A General Framework for Estimating Graphlet Statistics via Random Walk

Xiaowei Chen[1], Yongkun Li[2], Pinghui Wang[3], John C.S. Lui[1]
[1]The Chinese University of Hong Kong
[2]University of Science and Technology of China
[3]Xi'an Jiaotong University

[1]{xwchen, cslui}@cse.cuhk.edu.hk, [2]ykli@ustc.edu.cn, [3]phwang@mail.xjtu.edu.cn

ABSTRACT

Graphlets are induced subgraph patterns and have been frequently applied to characterize the local topology structures of graphs across various domains, e.g., online social networks (OSNs) and biological networks. Discovering and computing graphlet statistics are highly challenging. First, the massive size of real-world graphs makes the exact computation of graphlets extremely expensive. Secondly, the graph topology may not be readily available so one has to resort to web crawling using the available application programming interfaces (APIs). In this work, we propose a general and novel framework to estimate graphlet statistics of "*any size*". Our framework is based on collecting samples through consecutive steps of random walks. We derive an analytical bound on the sample size (via the Chernoff-Hoeffding technique) to guarantee the convergence of our unbiased estimator. To further improve the accuracy, we introduce two novel optimization techniques to reduce the lower bound on the sample size. Experimental evaluations demonstrate that our methods outperform the state-of-the-art method up to an order of magnitude both in terms of accuracy and time cost.

1. INTRODUCTION

Graphlets are defined as induced subgraph patterns in real-world networks [23]. Unlike some global properties such as degree distribution, the frequencies of graphlets provide important statistics to characterize the local topology structures of networks. Decomposing networks into small k-node graphlets has been a fundamental approach to characterize the local structures of real-world complex networks. Graphlets also have numerous applications ranging from biology to network science. Applications in biology include protein detection [21], biological network comparison [28] and disease gene identification [22]. In network science, the researchers have applied graphlets for web spam detection [5], anomaly detection [4], as well as social network structure analysis [34].

Research Problem. In most applications, relative frequencies among various graphlets are sufficient. One such example is building graphlet kernels for large graph comparison [32]. In this work, we focus on relative graphlet frequencies discovery and computation. More specifically, we propose efficient sampling methods to compute the *percentage* of a specific k-node graphlet type within all k-node graphlets in a given graph. The percentage of a particular k-node graphlet type is called the "*graphlet concentration*" or "*graphlet statistics*".

Challenges. The straightforward approach to compute the graphlet concentration is via exhaustive counting. However, there exist a large number of graphlets even for a moderately sized graph. For example, Facebook [1] in our datasets with 817K edges has 9×10^9 4-node graphlets and 2×10^{12} 5-node graphlets. Due to the combinatorial explosion problem, how to count graphlets efficiently is a long standing research problem. Some techniques, such as leveraging parallelism provided by multi-core architecture [3], exploiting combinatorial relationships between graphlets [13], and employing distributed systems [33], have been applied to speed up the graphlet counting. However, these exhaustive counting algorithms are not scalable because they need to explore each graphlet at least once. Even with those highly tuned algorithms, exhaustive counting of graphlets has prohibitive computation cost for real-world large graphs. An alternative approach is to adopt "*sampling algorithms*" to achieve significant speedup with acceptable error. Several methods based on sampling have been proposed to address the challenge of graphlet counting [14, 37, 35, 6, 36, 9].

Another challenge is the restricted access to the complete graph data. For example, most OSNs' service providers are unwilling to share the complete data for public use. The underlying network may only be available by calling some application programming interfaces (APIs), which support the function to retrieve a list of user's friends. Graph sampling through crawling is widely used in this scenario to estimate graph properties such as degree distribution [17, 19, 11], clustering coefficient [12] and size of graphs [15]. In this work, we assume that the graph data has to be externally accessed, either through remote databases or by calling APIs provided by the operators of OSNs.

The aim of this work is to design and implement efficient random walk-based methods to estimate graphlet concentration for restricted accessed graphs. Note that estimating graphlet concentration is a more complicated task than estimating other graph properties such as degree distribution. For degree distribution, one can randomly walk on

the graph to collect node samples and then remove the bias. For graphlet concentration, one needs to consider the various local structures. A single node sample *cannot* tell us information about the local structures. We need to map a random walk to a Markov chain, and carefully define the state space and its transition matrix so as to ensure that the state space contains all the k-node graphlets.

1.1 Related Works and Existing Problems.

Previous studies on graphlet counts or concentration include sampling methods for (1) memory-based graphs [31, 14, 37, 29], (2) streaming graphs [36, 2], and (3) restricted accessed graphs [12, 6, 35]. The state-of-the-art sampling methods for memory-based graphs are wedge sampling [31] and path sampling [14]. Wedge sampling in [31] generates uniformly random wedges (i.e., paths of length two) from the graphs to estimate triadic measures (e.g., number of triangles, clustering coefficient). Later on, Jha et al. [14] extended the idea of wedge sampling and proposed path sampling to estimate the number of 4-node graphlets in the graphs. However, both of wedge sampling and path sampling need to access the whole graph data, which renders them impractical for restricted accessed graphs.

Estimating graphlet counts for streaming graphs has been studied in [36, 2]. Ahmed et al. [2] proposed the *graph sample and hold* method to estimate the triangle counts in the graphs. Wang et al. [36] proposed a method to infer the number of any k-node graphlets in the graph with a set of uniformly sampled edges from the graph stream. These streaming sampling methods access each edge at least once and are not applicable to restricted accessed graphs.

Most relevant to our framework are previous random walk-based methods [12, 6, 35] designed for graphs with restricted access. Hardiman and Katzir[12] proposed a random walk-based method to estimate the clustering coefficient, which is a variant of the 3-node graphlet concentration we study. Bhuiyan et al. [6] proposed *GUISE*, which is based on the Metropolis-Hasting random walk on a subgraph relationship graph, whose nodes are all the 3, 4, 5-node subgraphs. They aimed to estimate 3, 4, 5-node graphlet statistics simultaneously, but *GUISE* suffers from rejection of samples. In [35], authors proposed three random walk-based methods: subgraph random walk (SRW), pairwise subgraph random walk ($PSRW$), and mix subgraph sampling (MSS). MSS is an extension of $PSRW$ to estimate $k-1, k, k+1$-node graphlets jointly. The simulation results show that $PSRW$ outperforms SRW in estimation accuracy. To the best of our knowledge, $PSRW$ is the state-of-the-art random walk-based method to estimate graphlet statistics for restricted graphs.

We denote the subgraph relationship graph in [35] as $G^{(d)}$, and each element in $G^{(d)}$ is a d-node connected subgraphs in the original graph. The main idea of $PSRW$ is to collect k-node graphlet samples through two consecutive steps of a simple random walk on $G^{(k-1)}$ to the estimate k-node graphlet concentration. One drawback of $PSRW$ is its inefficiency of choosing neighbors during the random walk. For example, $PSRW$ performs the random walk on $G^{(3)}$ to estimate 4-node graphlet concentration. Populating neighbors of nodes in $G^{(3)}$ is about *an order of magnitude* slower than choosing random neighbors of nodes in $G^{(2)}$. If one can figure out how to estimate 4-node graphlet concentration with random walks on $G^{(2)}$, the time cost can be reduced dramatically. Furthermore, since $PSRW$ is more accurate than

the simple random walk on $G^{(k)}$ (SRW) when estimating k-node graphlet concentration, we have reasons to believe that random walks on $G^{(d)}$ with smaller d have the potential to achieve higher accuracy. Faster random walks and more accurate estimation motivate us to propose more efficient sampling methods based on random walks on $G^{(d)}$ to estimate k-node graphlet concentration. Different from $PSRW$, we seek for d that is smaller than $k-1$.

1.2 Our Contributions

Novel framework. In this paper, we propose a novel framework to estimate the graphlet concentration. Our framework is *provably correct* and makes no assumption on the graph structures. The main idea of our framework is to collect samples through consecutive steps of a random walk on $G^{(d)}$ to estimate k-node graphlet concentration, here d can be *any positive integer less than k*, and $PSRW$ is just a special case where $d = k-1$. We construct the subgraph relationship graph $G^{(d)}$ on the fly, and we do not need to know the topology of the original graph in advance. In fact, one can view d as a parameter of our framework. As mentioned in [35], *it is non-trivial to analyze and remove the sampling bias when randomly walking on $G^{(d)}$ where d is less than $k-1$*. The analysis method in $PSRW$ cannot be applied to the situation where $d < k-1$. Our work is not a simple extension of $PSRW$. More precisely, we propose a new and general framework which subsumes $PSRW$ as a special case. When choosing the appropriate parameter d, our methods significantly outperform the state-of-the-art methods.

Efficient optimization techniques. We also introduce two novel optimization techniques to further improve the efficiency of our framework. The first one, *corresponding state sampling (CSS)*, modifies the re-weight coefficient and improves the efficiency of our estimator. The second technique integrates the non-backtracking random walk in our framework. Simulation results show that our optimization techniques can improve the estimation accuracy.

Provable guarantees. We give detailed theoretical analysis on our unbiased estimators. Specifically, we derive an analytic Chernoff-Hoeffding bound on the sample size. The theoretical bound guarantees the convergence of our methods and provides insight on the factors which affect the performance of our framework.

Extensive experimental evaluation. To further validate our framework, we conduct extensive experiments on real-world networks. In Section 6, we demonstrate that our framework with an appropriate chosen parameter d is more accurate than the state-of-the-art methods. For 3-node graphlets, our method with the random walk on G outperforms $PSRW$ up to 3.8× in accuracy. For 4, 5-node graphlets, our methods outperform $PSRW$ up to 10× in accuracy and 100× in time cost.

