections.
Orkut [44]	3.0M	106M	Social networks of Orkut users and their connections.
Twitter [44]	21.3M	265M	Graph about who follows whom on social media Twitter.
Weibo [44]	58.7M	261M	Graph of a micro-blogging service with millions of users in China.

8 EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of our proposed algorithm. We aim to answer the following questions.

- How accurate is our method generally?
- Do the improved estimators improve the accuracy?
- Does our method outperform the state-of-the-art methods?

8.1 Experimental Setup

We test the performance of our proposed algorithm on various social networks. Table 6 lists the datasets used in our experiments. For all the datasets, we remove the directions, self-loops, and multi-edges, which can be easily avoided during the random walks. We report the number of nodes and edges in the 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 [4]. For 5-node graphlets, we obtain the ground truth with the method in [21]. Figure 8(a) and (c) shows the 4-, 5-node graphlet counts for all the graphs whose ground-truth can be obtained. 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 at <https://tinyurl.com/GraphletCount-Journal>.

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 1,000 independent simulations.
- *Confidence bound:* we construct a [5%, 95%]-confidence interval for the estimate z , which is defined as the interval [LB, 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 1,000 times, and use the 5th and 95th 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 1,000 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 $\sqrt{\text{Var}[\hat{C}_i^k]}/C_i^k$.

Names of estimators: We denote the basic estimator in Equation (3) as BASIC. The improved estimator in Equation (10) for the general access model is referred as IMPRG, while the estimator in Equation (11) for the special access model is referred as IMPRS. If the number of edges is unknown, we need to replace the exact $|E|$ in the estimators with the estimated $|E|$. Correspondingly, we append “-U” at the end of the estimators’ name. For example, IMPRG-U represents the estimator (10) with exact $|E|$ replaced by estimated $|E|$.

8.2 Performance Analysis

Accuracy: We demonstrate the accuracy of our proposed estimator IMPRG in Table 7. Here, we choose IMPRG for presentation because it is applicable to the general access model and has better performance than BASIC. Note that for 3-node graphlets, the estimator BASIC and the estimator IMPRG are the same. We assume the exact number of edges is known. Only the accuracy for graphlets $\mathcal{G}_2^3, \mathcal{G}_3^4, \mathcal{G}_5^4, \mathcal{G}_6^4, \mathcal{G}_{17}^5, \mathcal{G}_{19}^5, \mathcal{G}_{20}^5, \mathcal{G}_{21}^5$ is reported since their counts are the *smallest* among 3-, 4-, 5-node graphlets, respectively, and they were observed to have *lower* accuracy. The extremely high computation cost of the exact enumeration algorithms makes it difficult to obtain the 5-node graphlets counts for all the graphs. Hence, we only show the results of 5-node graphlets for the graphs whose ground truth can be obtained with reasonable running time. The sample size equals to 20K. The findings are summarized as follows.

- *Our estimator is unbiased*: The 4th column of the table shows error of the average estimate over 1,000 independent runs, which measures the unbiasedness of the estimators. The error is below 0.73% for all the reported graphlets except the 5-node clique of Epinion and Slashdot. The results verify our claims in Theorems 4.1 and 6.1.
- *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, 3–12% for \mathcal{G}_5^4 and \mathcal{G}_6^4 except Sinaweibo, and 6.6–37% for the 5-node graphlets. These results are enough for many applications, e.g., the computation of graph kernel [48].
- *Our estimator has small variance*: Our estimator is asymptotic 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 1.8–18% except the 4-node clique in Sinaweibo. 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 [17]. Besides, given the sample size, the accuracy does not degenerate with

Table 7. Accuracy of the Proposed Estimator **IMPRG** When the Sample Size Equals to 20K, i.e., We Perform the Random Walk for 20K Steps

3-node triangle (\mathcal{G}_2^3) \blacktriangle								4-node cycle (\mathcal{G}_3^4) \blacksquare							
	C_2^3	$E[\hat{C}_2^3]$	$\frac{ E[\hat{C}_2^3]-C_2^3 }{C_2^3}$	LB	UB	MRE	NRMSE		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	1.6M	1.6M	0.0002	1.6M	1.7M	0.015	0.019	Epinion	71.5M	71.5M	0.0005	67.6M	75.5M	0.027	0.034
Slashdot	552K	552K	0.0002	520K	586K	0.030	0.037	Slashdot	27.2M	27.2M	0.0002	25.5M	28.9M	0.030	0.039
Facebook	3.5M	3.5M	0.0000	3.4M	3.6M	0.016	0.020	Facebook	35.8M	35.8M	0.0005	33.2M	38.4M	0.035	0.044
Pokec	32.6M	32.6M	0.0003	31.5M	33.7M	0.017	0.021	Pokec	358M	359M	0.0022	331M	389M	0.039	0.049
Flickr	838M	836M	0.0016	767M	913M	0.042	0.053	Flickr	219B	218B	0.0045	198B	238B	0.045	0.056
Orkut	525M	525M	0.0002	504M	546M	0.019	0.024	Orkut	51.1B	50.9B	0.0031	40.1B	62.9B	0.111	0.140
Twitter	17.3B	17.3B	0.0001	16.4B	18.3B	0.027	0.034	Twitter	16.8T	16.8T	0.0003	15.5T	18.3T	0.040	0.050
Weibo	213M	213M	0.0002	193M	234M	0.045	0.057	Weibo	259B	259B	0.0016	235B	287B	0.049	0.062
4-node chordal-cycle (\mathcal{G}_5^4) \blacklozenge								4-node clique (\mathcal{G}_6^4) \blackstar							
	C_5^4	$E[\hat{C}_5^4]$	$\frac{ E[\hat{C}_5^4]-C_5^4 }{C_5^4}$	LB	UB	MRE	NRMSE		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	77.7M	77.6M	0.0016	71.6M	84.1M	0.038	0.048	Epinion	5.8M	5.8M	0.0019	5.1M	6.5M	0.056	0.072
Slashdot	16.8M	16.9M	0.0050	14.7M	19.2M	0.067	0.083	Slashdot	2.0M	2.0M	0.0060	1.6M	2.5M	0.115	0.142
Facebook	76.0M	75.9M	0.0004	70.9M	81.4M	0.033	0.042	Facebook	13.3M	13.3M	0.0008	12.2M	14.4M	0.039	0.049
Pokec	467M	470M	0.0073	398M	607M	0.105	0.155	Pokec	42.9M	42.8M	0.0036	37.6M	48.3M	0.060	0.076
Flickr	373B	372B	0.0036	328B	417B	0.058	0.073	Flickr	40.2B	40.1B	0.0017	34.2B	46.4B	0.075	0.094
Orkut	33.3B	33.1B	0.0055	25.9B	41.9B	0.115	0.150	Orkut	2.4B	2.4B	0.0043	2.1B	2.8B	0.074	0.123
Twitter	28.4T	28.4T	0.0011	25.3T	31.7T	0.055	0.068	Twitter	2.1T	2.1T	0.0017	1.8T	2.5T	0.077	0.097
Weibo	27.2B	27.2B	0.0022	20.7B	35.9B	0.135	0.182	Weibo	663M	664M	0.0016	344M	1.2B	0.300	0.405
5-node semi-center-square (\mathcal{G}_{17}^5) \blacklozenge								5-node semi-clique (\mathcal{G}_{20}^5) \blackstar							
	C_{17}^5	$E[\hat{C}_{17}^5]$	$\frac{ E[\hat{C}_{17}^5]-C_{17}^5 }{C_{17}^5}$	LB	UB	MRE	NRMSE		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	854M	856M	0.0023	733M	985M	0.074	0.092	Epinion	158M	158M	0.0027	112M	207M	0.148	0.186
Slashdot	100M	100M	0.0013	74.3M	129M	0.138	0.191	Slashdot	54.8M	55.2M	0.0071	36.0M	77.2M	0.185	0.235
Facebook	283M	283M	0.0013	245M	324M	0.066	0.083	Facebook	198M	198M	0.0031	171M	227M	0.068	0.087
5-node center-square (\mathcal{G}_{19}^5) \blacklozenge								5-node clique (\mathcal{G}_{21}^5) \blackstar							
	C_{19}^5	$E[\hat{C}_{19}^5]$	$\frac{ E[\hat{C}_{19}^5]-C_{19}^5 }{C_{19}^5}$	LB	UB	MRE	NRMSE		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	206M	206M	0.0027	165M	253M	0.103	0.131	Epinion	17.4M	17.2M	0.0102	9.9M	25.8M	0.231	0.291
Slashdot	31.1M	30.9M	0.0067	21.5M	41.9M	0.165	0.208	Slashdot	10.7M	10.8M	0.0123	5.1M	18.0M	0.293	0.373
Facebook	107M	106M	0.0014	92.6M	123M	0.068	0.085	Facebook	46.5M	46.7M	0.0040	38.0M	56.8M	0.097	0.122

Here, we only show the results of 5-node graphlets for the graphs whose ground truth can be obtained with reasonable running time.

the increase of the graph sizes, e.g., Twitter have slightly smaller MRE and NRMSE for the triangle estimate than Slashdot given 20K sampled nodes. However, the number of nodes in Twitter is 277 times of that in Slashdot.

