International Journal on Applications in Information and Communication Engineering
Volume 4: Issue 1: March 2018, pp 1 - 14. www.aetsjournal.com ISSN (Online) : 2394 - 6237

INFERRING HIGHER-ORDER STRUCTURE STATISTICS OF LARGE NETWORKS FROM SAMPLED EDGES

E. BAIRAVI , D. SARIDHA , M. MAKURU

Abstract— Recently exploring locally connected subgraphs (also known as motifs or graphlets) of complex networks attracts a lot of attention. Previous work made the strong assumption that the graph topology of interest is known in advance. In practice, sometimes researchers have to deal with the situation where the graph topology is unknown because it is expensive to collect and store all topological information. Hence, typically what is available to researchers is only a snapshot of the graph, i.e., a subgraph of the graph. Crawling methods such as breadth first sampling can be used to generate the snapshot. However, these methods fail to sample a streaming graph represented as a high speed stream of edges. Therefore, graph mining applications such as network traffic monitoring usually use random edge sampling (i.e., sample each edge with a fixed probability) to collect edges and generate a sampled graph, which we call a "*RESampled graph*". Clearly, a RESampled graph's motif statistics may be quite different from those of the original graph. To resolve this, we propose a framework Minfer, which takes the given RESampled graph and accurately infers the underlying graph's motif statistics. Experiments using large scale datasets show the accuracy and efficiency of our method.

Keywords— graphlet, motif, subgraph sampling, graph mining.

I. INTRODUCTION

Complex networks are widely studied across many fields of science and technology, from physics to biology, and from nature to society. Networks which have similar lower-order topological features such as degree distribution at the level of individual nodes and edges can exhibit significant differences in their local structures. There is a growing interest to explore these local structures (also known as "*motifs*"), which are small connected and induced subgraph (or CIS) patterns that form during the growth of a network. For a set of nodes in the graph G of interest, its induced subgraph is defined as a graph that consists of all of the edges that connect them in G. Motifs have many applications, for example, they are used to characterize communication and evolution patterns in online social net-works (OSNs) [1]–[4], pattern recognition in gene expres-sion profiling [5], protein-protein interaction prediction [6], and coarse-grained topology generation of networks [7]. For instance, 3-node motifs can reveal relationships like "*the friend of my friend is my friend*" and "*the enemy of my enemy is my friend*", which are well known evolution patterns in signed (i.e., friend/foe) social networks. Kunegiset al. [2] considered the significance of motifs in Slashdot Zoo[1] and how they impact the stability of signed networks. Other more complex examples include 4-node motifs such as bi-fans and bi-parallels defined in [8]. Recently, Benson et al. [9] developed a motif-based graph mining framework, and they revealed that a variety of real-world networks exhibit rich higher-order organizational patterns, e.g., they observe that the bi-fan motif acts as the main information propagation unit in neuronal networks.

Although motifs are important in helping researchers understand the underlying network, one major technical hurdle is that it is computationally expensive to enumerate and count all CISes in a large network. Note that there exist a large number of CISes even for a medium size network with less than one million edges. For example, the graphs Slashdot and Epinions, which contain approximately 1.0×10^5 nodes and 1.0×10^6 edges, contain more than 2.0×10^{10} 4-node connected and induced subgraphs [10]. To address this problem, several sampling methods have been proposed to estimate the frequency distribution of motifs [10]–[13]. However, all previous methods focus on designing *computationally efficient* methods to character-ize motifs when the *entire graph* of interest either fits into memory, or an I/O efficient neighbor query API exists to allow one to explore the graph topology, when it is stored on disk. In summary, *these methods assume that the entire graph topology is known*.