2. PRELIMINARY

2.1 Notations and Definitions

Networks can be modeled as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. For a node $v \in V$, d_v denotes the degree of node v, i.e., the number of neighbors of node v. A graph with neither self-loops nor multiple edges is defined as a simple graph. In this work, we consider *simple*, *connected* and *undirected* graphs.

Induced subgraph. A k-node induced subgraph is a subgraph $G_k = (V_k, E_k)$ which has k nodes in V together with any edges whose both endpoints are in V_k. Formally, we have $V_k \subset V$, $|V_k| = k$ and $E_k = \{(u, v) : u, v \in V_k \land (u, v) \in E\}$.

Subgraph relationship graph. In [6, 35], the authors proposed the concept of subgraph relationship graph. Here we adopt the definition in [35] and define the d-node subgraph relationship graph $G^{(d)}$ as follows. Let $H^{(d)}$ denote the set of all d-node *connected induced* subgraphs of G. For $s_i, s_j \in H^{(d)}$, there is an edge between s_i and s_j if and only if they share $d-1$ common nodes in G. We use $R^{(d)}$ to denote the set of edges among all elements in $H^{(d)}$. Then we define $G^{(d)} = (H^{(d)}, R^{(d)})$. Specially, we define $G^{(1)} = G, H^{(1)} = V, R^{(1)} = E$ when $d = 1$. If the original graph G is connected, then $G^{(d)}$ is also connected [35, Theorem 3.1]. Figure 1 shows an example of $G^{(2)}$ and $G^{(3)}$ for a 4-node graph G. Let $H^{(2)}$ denote all 2-node induced subgraphs of G, then the node set of $G^{(2)}$ is $H^{(2)}$, i.e., node pairs $\{(1,2), (1,3), (1,4), (2,3), (3,4)\}$. Note that there is an edge between node pair $(1, 2)$ and $(2, 3)$ in $G^{(2)}$ because they share node 2 in G.
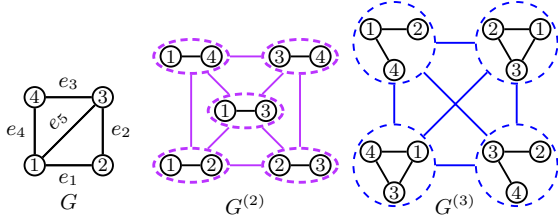
Figure 1: Original graph G and its 2 and 3-node subgraph relationship graph $G^{(2)}$ and $G^{(3)}$.

In general, constructing $G^{(d)}$ is impractical due to intensive computation cost. However, for our random walk-based framework, there is *no need* to construct $G^{(d)}$ in advance since we can generate the neighborhood subgraphs of $s \in H^{(d)}$ on the fly according to the definition of $G^{(d)}$.

Isomorphic. Two graphs $G = (V, E)$ and $G' = (V', E')$ are isomorphic if there exists a bijection $\varphi : V \rightarrow V'$ with $(v_i, v_j) \in E \Leftrightarrow (\varphi(v_i), \varphi(v_j)) \in E'$ for all $v_i, v_j \in V$. Such a bijection map is called an *isomorphism*, and we write isomorphic G and G' as $G \simeq G'$.

Definition 1. Graphlets are formally defined as connected, *non-isomorphic*, *induced subgraphs* of a graph G.

Figure 2 shows all $3, 4$-node graphlets. There are 2 different 3-node graphlets and 6 different 4-node graphlets. The second row of Table 3 shows 21 different 5-node graphlets. The number of distinct graphlets grows exponentially with the number of vertices in the graphlets. For example, there are 112 different 6-node graphlets and 853 different 7-node graphlets. Due to the combinatorial complexity, the computation for graphlets is usually restricted to 3, 4, 5 nodes [14, 37, 3, 35, 6, 36, 13]. Note that various applications, e.g., [32, 34], focus on graphlets with less than 6 nodes since graphlets with up to 5 nodes have the best cost-benefit trade off [6].

Problem definition. Given an undirected connected graph G and all the distinct k-node graphlets $\mathcal{G}^k = \{g_1^k, g_2^k, \cdots, g_m^k\}$, where g_i^k is the ith type of k-node graphlets. Let C_i^k denote the number of induced subgraphs that are isomorphic to
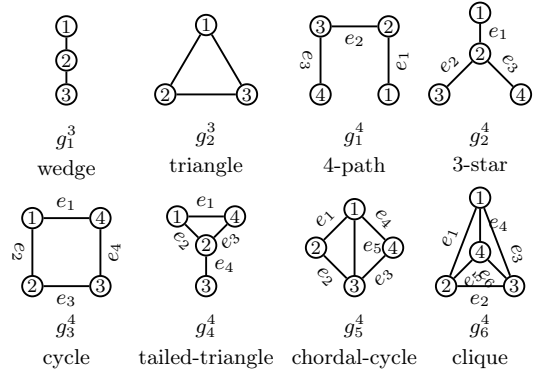
Figure 2: All $3, 4$-node distinct graphlets.

graphlet g_i^k. Our goal is to estimate the concentration of $g_i^k \in \mathcal{G}^k$ for G, where the concentration of g_i^k is defined as

$$c_i^k \triangleq \frac{C_i^k}{\sum_{j=1}^m C_j^k}. \tag{1}$$

Applications. In the following, we list some important applications involving graphlet concentration.

- *Clustering coefficient.* Friends of friends tend to become friends themselves in OSNs. This property is referred to as "transitivity". Clustering coefficient, which is defined as $3C_2^3/(C_1^3 + 3C_2^3) = 3c_2^3/(2c_2^3 + 1)$, quantifies the transitivity of networks, i.e., the probability that two neighbors of any vertex are connected. The clustering coefficient is important to understand the networks and can be obtained directly with the triangle concentration c_2^3.

- *Large graph comparison and classification.* One can use the graphlet concentration as the fingerprint for graph comparison [3]. The $3, 4, 5$-node graphlet concentration was proposed as the features for large graph classification in [32]. The intuition of using the graphlet concentration instead of the graphlet counts is that the differences in sizes of graphs skew the graphlet counts greatly and may result in poor performance of graph classification.

- *Intrinsic properties analysis.* Graphlet concentration can be used to understand the intrinsic properties of the networks. For example, Ahmed et al. [3] computed the 4-node graphlet concentration of the OSN Friendster and found that Friendster is lack of community related graphlets (e.g., cliques), which indicates the collapse of Friendster. Wang et al. in [35] used the concentration of directed 3-node graphlets to analyze the differences in functions of two OSNs, i.e., Douban and Sinaweibo.

Computing graphlet concentration is by no way an easier task than computing graphlet counts. Actually, graphlet counts are just reflections of the sizes of graphs. In Section 3.3 we show that the graphlet counts can be reconstructed easily if we have access to the whole graph data.

2.2 Random Walk and Markov Chain

Our framework is to generate samples from random walks. A *simple random walk* (SRW) over graph G is defined as follows: start from an initial node v_0 in G, we move to one of its neighbors which is chosen uniformly at random, and repeat this process until certain stopping criteria. The random walk on graph G can be viewed as a *finite* and *time-reversible* Markov chain with state space V. More specifically, let $\{X_t\}$

be the Markov chain representing the visited nodes of the random walk on G, the transition probability matrix $\mathbf{P}$ of this Markov chain is defined as

$$P(i,j) = \begin{cases} \frac{1}{d_i}, & \text{if } (i,j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

The SRW has a *unique* stationary distribution $\boldsymbol{\pi}$ where $\pi(v) = \frac{d_v}{2|E|}$ for $v \in V$ [20, 26]. The stationary distribution is vital for correcting bias of samples generated by random walks.
Strong Law of Large Numbers. Below, we review the *Strong Law of Large Numbers* (SLLN) for Markov chain which serves as the theoretical foundation for our random walk-based framework. For a Markov chain with finite state space $\mathcal{S}$ and stationary distribution $\boldsymbol{\pi}$, we define the expectation of the function $f : \mathcal{S} \to \mathbb{R}$ with respect to $\boldsymbol{\pi}$ as

$$\mu = \mathbb{E}_{\boldsymbol{\pi}}[f] \triangleq \sum_{X \in \mathcal{S}} f(X)\pi(X).$$

The subscript $\boldsymbol{\pi}$ indicates that the expectation is calculated with the assumption that $X \sim \boldsymbol{\pi}$. Let

$$\hat{\mu}_n = \frac{1}{n} \sum_{s=1}^{n} f(X_s)$$

denote the sample average of $f(X)$ over a run of the Markov chain. The following theorem SLLN ensures that the sample mean converges almost surely to its expected value.

THEOREM 1. *[10, 17] For a finite and irreducible Markov chain with stationary distribution $\boldsymbol{\pi}$, we have*

$$\hat{\mu}_n \to \mathbb{E}_{\boldsymbol{\pi}}[f] \text{ almost surely (a.s.)}$$

as $n \to \infty$ regardless of the initial distribution of the chain.

The SLLN is the fundamental basis for most random walk-based graph sampling methods, or more formally, Markov Chain Monte Carlo (MCMC) samplers, e.g., [19, 6, 12]. The SLLN guarantees the asymptotic unbiasedness of the estimators based on any finite and irreducible Markov chain.
Mixing Time. The mixing time of a Markov chain is the number of steps it takes for a random walk to approach its stationary distribution. We adopt the definition of mixing time in [24, 25, 8]. The mixing time is defined as follows.

Definition 2. The mixing time $\tau(\epsilon)$ (parameterized by ϵ) of a Markov chain is defined as

$$\tau(\epsilon) = \max_{X_i \in \mathcal{S}} \min\{t : |\boldsymbol{\pi} - \mathbf{P}^t \boldsymbol{\pi}^{(i)}|_1 < \epsilon\},$$

where $\boldsymbol{\pi}$ is the stationary distribution of the Markov chain, $\boldsymbol{\pi}^{(i)}$ is the initial distribution when starting from state $X_i \in \mathcal{S}$, $\mathbf{P}^t$ is the transition matrix after t steps and $|\cdot|$ is the variation distance between two distributions[1].