Benefit of the improved estimators: We show the gain of the improved estimators (IMPRG and IMPRS) in Figure 7. For fair comparison, we use the same set of 20K samples and then apply the basic and the improved estimators separately. We choose the MRE as the accuracy measure. We also show the performance of the corresponding estimators when the number of edges is unknown. Note that the basic and the improved estimators are the same for 3-node graphlets. For 4-node graphlet, the IMPRG only changes the estimates when the subgraph contains triangle, i.e., $\mathcal{G}_4^4, \mathcal{G}_5^4, \mathcal{G}_6^4$. From Figure 7, we can observe the following:

- The improved estimators reduces the error for *all the graphs* and *all the graphlet types* presented in the figure. For 4-node graphlets, the improved estimator IMPRG reduces MRE by 0.001–0.18, while for 5-node graphlets, it reduces MRE by 0.027–0.044. The improved estimator IMPRS reduces the MRE of 4-node graphlet estimation by 0.20 at most and reduce MRE of 5-node graphlet estimation by 0.01–0.059.

Comparison between the improved estimators: Figure 8(a) and (c) shows when MRE_{IMPRS} (the MRE of IMPRS) is smaller, larger, and equal to MRE_{IMPRG} (the MRE of IMPRG) for all the graphs in the datasets whose ground-truth can be obtained. Figure 8(b) and (d) presents the detailed comparison

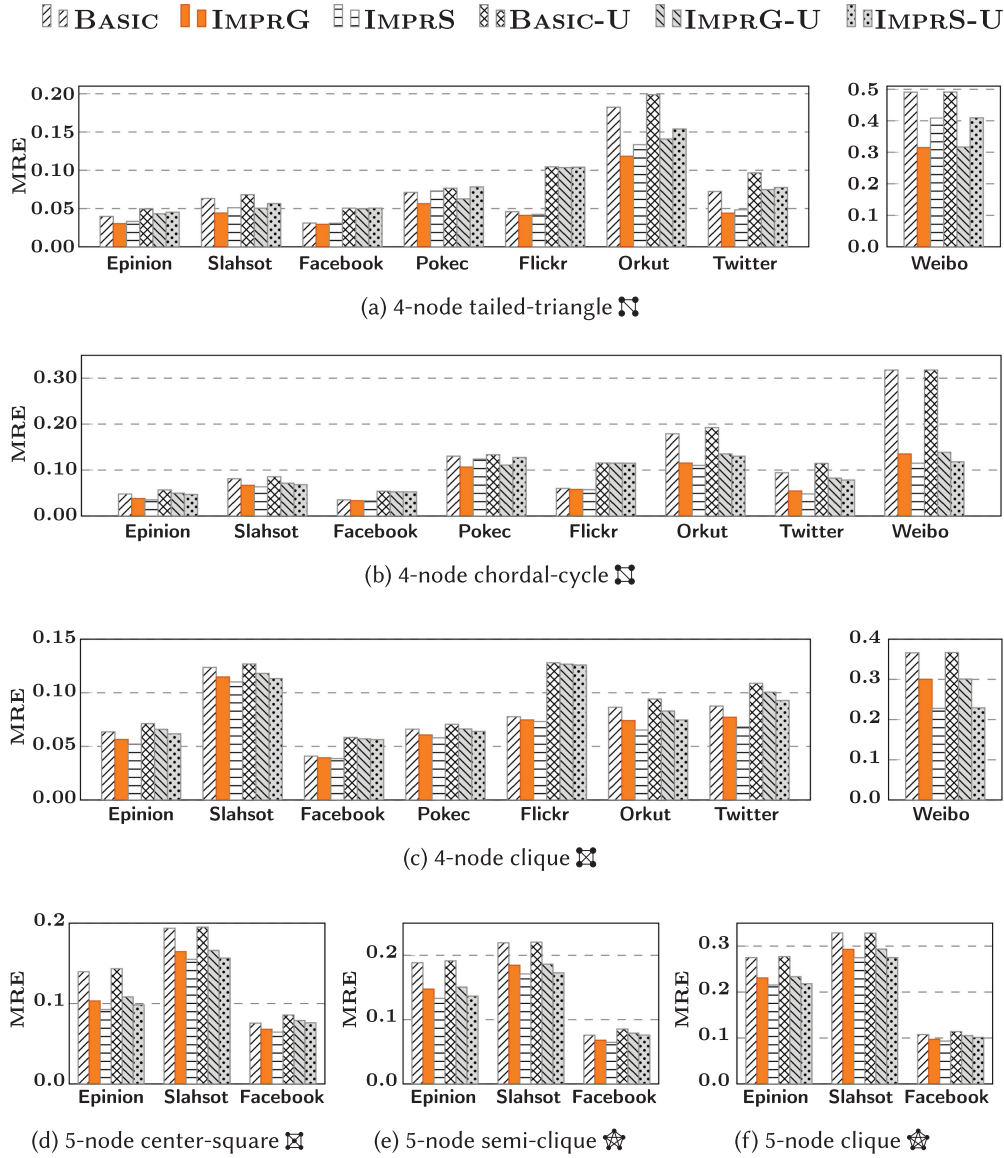


Fig. 7. Compare the accuracy of different estimators. The sample size is 20K.

between IMPRG and IMPRS for graphs Weibo and Slahsot, respectively. We choose Weibo and Slahsot since they have the largest number of nodes among the graphs whose 4- and 5-node graphlet counts can be obtained. For each graph, we apply different estimators to the same set of samples. Note that IMPRG and IMPRS are the same for the graphlets \mathcal{G}_4^1 and \mathcal{G}_4^2 theoretically, which is also validated in Figure 8(a) and (c). We denote the graphlets that contains nodes of degree one as *tailed graphlets*. All the 4-, 5-node tailed graphlets are $\mathcal{G}_4^1, \mathcal{G}_4^2, \mathcal{G}_4^3, \mathcal{G}_4^4, \mathcal{G}_4^5, \mathcal{G}_4^6, \mathcal{G}_4^7, \mathcal{G}_4^8, \mathcal{G}_4^9, \mathcal{G}_4^{10}, \mathcal{G}_4^{11}, \mathcal{G}_4^{12}, \mathcal{G}_4^{13}, \mathcal{G}_4^{14}, \mathcal{G}_4^{15}, \mathcal{G}_4^{16}, \mathcal{G}_4^{17}, \mathcal{G}_4^{18}, \mathcal{G}_4^{19}, \mathcal{G}_4^{20}, \mathcal{G}_4^{21}, \mathcal{G}_4^{22}, \mathcal{G}_4^{23}, \mathcal{G}_4^{24}, \mathcal{G}_4^{25}, \mathcal{G}_4^{26}, \mathcal{G}_4^{27}, \mathcal{G}_4^{28}, \mathcal{G}_4^{29}, \mathcal{G}_4^{30}, \mathcal{G}_4^{31}, \mathcal{G}_4^{32}, \mathcal{G}_4^{33}, \mathcal{G}_4^{34}, \mathcal{G}_4^{35}, \mathcal{G}_4^{36}, \mathcal{G}_4^{37}, \mathcal{G}_4^{38}, \mathcal{G}_4^{39}, \mathcal{G}_4^{40}, \mathcal{G}_4^{41}, \mathcal{G}_4^{42}, \mathcal{G}_4^{43}, \mathcal{G}_4^{44}, \mathcal{G}_4^{45}, \mathcal{G}_4^{46}, \mathcal{G}_4^{47}, \mathcal{G}_4^{48}, \mathcal{G}_4^{49}, \mathcal{G}_4^{50}, \mathcal{G}_4^{51}, \mathcal{G}_4^{52}, \mathcal{G}_4^{53}, \mathcal{G}_4^{54}, \mathcal{G}_4^{55}, \mathcal{G}_4^{56}, \mathcal{G}_4^{57}, \mathcal{G}_4^{58}, \mathcal{G}_4^{59}, \mathcal{G}_4^{60}, \mathcal{G}_4^{61}, \mathcal{G}_4^{62}, \mathcal{G}_4^{63}, \mathcal{G}_4^{64}, \mathcal{G}_4^{65}, \mathcal{G}_4^{66}, \mathcal{G}_4^{67}, \mathcal{G}_4^{68}, \mathcal{G}_4^{69}, \mathcal{G}_4^{70}, \mathcal{G}_4^{71}, \mathcal{G}_4^{72}, \mathcal{G}_4^{73}, \mathcal{G}_4^{74}, \mathcal{G}_4^{75}, \mathcal{G}_4^{76}, \mathcal{G}_4^{77}, \mathcal{G}_4^{78}, \mathcal{G}_4^{79}, \mathcal{G}_4^{80}, \mathcal{G}_4^{81}, \mathcal{G}_4^{82}, \mathcal{G}_4^{83}, \mathcal{G}_4^{84}, \mathcal{G}_4^{85}, \mathcal{G}_4^{86}, \mathcal{G}_4^{87}, \mathcal{G}_4^{88}, \mathcal{G}_4^{89}, \mathcal{G}_4^{90}, \mathcal{G}_4^{91}, \mathcal{G}_4^{92}, \mathcal{G}_4^{93}, \mathcal{G}_4^{94}, \mathcal{G}_4^{95}, \mathcal{G}_4^{96}, \mathcal{G}_4^{97}, \mathcal{G}_4^{98}, \mathcal{G}_4^{99}, \mathcal{G}_4^{100}$. Our observations are summarized as follows.