Later on, we will use the mixing time based Chernoff-Hoeffding bound [8] to compute the needed sample size to guarantee that our estimate is within $(1 \pm \epsilon)$ of the true value with probability at least $1 - \delta$.

---

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

Table 1: Summary of Notations

G	$G = (V, E)$, underlying undirected graph
$G^{(d)}$	$G^{(d)} = (H^{(d)}, R^{(d)})$, d-node subgraph relationship graph
g_i^k	the ith type of k-node graphlets
C_i^k	number of subgraphs isomorphic to g_i^k in graph G
c_i^k	concentration of graphlet g_i^k in G
d_v	degree of node v in G
d_X	degree of state X, X is also a node in $G^{(d)}$
$\mathcal{M}^{(l)}$	state space of the expanded Markov chain
$X^{(l)}$	state in state space $\mathcal{M}^{(l)}$
$V(X^{(l)})$	set of graph G's nodes contained in state $X^{(l)}$
$s(X^{(l)})$	subgraph induced by node set $V(X^{(l)})$
$\boldsymbol{\pi}$	stationary distribution of the random walk
$\boldsymbol{\pi}_{\mathbf{e}}$	stationary distribution of the expanded Markov chain
α_i^k	number of states $X^{(l)}$ in $\mathcal{M}^{(l)}$ s.t. $s \simeq g_i^k$, $s(X^{(l)}) = s$.
$\mathcal{C}(s)$	set of states $X^{(l)}$ in $\mathcal{M}^{(l)}$ s.t. $s(X^{(l)}) = s$.

3. GENERAL FRAMEWORK

In this section, we introduce our random walk-based general framework for graphlet concentration estimation. For ease of presentation, we summary the notations in Table 1.

3.1 Basic Idea

We are interested in finding the concentration of g_i^k, where $k \geq 3$ (since it is not difficult to find for $k = 1$ or 2). The core idea is to collect k-node graphlet samples through $l = k - d + 1$ consecutive steps of the random walk on $G^{(d)}$, where $d \in \{1, \ldots, k-1\}$. Specifically, for each state X_i ($i \geq l$) of the random walk on $G^{(d)}$, suppose we keep the history of previously visited $l - 1$ states, then we consider the subgraph induced by nodes contained in $X^{(l)} = (X_{i-l+1}, \ldots, X_i)$ as a k-node graphlet sample. Note that we only consider consecutive l steps which visit k distinct nodes in V. If the consecutive l steps on $G^{(d)}$ fail to collect k distinct nodes in V, we just continue the random walk until we find l consecutive steps which contain k distinct nodes in V.
Example: Consider the graph in Figure 1. (a) Suppose we want to get 3-node graphlet samples and choose a random walk on G. Then $l = 3 - 1 + 1 = 3$, i.e., we need to walk for 3 steps on G to get the 3-node graphlet samples. Assume the random walk sequence is $1 \to 2 \to 1 \to 4 \to 3$. Then we can obtain two 3-node graphlet samples induced by $\{2, 1, 4\}$ and $\{1, 4, 3\}$ respectively. The sequence $1 \to 2 \to 1$ is discarded since it only visits two distinct nodes. (b) If we want to explore possible graphlets g_i^4 and have decided to perform a random walk on $G^{(2)}$ (i.e., $d = 2$). Assume we make $l = 3$ transitions on the following states: $(1, 2) \to (1, 3) \to (3, 4)$, then we can obtain a 4-node graphlet sample induced by the node set $\{1, 2, 3, 4\}$, because $\{1, 2, 3, 4\}$ is contained in the three states $(1, 2), (1, 3)$ and $(3, 4)$. In this case, the obtained graphlet sample corresponds to g_5^4 in Figure 2.

The main technical challenge is to remove the bias of the obtained graphlet samples. To analyze the bias theoretically, we first introduce the concept of *expanded Markov chain*.

3.2 Expanded Markov Chain

We define an *expanded Markov chain* which remembers consecutive l steps of the random walk on $G^{(d)} = (H^{(d)}, R^{(d)})$. Each l consecutive steps are considered as a state $X^{(l)} = (X_1, \cdots, X_l)$ of the expanded Markov chain. Here we use

the superscript "l" to denote the length of the random walk block. Each time the random walk on $G^{(d)}$ makes a transition, the expanded Markov chain transits to the next state. Assume the expanded Markov chain is currently at state $X_i^{(l)} = (X_1, \cdots, X_l)$, it means the random walker is at X_l. If the walker jumps to $X_{(l+1)}$, i.e., one of the neighbors of X_l, then the expanded Markov chain transits to the state $X_{i+1}^{(l)} = (X_2, \cdots, X_{l+1})$. Let $\mathcal{N} = H^{(d)}$ denote the state space for the random walk and $\mathcal{M}^{(l)}$ denote the state space for the corresponding expanded Markov chain. The state space $\mathcal{M}^{(l)}$ consists of all possible consecutive l steps of the random walk. More formally, $\mathcal{M}^{(l)} = \{(X_1, \cdots, X_l) : X_i \in \mathcal{N}, 1 \le i \le l \ s.t. \ (X_i, X_{i+1}) \in R^{(d)} \ \forall 1 \le i \le l-1\} \subseteq \mathcal{N} \times \cdots \times \mathcal{N}$. For example, if we perform a random walk on G, any (u, v) and (v, u) where $e_{uv} \in E$ are states in $\mathcal{M}^{(2)}$. Note that the expanded Markov chain describes the same process as the random walk. The reason we define it here is for the convenience of deriving unbiased estimator.

The bias caused by the non-uniform sampling probabilities of the graphlet samples arises from two aspects. First, the states in the expanded Markov chain have unequal stationary probabilities. Second, a graphlet sample corresponds to several states in the expanded Markov chain. To derive the unbiased estimator of the graphlet concentration, we need to compute the stationary distribution and the number of states corresponding to the same graphlet sample.
Stationary distribution. Let $\boldsymbol{\pi}$ denote the stationary distribution of the random walk and $\boldsymbol{\pi_e}$ denote the stationary distribution of the expanded Markov chain. d_X is number of neighbors of state X in $G^{(d)}$. The following theorem states that $\boldsymbol{\pi_e}$ is unique and its value can be computed with $\boldsymbol{\pi}$.

THEOREM 2. *The stationary distribution $\boldsymbol{\pi_e}$ exists and is unique. For any $X^{(l)} = (X_1, \cdots, X_l) \in \mathcal{M}^{(l)}$, we have*

$$\pi_e(X^{(l)}) = \begin{cases} \frac{d_{X_l}}{2|R^{(d)}|} & if \ l = 1, \\ \frac{1}{2|R^{(d)}|} & if \ l = 2, \\ \frac{1}{2|R^{(d)}|} \frac{1}{d_{X_2}} \cdots \frac{1}{d_{X_{l-1}}} & if \ l > 2 \end{cases} \quad \textbf{(2)}$$

The proof is in the technical report [7]. In fact, $\boldsymbol{\pi_e}$ can be derived directly using conditional probability formula. For the state $X^{(l)} = (X_1, \cdots, X_l)$, X_1 is visited with probability $\pi(X_1) = \frac{d_{X_1}}{2|R^{(d)}|}$ during the random walk and $\pi_e(X^{(l)})$ can be written as $\frac{d_{X_1}}{2|R^{(d)}|} \times \frac{1}{d_{X_1}} \times \cdots \times \frac{1}{d_{X_l}}$.
Example: Still using the graph in Figure 1 as an example, if we walk on $G^{(2)}$ and visit states $X_1 = (1, 2), X_2 = (1, 3), X_3 = (3, 4)$, then the corresponding state $X_1^{(3)}$ in $\mathcal{M}^{(3)}$ is (X_1, X_2, X_3). The number of edges in $G^{(2)}$ is 8. The degrees of X_1, X_2, X_3 are 3, 4, 3 respectively. Then the stationary distribution of $X_1^{(3)}$ is $1/16 \cdot 1/4 = 1/64$.
State corresponding coefficient. Let $V(X^{(l)})$ represent the set of graph G's nodes contained in the state $X^{(l)}$, and $s(X^{(l)})$ be the subgraph induced by $V(X^{(l)})$. We define $X^{(l)}$ as the *corresponding state* for the subgraph $s(X^{(l)})$. A key observation is that a subgraph may have several corresponding states in $\mathcal{M}^{(l)}$. For example, if we perform a random walk on G, then a triangle induced by $\{u, v, w\}$ in G has 6 corresponding states (u, v, w), (u, w, v), (v, u, w), (w, v, u), (v, w, u), (w, u, v) in $\mathcal{M}^{(3)}$. To describe this idea formally, we define the *state corresponding coefficient* α_i^k.

Table 2: Coefficient α_i^k for $k = 3, 4$ nodes graphlets.

Graphlet		g_1^3	g_2^3	g_1^4	g_2^4	g_3^4	g_4^4	g_5^4	g_6^5
$\alpha_i^k/2$	$SRW(1)$	1	3	1	**0**	4	2	6	12
	$SRW(2)$	1	3	1	3	4	5	12	24
	$SRW(3)$	1/2	1/2	1	3	6	3	6	6

Definition 3. For any connected induced subgraph $s \simeq g_i^k$, we define the set of corresponding states for s as $\mathcal{C}(s) = \{X^{(l)}|s(X^{(l)}) = s, X^{(l)} \in \mathcal{M}^{(l)}\}$ and state corresponding coefficient α_i^k as $|\mathcal{C}(s)|$.

Coefficient α_i^k is vital to the design of unbiased estimator. If states in $\mathcal{M}^{(l)}$ are uniformly sampled, then the probability of getting a subgraph isomorphic to g_i^k is $\alpha_i^k C_i^k/|\mathcal{M}^{(l)}|$. The physical interpretation of α_i^k is that each subgraph isomorphic to g_i^k is *replicated* α_i^k times in the state space. If α_i^k is larger, we have a higher chance to get samples of g_i^k. Note that coefficient α_i^k only depends on different graphlets and random walk types. Hence we can compute α_i^k in advance.

In fact, α_i^k denotes how many ways we can traverse the g_i^k. Theoretically, α_i^k equals to *twice* of the number of Hamilton paths[2] (each Hamilton path is counted from both directions) in the subgraph relationship graph of g_i^k. The detailed computation process of α_i^k is in the technical report [7]. Table 2 lists the coefficient α_i^k for $k = 3, 4$. $SRW(d)$ in the table represents a random walk on $G^{(d)}$. Notice that $\alpha_2^4 = 0$ if we choose $SRW(1)$, i.e., if we walk on G, there is no chance to get samples of g_2^4. In this case, we can only estimate the relative concentration for all 4-node graphlets except g_2^4 (3-star). Table 3 lists the coefficient for all 5-node graphlets.

3.3 Unbiased Estimator

Now we derive an unbiased estimator for the graphlet concentration. Define an indicator function $h_i^k(X^{(l)})$ as

$$h_i^k(X^{(l)}) = \mathbb{1}\{s(X^{(l)}) \simeq g_i^k\}. \quad \textbf{(3)}$$

Function $h_i^k(X^{(l)}) = 0$ if the number of distinct nodes in $X^{(l)}$ is less than k or $s(X^{(l)})$ (subgraph induced by G's nodes in $X^{(l)}$) is not isomorphic to g_i^k. Since each subgraph isomorphic to g_i^k is *replicated* α_i^k times in the state space, we have

$$\sum_{X^{(l)} \in \mathcal{M}^{(l)}} h_i^k(X^{(l)}) = \alpha_i^k C_i^k.$$

Using above fact, we have

$$\mathbb{E}_{\boldsymbol{\pi_e}} \left[\frac{h_i^k(X^{(l)})}{\pi_e(X^{(l)})} \right] = \sum_{X^{(l)} \in \mathcal{M}^{(l)}} \frac{h_i^k(X^{(l)})}{\pi_e(X^{(l)})} \pi_e(X^{(l)}) = \alpha_i^k C_i^k.$$

Suppose there are n samples $\{X_s^{(l)}\}_{s=1}^n$ obtained from the expanded Markov chain. Combining the SLLN in Section 2.2 and above equation, we have

$$\hat{\mu}_n \triangleq \frac{1}{n} \sum_{s=1}^n \frac{h_i^k(X_s^{(l)})}{\pi_e(X_s^{(l)})} \to \alpha_i^k C_i^k.$$

Hence, we estimate C_i^k as

$$\hat{C}_i^k \triangleq \frac{1}{n} \sum_{s=1}^n \frac{h_i^k(X_s^{(l)})}{\alpha_i^k \pi_e(X_s^{(l)})} \to C_i^k. \quad \textbf{(4)}$$

[2]Hamilton path is defined as a path that visits each vertices in the graph exactly once.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Shape																					
$\alpha_i^5/2$ — SRW(1)	1	**0**	**0**	1	2	**0**	5	2	2	4	4	6	7	6	6	10	14	18	24	36	60
SRW(2)	1	2	12	5	4	16	5	6	24	24	12	18	15	54	36	42	34	82	76	144	240
SRW(3)	1	5	24	8	5	24	5	16	30	24	16	63	26	63	30	43	63	63	90	90	90
SRW(4)	1	3	6	3	3	6	10	12	12	12	12	10	10	10	12	10	10	10	10	10	10

The bias of each graphlet sample induced by nodes in $X^{(l)}$ is corrected by dividing its "*inclusion probability*" $\alpha_i^k \pi_e(X^{(l)})$. Note that this is a special case of *importance sampling* [27, Chapter 9]. With the *graphlet count estimator* in Equation (4), we can derive the graphlet concentration estimator easily. Define $h^k(X^{(l)}) = \mathbb{1}\{|V(X^{(l)})| = k\}$ and $\alpha^k(X^{(l)}) = \sum_{i=1}^{|\mathcal{G}^k|} \alpha_i^k h_i^k(X^{(l)})$. The concentration c_i^k is estimated with the following formula:

$$\hat{c}_i^k \triangleq \frac{\sum_{s=1}^n h_i^k(X_s^{(l)})/\left(\alpha_i^k \pi_e(X_s^{(l)})\right)}{\sum_{s=1}^n h^k(X_s^{(l)})/\left(\alpha^k(X_s^{(l)})\pi_e(X_s^{(l)})\right)} \to c_i^k. \quad (5)$$

Remarks: The common denominator in $\pi_e(X^{(l)})$ is $2|R^{(d)}|$, which is usually unknown for graphs with restricted access. Fortunately, $|R^{(d)}|$ in the numerator and denominator of $\hat{c}_i^k$ cancels out. That means c_i^k can be estimated without knowing $|R^{(d)}|$. Local information (i.e., degree of nodes, adjacent relationship) collected along the random walk is enough for the estimation. If we replace $\pi_e(X^{(l)})$ with $\tilde{\pi}_e(X^{(l)}) \triangleq 2|R^{(d)}|\pi_e(X^{(l)})$ in Equation (5), the estimator $\hat{c}_i^k$ remains unchanged. For the state $X^{(l)} = (X_1, \cdots, X_l)$, $\tilde{\pi}_e(X^{(l)})$ can be computed directly with the degrees of $X_1, \cdots, X_l$.

Algorithm 1 depicts the process of k-node graphlet concentration estimation. Note that we can easily estimate the count of graphlets with Equation (4) if we know $|R^{(d)}|$. For random walk on G, we have $R^{(1)} = |E|$. For SRW on $G^{(2)}$, $|R^{(2)}| = \frac{1}{2}\sum_{e_{uv}}(d_u + d_v - 2)$ and a single pass of graph data is enough to compute this value.

Bound on sample size. The next important question we like to ask is what is the *smallest sample size* (or random walk steps) to guarantee high accuracy of our estimator? In the following, we show the relationship between accuracy and sample size using the Chernoff-Hoeffding bound for Markov chain. Let W denote $\max_{X^{(l)}} 1/\pi_e(X^{(l)})$ and $\alpha_{\min} = \min_i \alpha_i^k$. The following theorem states the relationship between the estimation accuracy and sample size.

THEOREM 3. *For any $0 < \delta < 1$, there exists a constant ξ such that*

$$\Pr\left[(1-\epsilon)c_i^k \leq \hat{c}_i^k \leq (1+\epsilon)c_i^k\right] > 1 - \delta \quad (6)$$

when the sample size $n \geq \xi(\frac{W}{\Lambda})\frac{\tau}{\epsilon^2}(\log\frac{\|\varphi\|_{\pi_e}}{\delta})$. Here $\Lambda = \min\{\alpha_i^k C_i^k, \alpha_{\min}C^k\}$, τ is the mixing time $\tau(1/8)$ of the original random walk, φ is the initial distribution and $\|\varphi\|_{\pi_e}$ is defined as $\sum_{X^{(l)}} \varphi^2(X^{(l)})/\pi_e(X^{(l)})$.

Remarks: The proof for Theorem 3 is in the technical report [7]. From Theorem 3, we know that the needed sample size is *linear* with the mixing time τ. This implies our framework performs better for graphs with smaller mixing time.

Algorithm 1 Unbiased Estimate of Graphlet Statistics

Input: sample budget n, SRW on $G^{(d)}$, graphlet size k
Output: estimate of $[c_1^k, \cdots, c_m^k]$ $(m = |\mathcal{G}^k|)$
1: random walk block length $l \leftarrow k - d + 1$
2: counter $\hat{C}_i^k \leftarrow 0$ for $i \in \{1, \cdots, m\}$
3: walk l steps to get the initial state $X^{(l)} = (X_1, \cdots, X_l)$
4: random walk step $t \leftarrow 0$
5: **while** $t < n$ **do**
6: $\quad i \leftarrow$ graphlet type id of subgraph $s(X^{(l)})$
7: $\quad \hat{C}_i^k \leftarrow \hat{C}_i^k + 1/\left(\alpha_i^k \tilde{\pi}_e(X^{(l)})\right)$
8: $\quad X_{l+t+1} \leftarrow$ uniformly chosen neighbor of X_{l+t}
9: $\quad X^{(l)} \leftarrow (X_{t+2}, \cdots, X_{l+t+1})$
10: $\quad t \leftarrow t + 1$
11: $\hat{c}_i^k = \hat{C}_i^k / \sum_{j=1}^m \hat{C}_j^k$ for all $i \in \{1, \cdots, m\}$
12: **return** $[\hat{c}_1^k, \cdots, \hat{c}_m^k]$

Furthermore, some graphlet types are relatively rare in the graphs. For these rare graphlet types, we need larger sample size to guarantee the same accuracy. If α_i^k is higher for rare graphlet g_i^k, the needed sample size is smaller.

4. IMPROVED ESTIMATION

We now introduce two novel optimization techniques to reduce the need sample size in Theorem 3. The first technique is to correct the bias by combining the stationary probabilities of the corresponding states. The second technique integrates the non-backtracking random walk into our sampling framework. With these two optimization techniques, we obtain a more efficient estimator.