- For the tailed graphlets, IMPRS is no better than IMPRG. For example, the MRE of IMPRG is 0.315 for 4-node tailed-triangle in Weibo, while the MRE of IMPRS is 0.409.

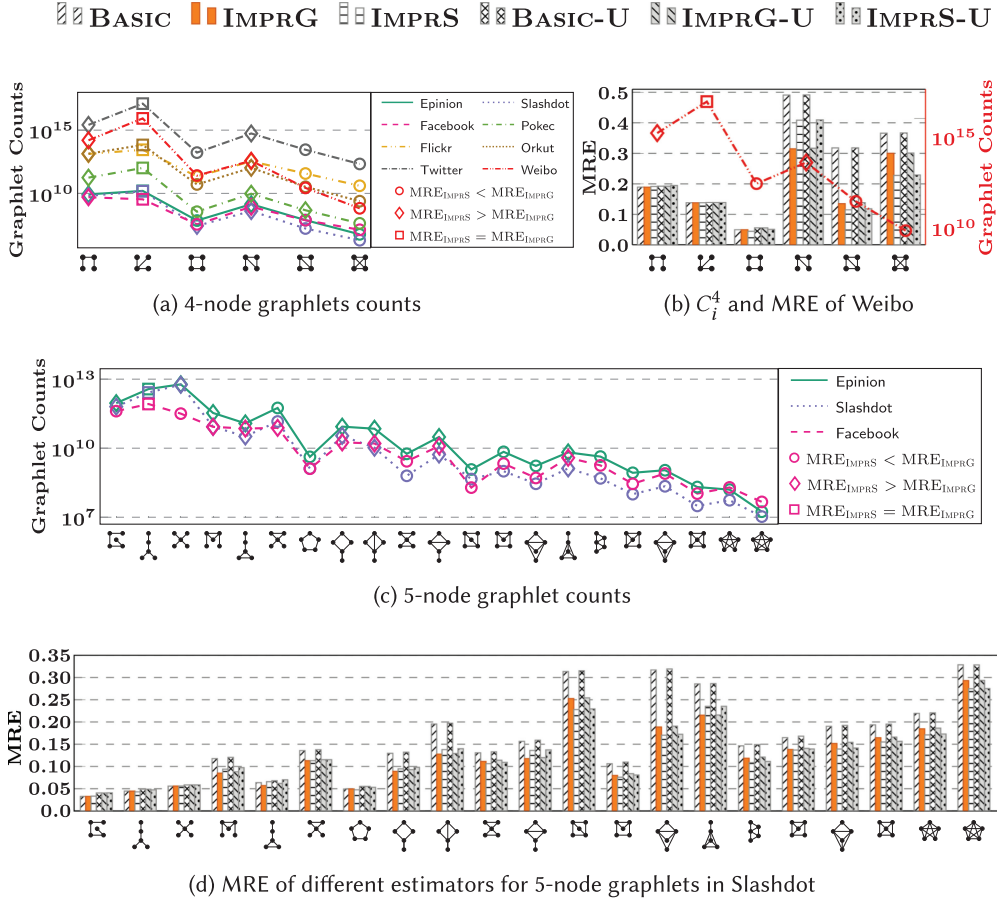


Fig. 8. Compare the accuracy of the improved estimators IMPRG and IMPRS. The sample size is 20K for all the estimators. Here, MRE_{IMPRS} denotes the MRE of the estimator IMPRS and MRE_{IMPRG} denotes the MRE of the estimator IMPRG. The markers of the line plots are explained as follows. The circle \circ denotes where $MRE_{IMPRS} < MRE_{IMPRG}$. The diamond \diamond denotes, where $MRE_{IMPRS} > MRE_{IMPRG}$. The square \square denotes where $MRE_{IMPRS} = MRE_{IMPRG}$, i.e., IMPRG and IMPRS have the same estimation.

- For graphlets without tails, the IMPRS has higher accuracy than IMPRG. For example, IMPRS reduces the MRE by 0.138 compared with BASIC estimator, while IMPRG only reduces the MRE by 0.066 when estimating the 4-node cliques in Weibo.
- For social networks with special access model, we can achieve the best performance by applying the estimator IMPRG to estimate the tailed graphlet counts and IMPRS to estimate the counts of graphlets without tails.

Note that for the tailed graphlets, the degrees of nodes in the tail have no contribution to the estimators. That maybe one reason why IMPRG has better performance than IMPRS for the tailed graphlets. Further theoretical analysis of estimator IMPRS is left as future work.

Convergence: To show the convergence properties of the estimators, we choose graphs Weibo, Twitter for 3-, 4-node graphlets, and Epinion, Slashdot for 5-node graphlets since they have the largest number of nodes for each sized graphlets whose ground truth can be obtained. The graphlets \mathfrak{A} ,

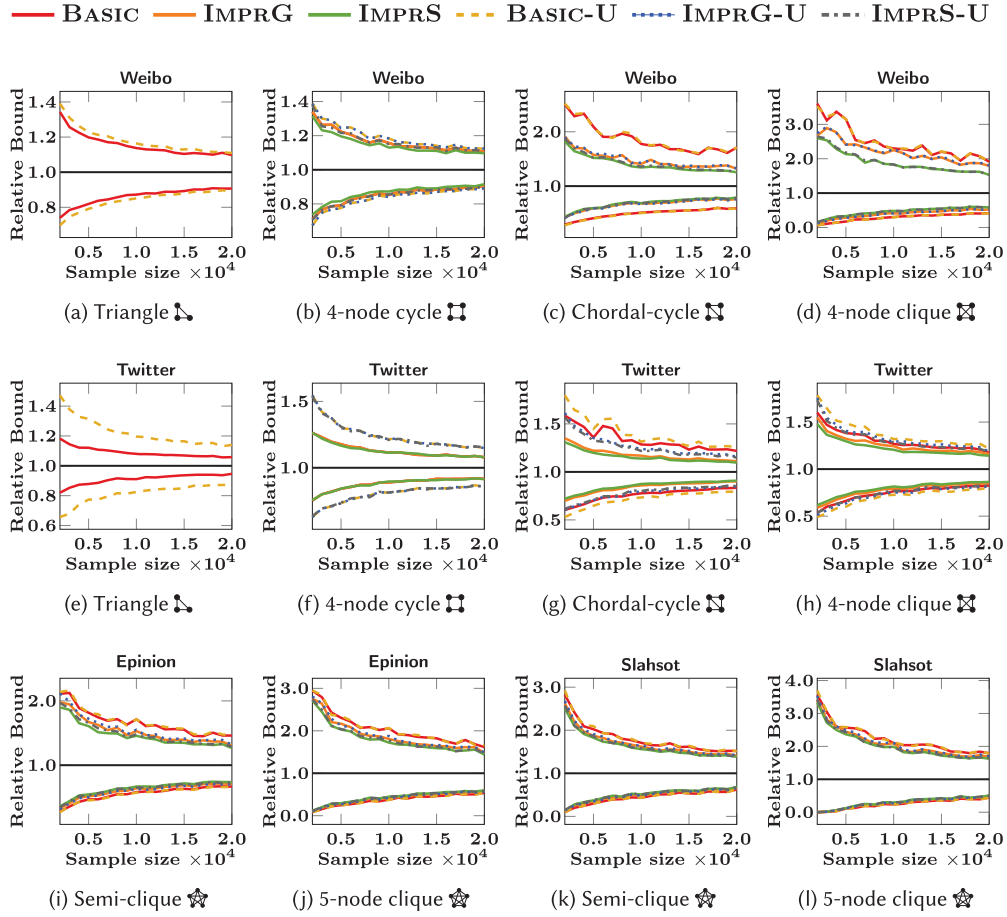


Fig. 9. Convergence analysis of the estimators. We show the relative confidence bound of the estimates, i.e., LB/Actual and UB/Actual. Here, actual represents the actual counts.