4.1 Corresponding State Sampling (CSS)

Recall that $\mathcal{C}(s)$ is defined as the set of states which correspond to the subgraph s, i.e., states in $\mathcal{C}(s)$ contain the same set of nodes as subgraph s. The key observation is that for $X_a^{(l)}, X_b^{(l)} \in \mathcal{C}(s)$, the "*inclusion probabilities*" $\alpha_i^k \pi_e(X_a^{(l)})$ and $\alpha_i^k \pi_e(X_b^{(l)})$ may be different even though $X_a^{(l)}$ and $X_b^{(l)}$ correspond to the same subgraph. In other words, the inclusion probability of a subgraph is determined not only by the nodes in the subgraph, but also the orders in which these nodes are visited.

Example: To illustrate, consider a triangle Δ induced by nodes u, v, w. Suppose we choose a random walk on G. Then both of states $X_1^{(3)} = (u, v, w)$ and $X_2^{(3)} = (v, u, w)$ correspond to the triangle Δ. We know that $\alpha_2^3 \pi_e(X_1^{(3)}) = \frac{6}{2|E|}\frac{1}{d_v}$ while $\alpha_2^3 \pi_e(X_2^{(3)}) = \frac{6}{2|E|}\frac{1}{d_u}$. If nodes u and v have different degrees, then $\alpha_2^3 \pi_e(X_1^{(3)}) \neq \alpha_2^3 \pi_e(X_2^{(3)})$, i.e., the same triangle Δ has different inclusion probabilities when visited in orders u, v, w and v, u, w.

Based on this observation[3], we define "*sampling probability*" $p(X^{(l)})$ for the subgraph induced by nodes in $X^{(l)}$. The value of $p(X^{(l)})$ only depends degrees of nodes in $X^{(l)}$.

Definition 4. For a state $X^{(l)}$ and a subgraph $s = s(X^{(l)})$, we define the sampling probability for s as

$$p(X^{(l)}) \triangleq \sum_{X_j^{(l)} \in \mathcal{C}(s)} \pi_e(X_j^{(l)}).$$

In the following, we prove that if we substitute $\alpha_i^k \pi_e(X^{(l)})$ with $p(X^{(l)})$ in Equation (4), we still obtain an unbiased estimator of C_i^k.

LEMMA 4. *For a specific subgraph $s \simeq g_i^k$, we have*

$$\mathbb{E}_{\boldsymbol{\pi_e}}\big[\frac{1}{\alpha_i^k \pi_e X^{(l)}} \mathbb{1}\{V(X^{(l)}) = V(s)\}\big] =$$
$$\mathbb{E}_{\boldsymbol{\pi_e}}\big[\frac{1}{p(X^{(l)})} \mathbb{1}\{V(X^{(l)}) = V(s)\}\big]$$

Lemma 4 can be proved directly using the definition. It is trivial to verify that the function $h_i^k(X^{(l)})$ in Equation (3) is the linear combination of function $\mathbb{1}\{V(X^{(l)}) = V(s)\}$. Using the linearity of expectation and the result in Lemma 4, we have

$$\mathbb{E}_{\boldsymbol{\pi_e}}\left[\frac{h_i^k(X^{(l)})}{p(X^{(l)})}\right] = \mathbb{E}_{\boldsymbol{\pi_e}}\left[\frac{h_i^k(X^{(l)})}{\alpha_i^k \pi_e(X^{(l)})}\right].$$

Hence, we can rewrite the estimator in Equation (4) as

$$\frac{1}{n} \sum_{s=1}^{n} \frac{h_i^k(X_s^{(l)})}{p(X_s^{(l)})} \to C_i^k \ a.s. \tag{7}$$

Similarly, we estimate graphlet concentration as

$$\frac{\sum_{s=1}^{n} h_i^k(X_s^{(l)})/p(X_s^{(l)})}{\sum_{s=1}^{n} h^k(X_s^{(l)})/p(X_s^{(l)})} \to c_i^k \ a.s. \tag{8}$$

Remarks: The pseudo code of computing $p(X^{(l)})$ is presented in the technical report [7]. The estimator in Equation (7) corrects the bias of graphlet samples using the sampling probability $p(X^{(l)})$ instead of $\alpha_i^k \pi_e(X^{(l)})$. It indicates that the probability that a subgraph s is generated by the random walk actually equals to $\sum_{X_j^{(l)} \in \mathcal{C}(s)} \pi_e(X_j^{(l)})$.

Example: Table 4 lists the corresponding $p(X^{(l)})$ when we choose $SRW(1)$ for 3-node graphlets and $SRW(2)$ for 4-node graphlets. Labels for nodes and edges are defined in Figure 2. Note that an edge e_{uv} in the graph G is a node in $G^{(2)}$. The degree of e_{uv} in $G^{(2)}$ should be $d_u + d_v - 2$, i.e., $d_{e_{uv}} = d_u + d_v - 2$. Here d_u and d_v are degrees of nodes u and v in G. In Table 4, the first column is the graphlet type of subgraph induced by nodes in $X^{(l)}$. The second column is the random walk type. The third column is the state corresponding coefficient and the fourth column is the sampling probability $p(X^{(l)})$. To further understand the sampling probability, we give an example of triangle (g_2^3). If we randomly walk on G and visit nodes $1, 2, 3$ sequentially, then the state we are visiting is $X^{(3)} = (1, 2, 3)$. Assume

---

[3]Here we require $l > 2$ since when $l = 2$, the inclusion probabilities are the same for the states corresponding to the same subgraph.

$\{1, 2, 3\}$ induces a triangle Δ. We know that the corresponding states of Δ are $(1, 2, 3)$, $(3, 2, 1)$, $(1, 3, 2)$, $(2, 3, 1)$, $(2, 1, 3)$ and $(3, 1, 2)$. The sampling probability $p(X^{(l)})$ for $X^{(3)} = (1, 2, 3)$ is $\frac{1}{2|E|}(2/d_1 + 2/d_2 + 2/d_3)$ while $\alpha_i^k \pi_e(X^{(l)})$ for $X^{(3)}$ is $\frac{6}{2|E|} \frac{1}{d_2}$. Observe that $p(X^{(l)})$ *makes better use of the degree information for the nodes in the subgraph.*

Table 4: Sampling probability $p(X^{(l)})$ for all $3, 4$-node graphlets.

Graphlet	SRW(d)	$\alpha_i^k/2$	$2\lvert R^{(d)}\rvert \cdot p(X^{(l)})/2$
g_1^3	$SRW(1)$	1	$1/d_2$
g_2^3		3	$1/d_1 + 1/d_2 + 1/d_3$
g_1^4		1	$1/d_{e_2}$
g_2^4		3	$\sum_{j=1}^{3} 1/d_{e_j}$
g_3^4		4	$\sum_{j=1}^{4} 1/d_{e_j}$
g_4^4	$SRW(2)$	5	$2/d_{e_2} + 2/d_{e_3} + 1/d_{e_4}$
g_5^4		12	$2\sum_{j=1}^{5} 1/d_{e_j} + 2/d_{e_5}$
g_6^4		24	$4\sum_{j=1}^{6} 1/d_{e_j}$

Bound on sample size. Define $W' \triangleq \max_{X^{(l)}} 1/p(X^{(l)})$. When the sample size $n \geq \xi(\frac{W'}{C_i^k})\frac{\tau}{\epsilon^2}(\log\frac{\|\varphi\|_{\boldsymbol{\pi_e}}}{\delta})$, the estimate in Equation (8) is within $(1 \pm \epsilon)c_i^k$ with probability at least $1 - \delta$. Here, ξ is a constant independent of ϵ, δ. τ is the mixing time $\tau(1/8)$ of the random walk. φ is the initial distribution. Since $\max 1/p(X^{(l)}) \leq \max 1/\alpha_i^k \pi_e(X^{(l)})$, the bound on the sample size for the new estimator in Equation (8) is smaller.

4.2 Non-backtracking Random Walk

Many techniques have been proposed to improve the efficiency of random walk-based algorithms, for example, non-backtracking random walk [17], random walk leveraging walk history [39], rejection controlled Metropolis-Hasting random walk [19], random walk with jump [38], etc. In this subsection, we introduce non-backtracking random walk (NB-SRW) to our estimation framework *as an example to show how to integrate these techniques with our framework.*

The basic idea of NB-SRW is to avoid backtracking to the previously visited node. Due to the dependency on previously visited node, the random walk on $G^{(d)} = (H^{(d)}, R^{(d)})$ is not a Markov chain on state space $H^{(d)}$. However, we can define an augmented state space $\Omega = \{(i, j) : i, j \in H^{(d)}, \ s.t. \ (i, j) \in R^{(d)}\} \subseteq H^{(d)} \times H^{(d)}$. The transition matrix $\mathbf{P}' \triangleq \{P'(e_{ij}, e_{lk})\}_{e_{ij}, e_{lk} \in \Omega}$ for the NB-SRW is defined as follows

$$P'(e_{ij}, e_{jk}) = \begin{cases} \frac{1}{d_j - 1}, & \text{if } i \neq k \text{ and } d(j) \geq 2, \\ 0, & \text{if } i = k \text{ and } d(j) \geq 2, \\ 1, & \text{if } i = k \text{ and } d(j) = 1. \end{cases}$$

All other elements of matrix $\mathbf{P}$ are zeros. Let $\boldsymbol{\pi}'$ be the stationary distribution of the NB-SRW. A useful fact is that NB-SRW preserves the stationary distribution of the original random walk, i.e., $\pi'(i) = d_i/2\lvert R^{(d)}\rvert$ and $\pi'(e_{ij}) = 1/2\lvert R^{(d)}\rvert$. To apply NB-SRW, we just need to replace our previously used simple random walk with NB-SRW. The estimator in Equation (5) and (8) can still be used except that we need to replace the $\pi_e(X^{(l)})$ with $\pi'_e(X^{(l)})$. Define the nominal degree for $X_i \in H^{(d)}$ as $d'_{X_i} = \max\{d_{X_i} - 1, 1\}$. For any $X^{(l)} = (X_1, \cdots, X_l)$, the $\pi'_e(X^{(l)})$ can be computed by substituting d_X with d'_X in Equation (2).

259

Applying NB-SRW helps us eliminate some "*invalid*" states from the state space. For example, if we want to estimate 3-node graphlet concentration using $SRW(1)$, we need to walk for 3 steps on G to collect a sample. It is possible for us to get only 2 distinct nodes from 3 steps. We call such samples as invalid samples. The invalid samples do not contribute to the estimation. If we apply NB-SRW here, it is less likely to get such invalid samples. Hence NB-SRW helps to improve the estimation efficiency of our framework.

5. IMPLEMENTATION DETAILS

We explain how to obtain neighbors of currently visited state (subgraph) $s \in H^{(d)}$ on the fly. Obtaining a uniformly chosen neighbor of a node in G or $G^{(2)}$ takes $O(1)$ time. We give details about the random walk on $G^{(2)}$. The set of neighbors of e_{uv} is $N(e_{uv}) = \{e_{uw} : w \in N(u)\backslash v\} \cup \{e_{vz} : z \in N(v)\backslash u\}$. Recall that $N(v)$ denotes the set of neighbors of $v \in V$. To ensure each neighbor of e_{uv} is chosen uniformly, we first select one of u and v with probability $d_u/(d_u + d_v)$ and $d_v/(d_u+d_v)$ respectively. Suppose we have chosen node u, we then randomly select a node $w \in N(u)$. If $w \neq v$, e_{uw} is proposed as the next step after e_{uv}. Otherwise, we restart the process until we obtain a neighbor of e_{uv}. Based on above discussion, we know that getting a uniformly sampled neighbor of a state in $G^{(2)}$ can be done in constant time.

To obtain a randomly chosen neighbor of s in $G^{(d)}$, we can replace one node v_i in $V(s)$ with a node $v_j \in \cup_{v \in V(s)\backslash v_i} N(v)$ and ensure the connectivity of this new subgraph induced by node set $\{v_j\} \cup V(s)\backslash v_i$. Here $V(s)$ is the node set of the subgraph s. However, to ensure the neighbors of s are uniformly sampled, we need to generate all neighbors of s when $d > 2$, which requires d merge operations over $d - 1$ adjacent lists of nodes in the currently visited subgraph. Hence, the time complexity of selecting a random neighbor for a state in $G^{(d)}$ is simply $O(d^2 \frac{|E|}{|V|})$ when $d > 2$.

According to Algorithm 1, we also need to identify the graphlet type of the obtained samples at each step. We refer interested readers to the implementation details and time complexity analysis in the technical report [7].

Note that our framework does not need any preprocessing of the graph data. The time complexity of our framework is $O(n)$ when $d \leq 2$ and $O(nd^2 \frac{|E|}{|V|})$ when $d > 2$ for the k-node graphlets. Here n is the random walk steps.

6. EXPERIMENTAL EVALUATION

We evaluate the performance of our framework on $3, 4, 5$-node graphlets. The algorithms are implemented in C++ and we run experiments on a Linux machine with Intel 3.70GHz CPU. We aim to answer the following questions.

Q1: How accurate is our framework? Do the optimization techniques really improve the accuracy?

Q2: How does the random walk on $G^{(d)}$ affect the performance? What is the best parameter d for $3, 4, 5$-node graphlets?

Q3: Does our framework provide more accurate estimation than the state-of-the-art methods?

6.1 Experiment Setup

We use publicly available real-world networks to evaluate our framework. We focus on undirected graphs by removing the directions of edges if the graphs are directed. We only retain the largest connected component (LCC) of the graphs and discard the rest nodes and edges. The detailed information about the LCC of the graphs is reported in Table 5.

The exact graphlet concentration is obtained through well-tuned enumeration methods [3, 13]. For 5-node graphlets, the ground-truth value is only computed for the four smaller datasets due to the extremely high computation cost. As an example, we state the exact concentration of the $3, 4, 5$-node cliques (i.e., c_2^3, c_6^4, and c_{21}^5) in Table 5, and we can see that the $3, 4, 5$-node cliques take a relatively low percentage.

Table 5: Datasets

Graph	$|V|$	$|E|$	c_2^3 (10^{-2})	c_6^4 (10^{-3})	c_{21}^5 (10^{-5})
BrightKite [1]	57K	213K	3.98	1.447	4.661
Epinion [18]	76K	406K	2.29	0.225	0.147
Slashdot [18]	77K	469K	0.82	0.092	0.115
Facebook [1]	63K	817K	5.46	1.419	2.511
Gowalla [18]	197K	950K	0.80	0.008	-
Wikipedia [1]	1.9M	36.5M	0.10	0.00009	-
Pokec [1]	1.6M	22.3M	1.6	0.035	-
Flickr [1]	2.2M	22.7M	3.87	0.886	-
Twitter [30]	21.3M	265M	0.86	0.0166	-
Sinaweibo [30]	58.7M	261M	0.03	0.00008	-

We use the following normalized root mean square error (NRMSE) to measure the estimation accuracy:

$$\text{NRMSE}(\hat{c}_i^k) \triangleq \frac{\sqrt{\mathbb{E}[(\hat{c}_i^k - c_i^k)^2]}}{c_i^k} = \frac{\sqrt{\text{Var}[\hat{c}_i^k] + (c_i^k - \mathbb{E}[\hat{c}_i^k])^2}}{c_i^k},$$

where $\hat{c}_i^k$ is the estimated value and c_i^k is the ground-truth. $\text{NRMSE}(c_i^k)$ is a combination of variance and bias of the estimate $\hat{c}_i^k$, both of which are important to characterize the accuracy of the estimator.

The names of the methods are given in the following way. SRWd represents random walks on $G^{(d)}$. If the method also integrates the optimization techniques *corresponding state sampling* (CSS) and *non-backtracking random walk* (NB-SRW), we append CSS and NB at the end of the method name, respectively. For example, SRW1CSSNB means that we perform the random walk on $G^{(1)}$ (i.e., G) and use both techniques *CSS* and *NB-SRW* for further optimization.

6.2 Framework Evaluation

6.2.1 Accuracy

Comparison between different random walks. We first demonstrate the effects of the parameter d and the optimization techniques on the estimation accuracy. The estimation results are presented in Figure 3 for all the graphs whose ground truth has been obtained. Note that only graphlets g_2^3, g_6^4, g_{21}^5 are presented since they have the smallest concentration value among $3, 4, 5$-node graphlets respectively and are observed to have the least accurate estimates. The NRMSE is estimated over 1,000 independent simulations except that the NRMSE of SRW4 is only estimated over 100 simulations since the random walk on $G^{(4)}$ is relatively slow. We do not study SRW1 for $4, 5$-node graphlets because $\alpha_2^4, \alpha_2^5, \alpha_3^5, \alpha_6^5$ are zeros with SRW1. The sample size, i.e., the random walk steps, equals to 20K for all methods in the framework. We summarize our findings as follows.

- The method SRW1CSSNB, i.e., random walk on G with optimization techniques CSS and NB-SRW, has the highest accuracy in estimating the concentration of 3-node

graphlets. The method SRW2CSS has the best performance in estimating 4, 5-node graphlet concentration.

- The best methods in our framework provide accurate estimates. The NRMSE of SRW1CSSNB for graphlet g_2^3 is in the range $0.025 \sim 0.13$. The NRMSE of SRW2CSS for graphlets g_6^4, g_{21}^5 is in the range $0.08 \sim 4.3$, and $0.20 \sim 0.86$ respectively. Note that we only use 20K random walk steps. The sample size is small compared with the graph size. E.g., we only exploit 0.03% nodes of Sinaweibo.
- For the same graphlets, the random walk on $G^{(d)}$ with smaller d outputs better estimates. E.g., SRW1 outputs estimates of c_2^3 which has $3.8\times$ smaller NRMSE than that of SRW2 for Twitter; SRW2 produces estimates of c_6^4 with $10\times$ smaller NRMSE than SRW3 for Gowalla. In conclusion, we should choose $d = \{1, 2, 2\}$ for 3, 4, 5-node graphlets respectively.
- The optimization technique CSS improves the accuracy of estimates a lot while the performance gain of NB-SRW is negligible. For example, SRW1CSS reduces the NRMSE of SRW1 more than 3 times for Wikipedia and Sinaweibo when estimating the triangle (g_2^3) concentration.

Weighted concentration and accuracy. We now introduce the concept of *weighted concentration* to explain how the parameter d affects the performance. From Equation (4), we know that $\frac{1}{n} \sum_{s=1}^{n} \frac{h_i^k(X_s^{(l)})}{\pi_e(X_s^{(l)})} \to \alpha_i^k C_i^k$. To further understand the performance of our framework, we define the *weighted concentration* for graphlet g_i^k as $\alpha_i^k C_i^k / (\sum_{j=1}^{m} \alpha_j^k C_j^k)$. As an example, we plot the weighted concentration of 4-node graphlets for Epinion in Figure 4a. From Figure 4, we know the parameter d and the concentration value affect the performance of our framework in the following way.