\bowtie , \square , \boxtimes , \star , \star are selected as the representative graphlets since they have the smallest counts among the 3-, 4-, 5-node graphlets, respectively. Figure 9 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 1,000 independent simulations. From the figure, we can observe that

- The estimates converge to the ground truth rapidly. Take Figure 9(h) as example. When the sample size varies from 10K to 20K, the relative interval between UB and LB changes from [0.79, 1.24] to [0.85, 1.16]. Besides, as we increase the sample size, the LB and UB are more balanced over the ground truth value.

Effect of estimated edges: Figures 7–9 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 Figure 9, 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.

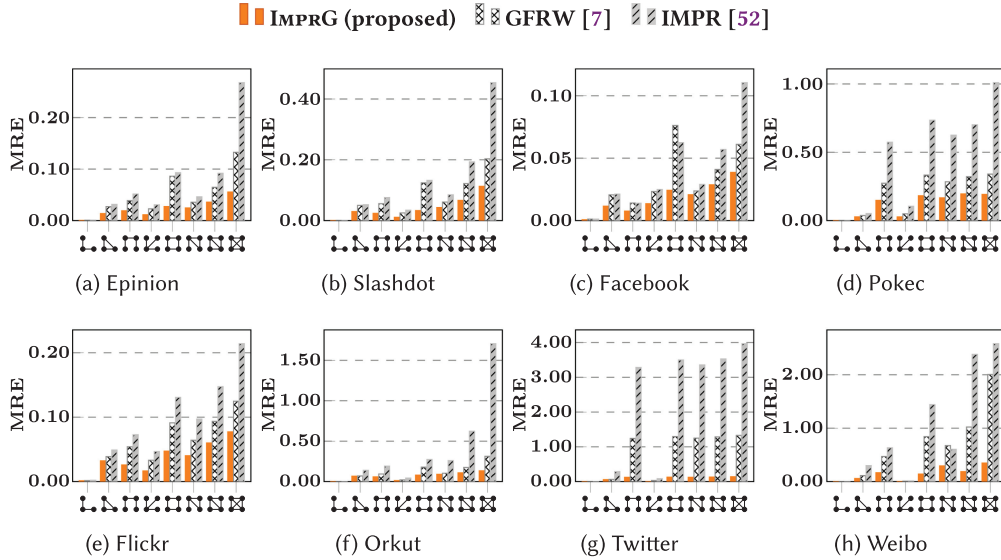


Fig. 10. Compare the accuracy of our proposed estimator IMPRG and prior state-of-the-art methods. The sample size for both methods is 20K.

8.3 Comparison with Previous Works

Here, we compare our improved estimator IMPRG with the state-of-the-art methods. We focus on the method **Pairwise Subgraph Random Walk (PSRW)** in [52] and the **General Framework based on Random Walk** (denoted as **GFRW** for short) in [7]. We choose IMPRG not IMPRS since both of IMPRG and the compared methods do not consider the optimization for the special access model. Hence, the comparison results reported in this section are pessimistic. Note that both of PSRW and GFRW are designed to estimating the 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 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* $|E|$. We give a brief synopsis for the compared methods.

8.3.1 PSRW. 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 since the number of edges in the super graph is difficult to compute.

8.3.2 GFRW. Different from [52], the method in [7] performs random walks on super graph whose node is a d -node induced connected subgraph of the original graph to estimate the relative counts of k -node graphlets. Here, d is a tunable parameter. The consecutive $k-d+1$ steps of the random walk on the super graph are considered as a k -node graphlet sample. For fair comparison with the method in [7], we choose the optimal parameter d reported in the article for 3-, 4-, 5-node relative graphlet count estimation.

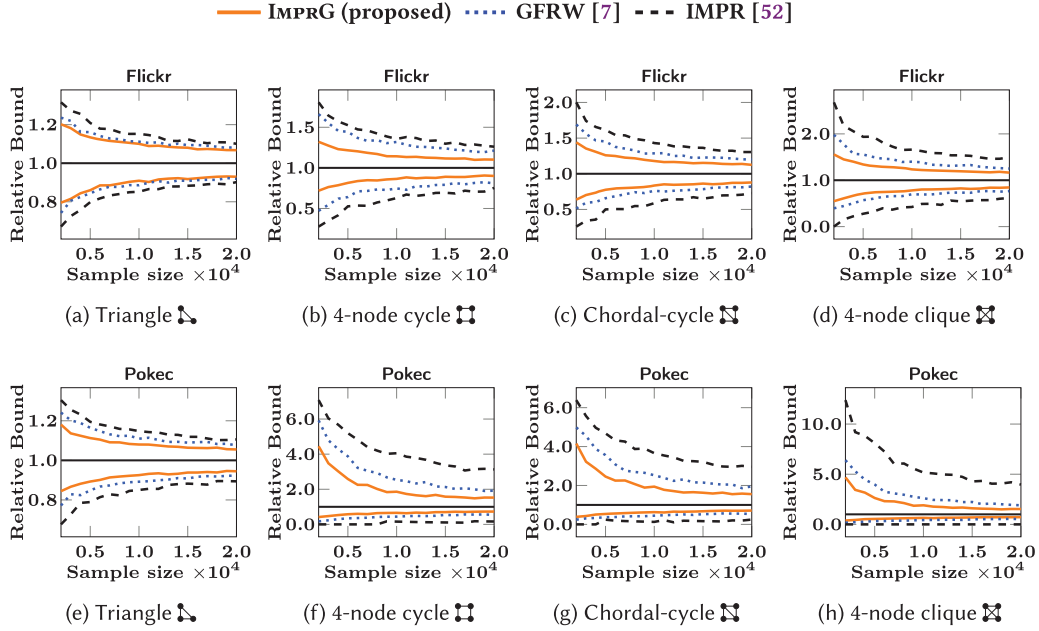


Fig. 11. Compare the convergence performance of the estimators. We show the relative confidence bound of the estimates, i.e., LB/Actual and UB/Actual. Here, actual represents the actual relative counts of subgraphs.

In the following, we present the empirical comparison between our methods and the state-of-the-art methods from various aspects.

Accuracy: The accuracy of GFRW, PSRW, and the proposed method IMPRG are compared in Figure 10. The sample size equals to 20K for all the methods. We show the accuracy on all the graphs in the datasets for all 3-, 4-node graphlets. We use the error metric MRE. Our proposed method outperforms PSRW and GFRW significantly. For example, the MRE of our method on estimating the relative count of 4-node clique (σ_6^4) is 2.8–28 times smaller than that of PSRW, 1.6–9.4 times smaller than that of GFRW. 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: Figure 11 compares the convergence performance of the estimators. We choose Flickr and Pokec for demonstration. All the methods 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 Figure 11, the gap between LB and UB of our estimator is no more than half of that produced by PSRW when estimating the 4-node graphlet relative counts. It implies that our proposed method also shows extraordinary performance on the estimation of relative counts.

Computation time: Table 8 compares the computation time of different methods. We ignore the query time of OSNs and only show the in-memory computation time of sampling methods when the sample size equals to 20K. Theoretically, the computation time of our method is $O(nL + 2^{2k-1}T)$, here $L = \sum_{v \in V} d_v^2 / |E|$, and n is the sample size. L represents the time spent on computing $f_i^k(X)$ and T represents the time to judge the graphlet type of a k -node subgraph sample. Let $C^{(d)}$ denote the time spent on generating neighbors of nodes in the super graph, where each node is a d -node

Table 8. Computation Time (In Seconds) of Different Methods

	3-node				4-node				5-node			
	Exact	IMPRG	GFRW	PSRW	Exact	IMPRG	GFRW	PSRW	Exact	IMPRG	GFRW	PSRW
Epinion	0.10	0.018	0.002	0.002	1.7	0.029	0.012	0.39	11091.1	0.16	0.06	44.1
Slashdot	0.09	0.014	0.002	0.002	1.5	0.024	0.012	0.41	5702.4	0.17	0.05	47.5
Facebook	0.52	0.019	0.007	0.008	1.6	0.038	0.026	0.25	4405.0	0.34	0.19	29.4
Pokec	7.78	0.020	0.008	0.010	249	0.094	0.027	2.79	–	–	–	–
Flickr	39.7	0.131	0.011	0.015	11194	0.552	0.034	3.62	–	–	–	–
Orkut	197.8	0.037	0.005	0.007	13608	0.077	0.018	3.74	–	–	–	–
Twitter	667.5	0.346	0.007	0.008	>1 week	2.349	0.018	79.9	–	–	–	–
Weibo	184.4	0.032	0.005	0.005	>1 day	0.850	0.014	36.4	–	–	–	–

Exact represents the exact enumeration methods. here we only show the results of 5-node graphlets for the graphs whose ground truth can be obtained with reasonable time.

subgraph in the original graph. The computation time of GFRW is $O(nC^{(d)} + 2^k T)$ and PSRW is $O(nC^{(k-1)} + 2^k T)$. We summarize the findings in Table 8 as follows:

- When $k = 3$, PSRW is computationally more efficient than our method IMPRG. However, when $k = 4, 5$, the computation cost of our method IMPRG is much smaller than PSRW. The reason is that $C^{(k)}$ increases quickly with k while L is independent on k .
- The method GFRW always has the least computation cost.
- Compared with GFRW, our method spends more time in exploring the neighborhood to increase the accuracy of the estimation. The computation time of our method and GFRW is negligible compared with the query time of OSNs. For all the graphs, the computation time of our method is less than 2.4 seconds. We conclude that our method achieves the best time cost and accuracy tradeoff.

9 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.

APPENDIXES

A PROOF OF THEOREMS AND LEMMAS

A.1 Proof of Theorem 4.1

THEOREM 4.1. *The average of the function $w_i^k(X)f_i^k(X)$ is*

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

which is an asymptotic unbiased estimator of C_i^k , i.e., count of graphlet g_i^k , when $\beta_i^k \neq 0$.

PROOF. Use the definition of the expectation and the Equation (1), we have

$$\begin{aligned} & \mathbb{E}_{\pi_M} \left[w_i^k(X) f_i^k(X) \right] \\ &= \sum_{X \in \mathcal{M}^{(k-1)}} \pi_M(X) \frac{1}{\beta_i^k} \frac{1}{\pi_M(X)} f_i^k(X) \\ &= \sum_{X \in \mathcal{M}^{(k-1)}} f_i^k(X) / \beta_i^k = C_i^k. \end{aligned}$$

According to the Theorem 3.1, we have \hat{C}_i^k converge to C_i^k almost surely. \square

A.2 Proof of Lemma 5.1

The following concentration inequality serves as the mathematical basis of our analytical bound. We use the inequality to prove Lemma 5.1.

THEOREM A.1 ([9, THEOREM 3]). *For an ergodic Markov chain with the state space V and stationary distribution π , define $T = \tau(\xi)$ as its ξ -mixing time for $\xi \leq 1/8$. Let v_1, \dots, v_n denote an n -step random walk starting from an initial distribution φ . Let $f : V \rightarrow [0, 1]$ be a function defined on the state space V . Define $\mu \triangleq \mathbb{E}_{\pi}[f]$ and $\mu_n \triangleq \frac{1}{n} \sum_{i=1}^n f(v_i)$. There exists some constant c (which is independent of μ , ξ and ϵ) such that*

$$\Pr [|\mu_n - \mu| > \epsilon\mu] \leq c \|\varphi\|_{\pi} \exp(-\epsilon^2 \mu n / (72T))$$

for $0 \leq \epsilon \leq 1$. Here, $\|\varphi\|_{\pi} \triangleq \sum_{v \in V} \frac{\varphi^2(v)}{\pi(v)}$.

LEMMA 5.1. *There is a constant value ζ , such that if*

$$n \geq B_1 \triangleq \zeta \frac{M_i^k \log(\|\varphi\|_{\pi} / \delta)}{C_i^k \epsilon^2} T,$$

we have

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

PROOF. Let $h_i^k(X) = \frac{w_i^k(X) f_i^k(X)}{M_i^k}$ ($h_i^k(X) \in [0, 1]$). We have $\mathbb{E}_{\pi_M}[h_i^k] = \mathbb{E}_{\pi_M}[w_i^k f_i^k] / M_i^k = C_i^k / M_i^k$. According to Theorem A.1, we have

$$\begin{aligned} & \Pr [|\hat{C}_i^k - C_i^k| \geq \epsilon C_i^k] = \Pr \left[\left| \frac{\hat{C}_i^k}{M_i^k} - \frac{C_i^k}{M_i^k} \right| \geq \epsilon \frac{C_i^k}{M_i^k} \right] \\ &= \Pr \left[\left| \frac{1}{n} \sum_{t=1}^n h_i^k(X_t) - \mathbb{E}_{\pi}[h_i^k] \right| \geq \epsilon \mathbb{E}_{\pi}[h_i^k] \right] \\ &\leq c \|\varphi\|_{\pi} \exp\left(-\epsilon^2 \frac{C_i^k}{M_i^k} n / (72T')\right) \end{aligned}$$

where T' is the 1/8-mixing time of the expanded Markov chain. Simple calculation shows that $T' = T$. Extracting n for which $\delta = c \|\varphi\|_{\pi} \exp(-\epsilon^2 \frac{C_i^k}{M_i^k} n / (72T))$, we have $n = \zeta \frac{M_i^k \log(\|\varphi\|_{\pi} / \delta)}{C_i^k \epsilon^2} T$. \square

We now prove the bound for the estimator designed for the situation where $|E|$ is unknown.

LEMMA A.2. *There is a constant ζ , such that if*

$$n \geq B_2 \triangleq \zeta \max \left\{ \frac{M_i^k}{C_i^k}, \frac{2|E|}{|V|} \right\} \frac{\log(2 \|\varphi\|_{\pi} / \delta)}{\epsilon^2} (9T),$$

we have

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

where \hat{C}_i^k is the estimator of C_i^k designed for unknown $|E|$.

PROOF. Recall that

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

We prove the bound by finding the steps to guarantee that both $\tilde{C}_i^k \triangleq \frac{1}{n} \sum_{t=1}^n \tilde{w}(X_t) f_i^k(X_t)$ and $\hat{d} \triangleq \frac{1}{n} \sum_{t=1}^n \frac{1}{d_{v_t}}$ are within $(1 \pm \epsilon/3)$ of their expected value with probability at least $1 - \delta/2$. According to the definition of $\tilde{w}_i^k(X) = w_i^k(X)/(2|E|)$, we have $\mathbb{E}_{\pi_M}[\tilde{C}_i^k] = \frac{C_i^k}{2|E|}$. Similar to the proof in Lemma 5.1, to guarantee \tilde{C}_i^k is within $(1 \pm \epsilon/3)$ of the expected value with probability at least $1 - \delta/2$, the sample size is at least $n_1 \triangleq \zeta \frac{M_i^k}{C_i^k} \frac{\log(2\|\varphi\|_{\pi}/\delta)}{\epsilon^2} (9T)$.

On the other hand, we have

$$\Pr \left[\left| \hat{d} - \mathbb{E}_{\pi}[\hat{d}] \right| \geq \frac{\epsilon}{3} \mathbb{E}_{\pi}[\hat{d}] \right] \leq c \|\varphi\|_{\pi} \exp \left(-\frac{\epsilon^2}{9} \frac{|V|}{2|E|} n / (72T) \right).$$

To guarantee that \hat{d} is within $(1 \pm \epsilon/3)$ of the expected value with probability $1 - \delta/2$, the sample size is at least $n_2 \triangleq \zeta \left(\frac{2|E|}{|V|} \right) \frac{\log(2\|\varphi\|_{\pi}/\delta)}{\epsilon^2} (9T)$. Hence, when the sample size is at least $\max\{n_1, n_2\}$, we have

$$(1 - \epsilon)C_i^k \leq \frac{(1 - \epsilon/3)C_i^k/2|E|}{(1 + \epsilon/3)1/2|E|} \leq \frac{\tilde{C}_i^k}{\hat{d}} \leq \frac{(1 + \epsilon/3)C_i^k/2|E|}{(1 - \epsilon/3)1/2|E|} \leq (1 + \epsilon)C_i^k$$

with probability at least $1 - \delta$. This completes the proof. \square

A.3 Properties of the Improved Estimator for General Access Model

A.3.1 Unbiasedness of the Improved Estimator.