- **_Effect of the parameter_ d.** Compared with the original concentration, the weighted version lifts the percentage of relatively rare graphlets, i.e., g_3^4 (⬓), g_5^4 (⬔), and g_6^4 (⬚). For the graphs Epinion, the weighted concentration of SRW2 is much larger than that of SRW3 for graphlets g_5^4, g_6^4, while for graphlet g_3^4, the weighted concentration of SRW3 is slightly higher than that of SRW2. For example, the weighted concentration for g_6^4 with SRW2 is about $8\times$ higher than the original one while SRW3 only increases the concentration $2\times$ higher. In other words, SRW2 increases the probability of getting a sample of g_6^4 about $8\times$ higher compared with uniform sampling of graphlets, while SRW3 only increases the probability about $2\times$ higher. Consequently, the NRMSE of SRW2 in estimating c_6^4 is $2\times$ smaller than that of SRW3. From Theorem 3 we know that more samples are needed to achieve specific accuracy for graphlets with smaller $\alpha_i^k C_i^k$. Hence the error of the estimation for rare graphlets is the major error source. If we are able to assign rare graphlets higher weighted concentration, the overall performance is less likely to degenerate. *Based on above discussion, we conclude that random walks on $G^{(d)}$ with smaller d have better overall performance since they have a higher chance of getting the relatively rare graphlets.*
- **_Effect of the concentration value._** From Figure 4b we can see that SRW2 and SRW2CSS perform better than SRW3 for all the 4-node graphlets except g_3^4 (because the weighted concentration of g_3^4 computed with SRW3 is higher than that of SRW2). Besides, the smaller the concentration value, the higher the estimation error, which is consistent with our analysis in Theorem 3.

Table 6: Running time of performing 20K random walk steps for different methods.

Graph	SRW2	SRW2CSS	SRW3	SRW4	Exact
BrightKite [1]	19.4 ms	110.2 ms	271.1 ms	20.6 s	511 s
Epinion [18]	20.6 ms	68.6 ms	540.0 ms	51.4 s	11091 s
Slashdot [18]	19.6 ms	50.6 ms	538.8 ms	47.4 s	5702 s
Facebook [1]	21.8 ms	114.8 ms	214.2 ms	19.8 s	4405 s

6.2.2 Convergence

To show the convergence properties of the methods, we vary the sample size from 2K to 20K in increment of 1K. We present the simulation results in Figure 5 for 3, 4, 5-node cliques since they have the smallest concentration value and the least accuracy. Due to the space limitation, we only choose 6 representative graphs in the datasets for the presentation. We summarize the observations as follows.

- The estimates are more concentrated around the ground-truth as we increase the sample size.
- SRW1CSSNB exhibits consistent best performance in estimating c_2^3. SRW2CSS has consistent best performance in estimating 4, 5-node clique concentration.
- There are spikes in the line plot of NRMSE v.s. sample size. This is caused by burn-in period of the random walks and inadequate simulation times.

6.2.3 Running time

In Table 6, we show the average running time of performing 20K random walk steps for different methods when estimating 5-node graphlet concentration. Among them, Exact represents the enumeration time of [13]. Since SRW3CSS incurs high computation cost, we do not consider this algorithm. The time cost of the random walk consists of populating a random neighbor and identifying the graphlet types. Here we do not consider the APIs response delay. From Table 6, we know that the random walk on $G^{(d)}$ is faster when d is smaller. For example, SRW2 is much faster than SRW3 and SRW4. This also validates the choice of smaller d.

6.3 Comparison with Competing Methods

6.3.1 Methods with restricted access assumption

Our framework is mainly designed for graphs with restricted access, i.e., the graph data can only be accessed through APIs. For graphs with restricted access, random walk-based methods are commonly used to exploit the properties of the graphs. In this part, we compare the best methods in our framework with two state-of-the-art random walk-based methods [35] and [12].

The state-of-the-art random walk-based method in estimating any k-node graphlet concentration is the PSRW proposed in [35]. Note that PSRW is equivalent to choosing $d = k - 1$ in our framework. Specifically, PSRW corresponds to SRW2, SRW3, and SRW4 when estimating 3, 4 and 5-node graphlet concentration respectively. We recap the comparison results in Figure 3 and Table 6 as follows.

- For 3-node graphlets, SRW1CSSNB performs best for all the datasets in Table 5, and it outperforms PSRW up to $3.8\times$ (for "Twitter"). For 4, 5-node graphlets, SRW2CSS performs better than PSRW both in time cost and accuracy, e.g., SRW2CSS outperforms PSRW in estimating 4-node graphlet concentration *up to an order of magnitude* (for "Gowalla") in accuracy.
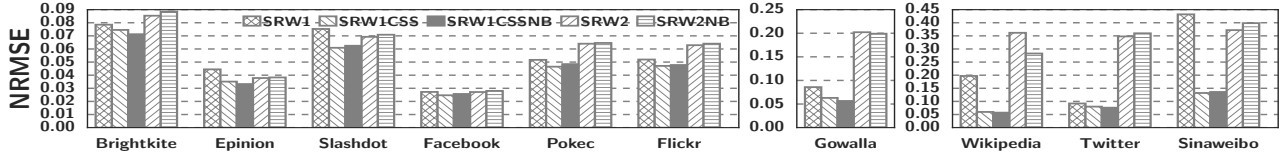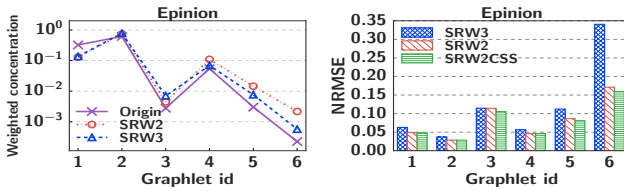
(a) Triangle (g_2^3, ⊾)
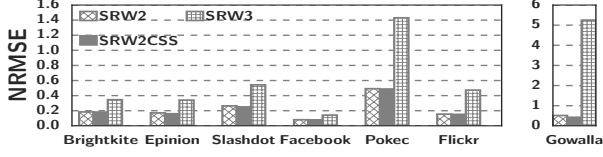
(b) 4-node clique (g_6^4, ⧖)
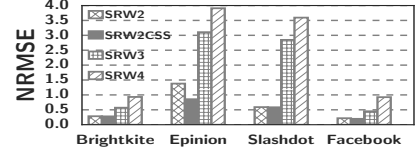
(c) 5-node clique (g_{21}^5, ✩)

Figure 3: NRMSE of concentration estimates (sample size = 20K).


(a) Weighted concentration

(b) NRMSE

Figure 4: Relationship between the weighted concentration and the accuracy (sample size = 20K).

The triangle concentration has strong relationship with the global clustering coefficient which is defined as $3C_2^3/(C_1^3 + 3C_2^3) = 3c_2^3/(2c_2^3+1)$. Similarly, c_2^3 can be computed directly with the clustering coefficient. Hence we also consider the method proposed by Hardiman et al. [12], which is primarily designed for clustering coefficient, as the competing method for 3-node graphlet concentration. The method in [12] uses the simple random walk on G, and at each step, the visited node v_t checks whether the node visited before v_t and the node visited after v_t are connected. *Detailed analysis of this method reveals that it is equivalent to SRW1 under our framework*, but derived in a totally different way. From Figure 3, we observe that our method SRW1CSSNB outperforms [12] (SRW1). Especially for Wikipedia and Sinaweibo, SRW1CSSNB has at least 3× smaller NRMSE.

6.3.2 Methods with full access assumption

Now we assume the graph data is readily available and fits in the main memory. We evaluate our framework in such full access setting with the purpose to shed light on the advantages and disadvantages of applying the MCMC samplers for memory-based graphs. We compare with two state-of-the-art methods: wedge sampling [31], and path sampling [14]. **Wedge sampling [31].** This method estimates the triadic measures (e.g., number of triangles) by generating uniform wedge (⊾) samples. To get a uniform wedge sample, it first selects a random node v according to the probability p_v, here $p_v \triangleq \binom{d_v}{2}/(\sum_{u \in V}\binom{d_u}{2})$, and then chooses a uniform random pair of neighbors of v to generate a wedge. Note that wedge sampling needs to compute the probability p_v for each node v. The time complexity of wedge sampling is $O(|V| + n\log|V|)$, where n is the number of wedge samples.
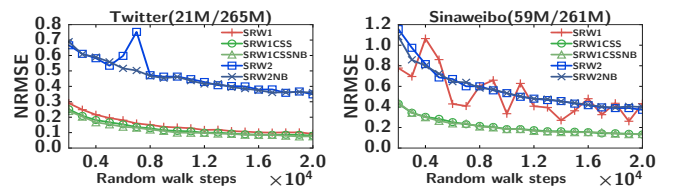
(a) Triangle (⊾)
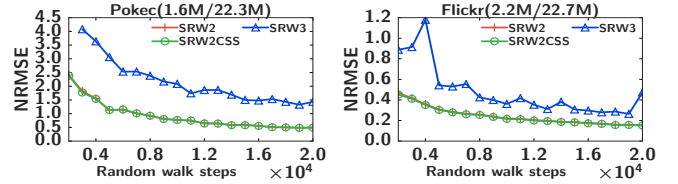
(b) 4-node clique (⧖)
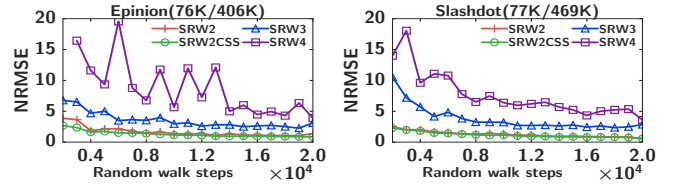
(c) 5-node clique (✩)

Figure 5: Convergence of the estimates.

Path sampling [14]. It estimates 4-node graphlet counts by generating uniform 3-path (⋀) samples. For each edge $e = (u,v) \in E$, denote $\tau_e = (d_u - 1)(d_v - 1)$ and $S \triangleq \sum_e \tau_e$. To get a uniform 3-path, it first selects an edge $e = (u,v)$ with probability $p_e \triangleq \tau_e/S$. Then picks uniform random neighbor u' of u other than v, picks uniform random neighbor v' of v other than u, and outputs the three edges $\{(u',u),(u,v),(v,v')\}$ as the sampled 3-path. The time complexity of path sampling to generate n 3-path samples is $O(|E| + n\log|E|)$.