THEOREM 6.1. *The average of the function $W_i^k(X) f_i^k(X)$*

$$\hat{C}_i^k \triangleq \frac{1}{n} \sum_{t=1}^n W_i^k(X_t) f_i^k(X_t) \quad (10)$$

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

PROOF. Let \mathcal{H}_{k-1} denote the set of all $(k-1)$ -node connected subgraphs in G .

$$\begin{aligned} & \mathbb{E}_{\pi_M} \left[W_i^k(X) f_i^k(X) \right] \\ &= \sum_{X \in \mathcal{M}^{(k-1)}} \pi_M(X) \frac{1}{\beta_i^k} \frac{|\mathcal{A}(X)|}{\sum_{X_a \in \mathcal{A}(X)} \pi_M(X_a)} f_i^k(X) \\ &= \frac{1}{\beta_i^k} \sum_{G_{k-1} \in \mathcal{H}_{k-1}} \left(\frac{F_i^k(G_{k-1})}{\sum_{X_a \in \mathcal{A}(G_{k-1})} \pi_M(X_a)} \sum_{X \in \mathcal{A}(G_{k-1})} \pi_M(X) \right) \\ &= \frac{1}{\beta_i^k} \sum_{G_{k-1} \in \mathcal{H}_{k-1}} F_i^k(G_{k-1}) = \frac{1}{\beta_i^k} \sum_{X \in \mathcal{M}^{(k-1)}} f_i^k(X) = C_i^k \end{aligned}$$

By SLLN, $\hat{C}_i^k \rightarrow C_i^k$ almost surely. \square

A.3.2 Accuracy Comparison Between the Basic Estimator and the Improved Estimator. Our estimators are asymptotically unbiased, the mean squared error of the estimators is almost equal to their “variance.” Hence, smaller variance of the estimators indicates higher accuracy. For the sample average

$$\hat{\mu}_n \triangleq \frac{1}{n} \sum_{t=1}^n f(X_t)$$

based on an irreducible Markov chain $\{X_t\}$ with its stationary distribution $\boldsymbol{\pi}$, we introduce the asymptotic variance of the estimator $\hat{\mu}_n$, which is defined as

$$\sigma^2(f) \triangleq \lim_{n \rightarrow \infty} n \cdot \text{Var}(\hat{\mu}_n) = \lim_{n \rightarrow \infty} \frac{1}{n} \mathbb{E} \left\{ \left[\sum_{t=1}^n (f(X_t) - \mathbb{E}_{\boldsymbol{\pi}}[f]) \right]^2 \right\} \quad (12)$$

for any function f with $\mathbb{E}_{\boldsymbol{\pi}}[f^2] < \infty$. We review the central limit theorem (CLT) as follows.

THEOREM A.3 (CENTRAL LIMIT THEOREM [25]). *For a finite, irreducible Markov chain $\{X_t\}$ with its stationary distribution $\boldsymbol{\pi}$,*

$$\sqrt{n}[\hat{\mu}_n(f) - \mathbb{E}_{\boldsymbol{\pi}}[f]] \xrightarrow{d} N(0, \sigma^2(f)), \text{ as } n \rightarrow \infty,$$

for any function f with $\mathbb{E}_{\boldsymbol{\pi}}[f^2] < \infty$ regardless of initial distribution, and $\sigma^2(f)$ is given by Equation (12).

Asymptotic variance can be used to compare the performance of different MCMC estimators through its connection to the CLT. Similar to the i.i.d. samples, the asymptotic variance $\sigma^2(f)$ can be used to decide approximately how many samples are needed to achieve certain accuracy. Note that $\sigma^2(f)$ is independent on the initial state of the Markov chain. When the Markov chain is already in the stationary regimes, the estimator with smaller asymptotic variance has better performance. We rewrite the asymptotic variance formula for further analysis. Let \mathcal{M} denote the state space of the finite Markov chain. Suppose the chain $\{X_t\}$ is in the stationary regime, i.e., $X_0 \sim \boldsymbol{\pi}, X_t \sim \boldsymbol{\pi}$. Then, Equation (12) can be rewritten as follows:

$$\begin{aligned} \sigma^2(f) &= \text{Var}(f(X_0)) + 2 \sum_{k=1}^{\infty} \text{Cov}(f(X_0), f(X_k)) \\ &= \underbrace{\left(\sum_{X \in \mathcal{M}} [f(X) - \mathbb{E}[f]]^2 \pi(X) \right)}_{\gamma_0} + 2 \sum_{k=1}^{\infty} \underbrace{\left\{ \mathbb{E}[f(X_0)f(X_k)] - \mathbb{E}_{\boldsymbol{\pi}}^2[f] \right\}}_{\gamma_k} \\ &= \gamma_0 + 2 \sum_{k=1}^{\infty} \gamma_k. \end{aligned}$$

Here, γ_k is also called “lag k autocovariance.” Different from the i.i.d. samples, the correlation structure over random samples given by the Markov chains can significantly affect the asymptotic variances. The computation of the asymptotic variance depends on the transition matrix, which is basically another form the adjacent matrix of the underlying networks. Hence, it is impractical to compute the asymptotic variance in general.

To compare the accuracy of these estimators, we only need to compare their asymptotic variance. We first compare the γ_0 of the basic estimator and the improved estimator. The following lemma shows that γ_0 of the improved estimator is smaller.

LEMMA A.4. *Let $\gamma_0 = \text{Var}_{\boldsymbol{\pi}_M}[w_i^k(X)f_i^k(X)]$ and $\gamma'_0 = \text{Var}_{\boldsymbol{\pi}_M}[W_i^k(X)f_i^k(X)]$. We have $\gamma'_0 \leq \gamma_0$.*

PROOF. Let H_{k-1} denote the set of all $(k-1)$ -node connected subgraphs in G .

$$\begin{aligned}
\gamma_0 - \gamma'_0 &= \text{Var}_{\pi_M}[w_i^k(X)f_i^k(X)] - \text{Var}_{\pi_M}[W_i^k(X)f_i^k(X)] \\
&= \sum_{X \in \mathcal{M}^{k-1}} \left((w_i^k(X)f_i^k(X))^2 - (W_i^k(X)f_i^k(X))^2 \right) \pi_M(X) \\
&= \frac{1}{(\beta_i^k)^2} \sum_{G_{k-1} \in \mathcal{H}_{k-1}} \left\{ \left(\sum_{X_a \in \mathcal{A}(G_{k-1})} \frac{(f_i^k(X_a))^2}{\pi_M(X_a)} \right) - \left(\sum_{X_a \in \mathcal{A}(G_{k-1})} \frac{|A(G_{k-1})|^2 \pi_M(X) (f_i^k(X_a))^2}{(\sum_{X_b \in \mathcal{A}(G_{k-1})} \pi_M(X_b))^2} \right) \right\} \\
&= \sum_{G_{k-1} \in \mathcal{H}_{k-1}} \frac{(F_i^k(G_{k-1}))^2}{|\mathcal{A}(G_{k-1})|^2 (\beta_i^k)^2} \left\{ \left(\sum_{X_a \in \mathcal{A}(G_{k-1})} \frac{1}{\pi_M(X_a)} \right) - \left(\frac{|A(G_{k-1})|^2}{(\sum_{X_a \in \mathcal{A}(G_{k-1})} \pi_M(X_a))} \right) \right\}
\end{aligned}$$

According the fact that arithmetic mean is always not smaller than harmonic mean, we have

$$\frac{1}{|\mathcal{A}(G_{k-1})|} \sum_{X_a \in \mathcal{A}(G_{k-1})} \frac{1}{\pi_M(X_a)} \geq \frac{|A(G_{k-1})|}{(\sum_{X_a \in \mathcal{A}(G_{k-1})} \pi_M(X_a))}.$$

This leads to $\gamma_0 - \gamma'_0 \geq 0$. □

The covariance part is much more difficult to analyze. We give some detailed analysis of the covariance in the following. For a general Markov chain with finite state space \mathcal{M} and stationary distribution π , the sum of lag k autocovariance can also be expressed as

$$\sum_{k=1}^{\infty} \gamma_k = \sum_{\substack{X_i \neq X_a \\ X_i, X_j \in \mathcal{M}}} \pi(X_i) g_a(X_i, X_j) (f(X_i) - \mathbb{E}_{\pi}[f]) (f(X_j) - \mathbb{E}_{\pi}[f]),$$

where X_a is a state in \mathcal{M} and the asymptotic variance does not depend on the choice of X_a [50]. Here $g_a(X_i, X_j) = \mathbb{E}[N_j^a | X_i]$ and N_j^a is a random variable which counts, after the Markov chain leaves from the initial state X_i , how many times it has visited state X_j , until it reaches state X_a for the first time. The value of $g_a(i, j)$ depends on the transition matrix of the Markov chain. The detailed computation of $g_a(i, j)$ can be found in [50]. For our problem, let γ_k denote the lag k autocovariance of the basic estimator and γ'_k denote the lag k autocovariance of the improved estimator. For simplicity, we replace $w_i^k(X)f_i^k(X)$ with $w(X)f(X)$ and $W_i^k(X)f_i^k(X)$ with $W(X)f(X)$.

$$\sum_{k=1}^{\infty} \gamma_k - \sum_{k=1}^{\infty} \gamma'_k = \sum_{\substack{X_i \neq X_a \\ X_i, X_j \in \mathcal{M}^{k-1}}} \pi_M(X_i) g_a(X_i, X_j) f(X_i) f(X_j) (w(X_i)w(X_j) - W(X_i)W(X_j))$$

For the covariance part, we cannot decide whether the improved estimator has smaller covariance since the asymptotic variance depends on the transition matrix. We leave it as an open question to be solved in future work.

A.4 Proof of Theorem 6.2

THEOREM 6.2. *The average of the function $\sum_{G_k \in \mathcal{S}(X)} h_i^k(G_k) / \mathcal{P}(G_k)$*

$$\hat{C}_i^k = \frac{1}{n} \sum_{t=1}^n \left(\sum_{G_k \in \mathcal{S}(X_t)} \frac{h_i^k(G_k)}{\mathcal{P}(G_k)} \right) \quad (11)$$

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

PROOF. Let \mathcal{H}_k be the set of all k -node connected subgraphs in G . A subgraph G_k isomorphic to g_i^k is denoted as $G_k \simeq g_i^k$.

$$\mathbb{E}_{\pi_M} \left[\sum_{G_k \in \mathcal{S}(X)} \frac{h_i^k(G_k)}{\mathcal{P}(G_k)} \right] = \sum_{X \in \mathcal{M}^{(k-1)}} \left(\pi_M(X) \sum_{G_k \in \mathcal{S}(X)} \frac{h_i^k(G_k)}{\mathcal{P}(G_k)} \right) \quad (13a)$$

$$= \sum_{X \in \mathcal{M}^{(k-1)}} \left(\sum_{\substack{G_k \in \mathcal{S}(X) \\ G_k \simeq g_i^k}} \frac{\pi_M(X)}{\sum_{X_a \in \mathcal{B}(G_k)} \pi_M(X_a)} \right) \quad (13b)$$

$$= \sum_{\substack{G_k \in \mathcal{H}_k \\ G_k \simeq g_i^k}} \sum_{X \in \mathcal{B}(G_k)} \left(\frac{\pi_M(X)}{\sum_{X_a \in \mathcal{B}(G_k)} \pi_M(X_a)} \right) = C_i^k \quad (13c)$$

Equations (13a) and (13b) are expanded from previous lines based on the definitions. Equation (13c) exchanges the summation order of Equation (13b). Each state X corresponds to $f_i^k(X)$ k -node connected subgraphs. Each $G_k \simeq g_i^k$ corresponds to β_i^k states. The summation $\sum_{X \in \mathcal{M}^{(k-1)}} \sum_{G_k \in \mathcal{S}(X)}$ • and $\sum_{G_k \in \mathcal{H}_k} \sum_{X \in \mathcal{B}(G_k)}$ • have the same combination of X and G_k , which validates the correctness of Equation (13c). Finally, we have $\hat{C}_i^k \rightarrow C_i^k$ almost surely by SLLN. \square

B PSEUDO-CODE OF COMPUTING α_i^k AND β_i^k

ALGORITHM 2: Computation of α_j^{k-1}

Input: $G_{k-1} = (\{v_1, \dots, v_{k-1}\}, E_{k-1})$ isomorphic to g_j^{k-1}

Output: α_j^{k-1}

- 1: $\alpha_j^{k-1} \leftarrow 0$
 - 2: **for each** permutation x_1, \dots, x_{k-1} of $\{1, \dots, k-1\}$ **do**
 - 3: **for** $i \in \{1, \dots, k-2\}$ **do**
 - 4: **if** $(v_{x_i}, v_{x_{i+1}}) \notin E_{k-1}$ **then**
 - 5: **continue**
 - 6: $\alpha_j^{k-1} \leftarrow \alpha_j^{k-1} + 1$
 - 7: **return** α_j^{k-1}
-

Table 9. Comparison Between $\pi_M(X)$ and $P(G_{k-1})$ When $k = 4, 5$

G_{k-1}	α_i^k	State	$\pi_M(X)$	$P(G_{k-1})/ \mathcal{A}(G_{k-1}) $
	2	(u, v, w)	$\frac{1}{2 E } \frac{1}{d_v}$	$\frac{1}{2 E } \frac{1}{d_v}$
	6	(u, v, w)	$\frac{1}{2 E } \frac{1}{d_v}$	$\frac{1}{6 E } \left(\frac{1}{d_u} + \frac{1}{d_v} + \frac{1}{d_w} \right)$
	2	(u, v, w, z)	$\frac{1}{2 E } \frac{1}{d_v d_w}$	$\frac{1}{2 E } \frac{1}{d_v d_w}$
	0	(u, v, w, z)	-	-
	8	(u, v, w, z)	$\frac{1}{2 E } \frac{1}{d_v d_w}$	$\frac{1}{8 E } \left(\frac{1}{d_v d_w} + \frac{1}{d_w d_z} + \frac{1}{d_z d_u} + \frac{1}{d_u d_v} \right)$
	4	(u, v, w, z)	$\frac{1}{2 E } \frac{1}{d_v d_w}$	$\frac{1}{4 E } \left(\frac{1}{d_v d_w} + \frac{1}{d_u d_w} \right)$
	12	(u, v, w, z)	$\frac{1}{2 E } \frac{1}{d_v d_w}$	$\frac{1}{12 E } \left(\frac{1}{d_v d_w} + \frac{1}{d_w d_z} + \frac{1}{d_z d_u} + \frac{1}{d_u d_v} + \frac{2}{d_u d_w} \right)$
	24	(u, v, w, z)	$\frac{1}{2 E } \frac{1}{d_v d_w}$	$\frac{1}{24 E } \left(\frac{1}{d_v d_w} + \frac{1}{d_w d_z} + \frac{1}{d_z d_u} + \frac{1}{d_u d_v} + \frac{1}{d_u d_w} + \frac{1}{d_v d_z} \right)$

ALGORITHM 3: Computation of β_i^k .**Input:** $G_k = (V_k, E_k)$ isomorphic to g_i^k **Output:** β_i^k

- 1: $t_j \leftarrow 0, \forall 1 \leq j \leq |\mathcal{G}^{k-1}|$
- 2: **for** $v \in V_k$ **do**
- 3: $G_{k-1} \leftarrow (k-1)$ -node subgraph induced by $V_k \setminus \{v\}$
- 4: **if** G_{k-1} is a disconnected subgraph **then**
- 5: **continue**
- 6: $j \leftarrow \text{id of } (k-1)\text{-node graphlet } G_{k-1} \text{ isomorphic to}$
- 7: $t_j \leftarrow t_j + 1$
- 8: $\beta_i^k \leftarrow \sum_{j=1}^{|\mathcal{G}^{k-1}|} t_j \alpha_j^{k-1}$
- 9: **return** β_i^k

C COMPARISON BETWEEN $\pi_M(X)$ AND $P(G_{k-1})$

Here, we let G_{k-1} be the subgraph induced by the nodes in the state X . Table 9 illustrates the detailed comparison between $\pi_M(X)$ and $P(G_{k-1})/|\mathcal{A}(G_{k-1})|$.

REFERENCES

- [1] Louigi Addario-Berry and Tao Lei. 2012. The mixing time of the Newman: Watts small world. In *Proc. SODA*. 1661–1668.
- [2] Foto N. Afrati, Dimitris Fotakis, and Jeffrey D. Ullman. 2013. Enumerating subgraph instances using map-reduce. In *Proc. ICDE*. 62–73.
- [3] Nesreen K. Ahmed, Nick Duffield, Jennifer Neville, and Ramana Kompella. 2014. Graph sample and hold: A framework for big-graph analytics. In *Proc. KDD*. 1446–1455.
- [4] Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick Duffield. 2015. Efficient graphlet counting for large networks. In *Proc. ICDM*. 1–10.
- [5] N. Alon, R. Yuster, and U. Zwick. 1997. Finding and counting given length cycles. *Algorithmica* 17, 3 (1997), 209–223.