Note that these two methods focus on estimating graphlet counts. Our framework is also capable of estimating graphlet counts for memory-based graphs according to Equation (4) and (7). We compare SRW1CSSNB with wedge sampling, and SRW2CSS with path sampling for 3, 4-node graphlet
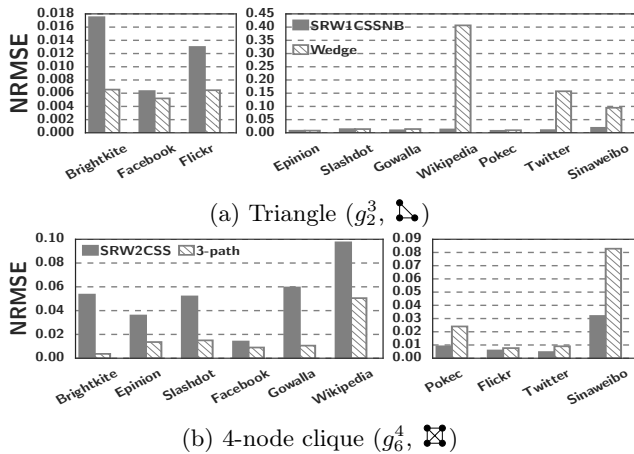
262

(a) Triangle (g_2^3, ⬡)

(b) 4-node clique (g_6^4, ⬡)

Figure 6: NRMSE of graphlet counts estimation.

counts estimation. For fair comparison, we run wedge sampling and path sampling for 200K samples (200K samples are used in the original papers), and then run the SRW1CSSNB and SRW2CSS for the same running time as wedge sampling and path sampling respectively. The results are shown in Figure 6. For path sampling, we only show the accuracy of 4-node clique due to space limitation. We summarize the results in Figure 6 as follows.

- In Figure 6a, we compare our method SRW1CSSNB with wedge sampling (denoted as Wedge in the figure). For graphs BrightKite, Facebook, and Flickr, Wedge is more accurate than SRW1CSSNB given the same running time. Actually, these three graphs have the highest triangle concentration in the datasets. SRW1CSSNB performs better than Wedge given the same running time for the rest graphs. Both of SRW1CSSNB and Wedge generate wedge samples. The difference is that Wedge generates independent wedge samples while SRW1CSSNB generates correlated samples. However, SRW1CSSNB generates samples much faster than Wedge. Hence for large graphs with small triangle concentration, we recommend to use the SRW1CSSNB to estimate the triangle counts.
- In Figure 6b, we evaluate our proposed method SRW2CSS and path sampling (which is denoted as 3-path in the figure). For the large graphs Pokec, Flickr, Twitter, and Sinaweibo, our method SRW2CSS performs better than 3-path. This is because SRW2CSS does not need a preprocess step and generates samples much faster than 3-path. For large graphs with more than 100M edges, we recommend to use our method SRW2CSS.

Remarks: The wedge sampling can be adapted to restricted access via Metropolis-Hasting random walks. We present the detailed adapted wedge sampling method and compare it with our framework in the technical report [7]. The simulation results show that our method SRW1CSSNB has much higher accuracy than the adapted wedge sampling method.

6.4 Applications

In this subsection, we apply our framework to analyze the intrinsic properties of the large scale graph Sinaweibo in our datasets. Sinaweibo is the most popular microblog service in China and has reached 222 million monthly active users as of September 2015. It allows users to follow oth-

ers, post and repost messages, comment on others' posts, etc. With the fact that Facebook is an online social network while Twitter is more like a news media [16], we now study whether Sinaweibo acts like a social network or a news media by measuring its similarity to Twitter and Facebook. To measure the similarity, we adopt the definition of graphlet kernel in [32] and restrict the definition to 4-node graphlets. Specifically, for two graphs with 4-node graphlet concentration $\mathbf{c}_1$ and $\mathbf{c}_2$, we define the similarity between them as $\mathbf{c}_1^T \cdot \mathbf{c}_2 / (\|\mathbf{c}_1\| \cdot \|\mathbf{c}_2\|)$. We run the random walk for 20K steps for graphs Facebook, Twitter, and Sinaweibo to estimate the 4-node graphlet concentration and compute the similarity with the estimated value. The results of 100 simulations are reported in Table 7. Our proposed method SRW2CSS gives a more accurate estimate of similarity value compared with PSRW. Besides, we find that Sinaweibo has more similar subgraph building blocks to Twitter, which indicates Sinaweibo also acts like an efficient platform for information diffusion. Note that for these large graphs, it is impractical to crawl the whole datasets for later on analysis. Our framework produces accurate estimates based on a small portion of crawled nodes, which implies our framework is an efficient analysis tool for large graphs with restrict access.

Table 7: Similarity between Sinaweibo and social network Facebook, as well as news media Twitter.

Graph	SRW2CSS	PSRW	Exact
Facebook [1]	**0.5809±0.0501**	0.5856±0.0676	0.5757
Twitter [30]	**0.9988±0.0236**	0.9957±0.0200	0.9999

7. CONCLUSION

In this paper, we propose a novel random walk-based framework which takes one tunable parameter to estimate the graphlet concentration. Our framework is general and can be applied to any k-node graphlets. We derive an analytic bound on the number of random walk steps required for convergence. We also introduce two optimization techniques to further improve the efficiency of our framework. Our experiments with many real-world networks show that the methods with appropriate parameter in our framework produce accurate estimates and outperform the state-of-the-art methods significantly in estimation accuracy.

8. ACKNOWLEDGMENTS

9. REFERENCES

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

[2] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella. Graph sample and hold: A framework for big-graph analytics. In *KDD*, pages 1446–1455, 2014.

[3] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield. Efficient graphlet counting for large networks. In *ICDM*, pages 1–10, 2015.

[4] L. Akoglu, H. Tong, and D. Koutra. Graph-based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery (DAMI)*, 2014.

[5] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *KDD*, pages 16–24, 2008.

[6] M. Bhuiyan, M. Rahman, and M. Al Hasan. GUISE: Uniform sampling of graphlets for large graph analysis. In *ICDM*, pages 91–100, 2012.

[7] X. Chen, Y. Li, P. Wang, and J. Lui. A general framework for estimating graphlet statistics via random walk. *arXiv:1603.07504*, 2016.

[8] K.-M. Chung, H. Lam, Z. Liu, and M. Mitzenmacher. Chernoff-hoeffding bounds for markov chains: Generalized and simplified. *STACS*, pages 124–135, 2012.

[9] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Dimakis. Beyond triangles: A distributed framework for estimating 3-profiles of large graphs. In *KDD*, pages 229–238, 2015.

[10] C. J. Geyer. Markov chain monte carlo lecture notes. *Course notes, Spring Quarter*, 1998.

[11] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *INFOCOM*, pages 1–9, 2010.

[12] S. J. Hardiman and L. Katzir. Estimating clustering coefficients and size of social networks via random walk. In *WWW*, pages 539–550, 2013.

[13] T. Hočevar and J. Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, pages 559–565, 2014.

[14] M. Jha, C. Seshadhri, and A. Pinar. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *WWW*, pages 495–505, 2015.

[15] L. Katzir, E. Liberty, and O. Somekh. Estimating sizes of social networks via biased sampling. In *WWW*, pages 597–606, 2011.

[16] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, pages 591–600, 2010.

[17] C.-H. Lee, X. Xu, and D. Y. Eun. Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. In *SIGMETRICS*, pages 319–330, 2012.

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

[19] R.-H. Li, J. Yu, L. Qin, R. Mao, and T. Jin. On random walk based graph sampling. In *ICDE*, pages 927–938, 2015.

[20] L. Lovász. Random walks on graphs: A survey. In *Combinatorics*, pages 1–46. 1993.

[21] T. Milenkoviæ and N. Pržulj. Uncovering biological network function via graphlet degree signatures. *Cancer Informatics*, 6:257–273, 2008.

[22] T. Milenković, V. Memišević, A. K. Ganesan, and N. Pržulj. Systems-level cancer gene identification from protein interaction network topology applied to melanogenesis-related functional genomics data. *Journal of The Royal Society Interface*, 7:423–437, 2010.

[23] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[24] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[25] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *IMC*, pages 383–389, 2010.

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

[27] A. B. Owen. *Monte Carlo theory, methods and examples*. 2013.

[28] N. Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, pages 853–854, 2010.

[29] M. Rahman, M. Bhuiyan, and M. A. Hasan. Graft: An approximate graphlet counting algorithm for large graph analysis. In *CIKM*, pages 1467–1471, 2012.

[30] R. A. Rossi and N. K. Ahmed. Social network collection - networkrepository. http://networkrepository.com/soc.php, 2013.

[31] C. Seshadhri, A. Pinar, and T. G. Kolda. Triadic measures on graphs: The power of wedge sampling. In *SDM*, pages 10–18, 2013.

[32] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pages 488–495, 2009.

[33] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. In *WWW*, pages 607–614, 2011.

[34] J. Ugander, L. Backstrom, and J. Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *WWW*, pages 1307–1318, 2013.

[35] P. Wang, J. C. S. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan. Efficiently estimating motif statistics of large networks. *TKDE*, pages 8:1–8:27, 2014.

[36] P. Wang, J. C. S. Lui, D. Towsley, and J. Zhao. Minfer: A method of inferring motif statistics from sampled edges. In *ICDE*, pages 1050–1061, 2016.

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

[38] X. Xu, C.-H. Lee, and D. Y. Eun. A General Framework of Hybrid Graph Sampling for Complex Network Analysis. In *INFOCOM*, pages 2795–2803, 2014.

[39] Z. Zhou, N. Zhang, and G. Das. Leveraging history for faster sampling of online social networks. *PVLDB*, pages 1034–1045, 2015.