- [6] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. Al Hasan. 2012. GUISE: Uniform sampling of graphlets for large graph analysis. In *Proc. ICDM*. 91–100.
- [7] Xiaowei Chen, Yongkun Li, Pinghui Wang, and John Lui. 2016. A general framework for estimating graphlet statistics via random walk. In *Proc. VLDB*, vol. 10, 3 (2016), 253–264.
- [8] Xiaowei Chen and John Lui. 2016. Mining graphlet counts in online social networks. In *Proc. ICDM*. 71–80.
- [9] Kai-Min Chung, Henry Lam, Zhenming Liu, and Michael Mitzenmacher. 2012. Chernoff-Hoeffding bounds for Markov chains: Generalized and simplified. In *Proc. STACS*, vol. 14. LIPIcs, 124–135.
- [10] Radu Curticapean, Holger Dell, and Dániel Marx. 2017. Homomorphisms are a good basis for counting small sub-graphs. In *Proc. STOC*. 210–223.
- [11] Anirban Dasgupta, Ravi Kumar, and Tamas Sarlos. 2014. On estimating the average degree. In *Proc. WWW*. 795–806.
- [12] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. 2016. TRIEST: Counting local and global triangles in fully-dynamic streams with fixed memory size. In *Proc. KDD*. 825–834.
- [13] Reinhard Diestel. 2012. *Graph Theory*, (4th ed.). Springer.
- [14] Ethan R. Elenberg, Karthikeyan Shanmugam, Michael Borokhovich, and Alexandros G. Dimakis. 2015. Beyond triangles: A distributed framework for estimating 3-profiles of large graphs. In *Proc. KDD*. 229–238.
- [15] Ethan R. Elenberg, Karthikeyan Shanmugam, Michael Borokhovich, and Alexandros G. Dimakis. 2016. Distributed estimation of graph 4-profiles. In *Proc. WWW*. 483–493.
- [16] Tobias Friedrich and Anton Krohmer. 2015. Cliques in hyperbolic random graphs. In *Proc. INFOCOM*. 1544–1552.
- [17] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. 2010. Walking in Facebook: A case study of unbiased sampling of OSNs. In *Proc. INFOCOM*. 1–9.
- [18] Olle Häggström. 2002. *Finite Markov Chains and Algorithmic Applications*. Vol. 52. Cambridge University Press.
- [19] Frank Harary and Edgar M. Palmer. 1973. *Graphical Enumeration*. Academic Press, NY. DOI: <https://doi.org/10.1137/1016039>
- [20] Adib Hasan, Po-Chien Chung, and Wayne Hayes. 2017. Graphettes: Constant-time determination of graphlet and orbit identity including (possibly disconnected) graphlets up to size 8. *PLoS One* 12, 8 (2017), 1–12.
- [21] Tomaž Hočevar and Janez Demšar. 2014. A combinatorial approach to graphlet counting. *Bioinformatics* 30, 4 (2014), 559–565.
- [22] Shweta Jain and C. Seshadhri. 2017. A fast and provable method for estimating clique counts using Turán’s theorem. In *Proc. WWW*. 441–449.
- [23] Madhav Jha, C. Seshadhri, and Ali Pinar. 2015. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proc. WWW*. 495–505.
- [24] Madhav Jha, C. Seshadhri, and Ali Pinar. 2015. A space-efficient streaming algorithm for estimating transitivity and triangle counts using the birthday paradox. *ACM Transactions on Knowledge Discovery from Data* 9, 3 (2015), 15:1–15:21.
- [25] Galin L. Jones and others. 2004. On the Markov chain central limit theorem. *Probability Surveys* 1, 299–320 (2004), 5–1.
- [26] Liran Katzir and Stephen J. Hardiman. 2015. Estimating clustering coefficients and size of social networks via random walk. *ACM Transactions on the Web* 9, 4 (2015), 19:1–19:20.
- [27] Liran Katzir, Edo Liberty, and Oren Somekh. 2011. Estimating sizes of social networks via biased sampling. In *Proc. WWW*. 597–606.
- [28] Hyeonji Kim, Junyoung Lee, Sourav S. Bhowmick, Wook-Shin Han, JeongHoon Lee, Seongyun Ko, and Moath H. A. Jarrah. 2016. DUALSIM: Parallel subgraph enumeration in a massive graph on a single machine. In *Proc. SIGMOD*. 1231–1245.
- [29] KONECT Datasets. 2015. KONECT Datasets: The Koblenz Network Collection. Retrieved April 27, 2017 from <http://konect.uni-koblenz.de>. (2015).
- [30] Longbin Lai, Lu Qin, Xuemin Lin, and Lijun Chang. 2015. Scalable subgraph enumeration in MapReduce. *Publication of the Very Large Database Endowment*. 8, 10 (2015), 974–985.
- [31] Chul-Ho Lee, Xin Xu, and Do Young Eun. 2012. Beyond random walk and metropolis-hastings samplers: Why you should not backtrack for unbiased graph sampling. In *Proc. SIGMETRICS*. 319–330.
- [32] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. Retrieved April 20, 2016 from <http://snap.stanford.edu/data>. (2014).
- [33] R. H. Li, J. X. Yu, L. Qin, R. Mao, and T. Jin. 2015. On random walk based graph sampling. In *Proc. ICDE*. 927–938.
- [34] Yongsub Lim and U. Kang. 2015. MASCOT: Memory-efficient and accurate sampling for counting local triangles in graph streams. In *Proc. KDD*. 685–694.
- [35] László Lovász. 1993. Random walks on graphs. *Combinatorics, Paul Erdos is Eighty* 2 (1993), 1–46.
- [36] Sean P. Meyn and Richard L. Tweedie. 2012. *Markov Chains and Stochastic Stability*. Springer Science & Business Media.

- [37] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and analysis of online social networks. In *Proc. IMC*. 29–42.
- [38] Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. 2010. Measuring the mixing time of social graphs. In *Proc. IMC*. 383–389.
- [39] James R. Norris. 1998. *Markov Chains*. Number 2. Cambridge University Press.
- [40] Art B. Owen. 2013. Monte Carlo theory, methods and examples. Retrieved September 11, 2015 from <http://statweb.stanford.edu/~owen/mc/>. (2013).
- [41] Ali Pinar, C. Seshadhri, and Vaidyanathan Vishal. 2017. ESCAPE: Efficiently counting all 5-vertex subgraphs. In *Proc. WWW*. 1431–1440.
- [42] M. Rahman, M. A. Bhuiyan, and M. Al Hasan. 2014. Graft: An efficient graphlet counting method for large graph analysis. *IEEE Transactions on Knowledge and Data Engineering* 26, 10 (2014), 2466–2478.
- [43] Gareth O. Roberts and Jeffrey S. Rosenthal. 2004. General state space Markov chains and MCMC algorithms. *Probability Surveys* 1 (2004), 20–71.
- [44] Ryan A. Rossi and Nesreen K. Ahmed. 2015. Network Repository: A Scientific Network Data Repository with Interactive Visualization and Mining Tools. Retrieved April 20, 2016 from <http://networkrepository.com>. (2015).
- [45] Thomas Schank and Dorothea Wagner. 2005. Finding, counting and listing all triangles in large graphs, an experimental study. In *Proc. WEA*. 606–609.
- [46] Comandur Seshadhri, Ali Pinar, and Tamara G. Kolda. 2013. Triadic measures on graphs: The power of wedge sampling. In *Proc. SIAM SDM*. 10–18.
- [47] Yingxia Shao, Bin Cui, Lei Chen, Lin Ma, Junjie Yao, and Ning Xu. 2014. Parallel subgraph listing in a large-scale graph. In *Proc. SIGMOD*. 625–636.
- [48] Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In *Proc. AISTATS*, vol. 5. 488–495.
- [49] N. J. A. Sloane. 1995. The Online Encyclopedia of Integer Sequences, A001349. Retrieved January 1, 2018 from <https://oeis.org/A001349>. (1995).
- [50] Samis Trevezas and Nikolaos Limnios. 2009. Variance estimation in the central limit theorem for Markov chains. *Journal of Statistical Planning and Inference* 139, 7 (2009), 2242–2253.
- [51] Johan Ugander, Lars Backstrom, and Jon Kleinberg. 2013. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *Proc. WWW*. 1307–1318.
- [52] Pinghui Wang, John C. S. Lui, Bruno Ribeiro, Don Towsley, Junzhou Zhao, and Xiaohong Guan. 2014. Efficiently estimating motif statistics of large networks. *ACM Transactions on Knowledge Discovery from Data* 9, 2 (2014), 8:1–8:27.
- [53] P. Wang, J. C. S. Lui, D. Towsley, and J. Zhao. 2016. Minfer: A method of inferring motif statistics from sampled edges. In *Proc. ICDE*. 1050–1061.
- [54] P. Wang, J. Zhao, X. Zhang, Z. Li, J. Cheng, J. C. S. Lui, D. Towsley, J. Tao, and X. Guan. 2018. MOSS-5: A fast method of approximating counts of 5-node graphlets in large graphs. *IEEE Transactions on Knowledge and Data Engineering* 30, 1 (2018), 73–86.
- [55] Junzhou Zhao, John C. S. Lui, Don Towsley, Pinghui Wang, and Xiaohong Guan. 2015. Tracking triadic cardinality distributions for burst detection in social activity streams. In *Proc. COSN*. 15–25.
- [56] Zhuojie Zhou, Nan Zhang, and Gautam Das. 2015. Leveraging history for faster sampling of online social networks. *Publication of the Very Large Database Endowment* 8, 10 (2015), 1034–1045.

Received April 2017; revised January 2018; accepted January 2018