In practice the graph of interest may not be known, and the available dataset is just a subgraph sampled from the original graph, because it is expensive to collect and store all topological and meta information. A simple example is given in Fig. 1, where the sampled graph G^* is derived from the dataset representing G. G^* can be generated by crawling methods such as breadth first sampling. However, these methods fail to sample a streaming graph represented as a high speed stream of edges. In this work, we assume the available graph G^* is an RESampled graph that is obtained through random edge sampling, i.e, each edge in G is independently sampled with the same probability $0 \leq p \leq 1$. In practice, this sampling method is popular and easy to implement for

E. Bairavi , Final Year Me-Cse, Meenakshi Ramaswamy Engineering College, Thathanur, Tamilnadu, India
D. Saridha , Final Year Me-Cse, Meenakshi Ramaswamy Engineering College, Thathanur, Tamilnadu, India
M. Makuru , Assistant Professor, Meenakshi Ramaswamy Engineering College, Thathanur, Tamilnadu, India.

streaming graphs. Obtaining a RESampled graph is easy and cheap (the computational and space complexities are both $O(1)$). A RESampled graph can also be used to estimate many other graph statistics such as average node degree, node label distribution, and edge label distribution, which have been studied in pre-vious work [14]. These properties make the random edge sampling technique suitable for the following applications.

- Network Traffic Analysis. Network traffic on network devices such as routers can be represented as a sequence of network packets/flows. Sampling is inevitable for collecting network traffic on backbone routers in order to study the network graph, where a node in the graph represents a host and an edge (u, v) represents a connection from host u to host v, because packets go through routers at too high a rate to gather information from all packets. Therefore, current network devices support simple sampling techniques such as random packets/flows sampling, where random flow sampling can be viewed as random edge sampling over the network graph.
- Network Data Publishing. It is common for service providers to release a small sampled dataset (e.g., a RE-Sampled graph) for a third-party research. Formally, we denote the graph G^* as a RESampled graph of G. One easily observes that a RESampled graph's motif statistics differ from those of the original graph due to uncertainties introduced by sampling. For example, Fig. 2 shows that s^* is a 4-node induced subgraph in the RESampled graph G^*, but we do not know which original induced subgraph s in G it derives from. In fact, s could be any one of the five subgraphs depicted in Fig. 2.

Unlike previous methods [10]–[13], in this paper we assume that it is impossible or computationally expensive to apply graph traversal algorithms over G and we aim to design an *accurate* method to infer the motif statistics of the original graph G from an available RESampled graph G^*. Note that previous methods focus on designing computationally efficient sampling/crawling methods based on sampling *induced* subgraphs in G to avoid the problem shown in Fig. 2. Hence they fail to infer the underlying graph's motif statistics from the given RESampled graph. The gSH method in [15] can be used to estimate the number of connected subgraphs (not necessary induced) from sampled edges. However motif statistics can differ significantly from connected subgraph statistics and the gSH method cannot be easily modified to solve the problem shown in Fig. 2, so the gSH method cannot be applied to characterize motifs.

Contribution: To the best of our knowledge, we are the "first" to study and provide an accurate and efficient solu-tion to estimate motif statistics from a given RESampled graph. Hence, we do away with the previous assumption requiring the entire topology of the graph to be available. We introduce a probabilistic model to study the relation-ship between motifs in the RESampled graph and in the underlying graph. Based on this model, we propose an accurate method, Minfer, to infer the underlying graph's motif statistics from the RESampled graph. We also provide a Fisher information based method to bound the error of our estimates. We further develop new algorithms for exactly counting 3-, 4-, and 5-node motifs that go beyond undirected graphs under both centralized and distributed settings. We conduct experiments on both synthetic and real-world graphs, and the empirical results demonstrate the efficiency and efficacy of our methods.

This paper is organized as follows: Preliminaries and the problem formulation is presented in Section 2. Section 3 presents our method (i.e. Minfer) for inferring subgraph class concentrations of the graph under study from a given RESampled graph. Section 4 presents methods for com-puting the given RESampled graph's motif statistics. The performance evaluation and testing results are presented in Section 5. Section 6 summarizes related work. Concluding remarks then follow.

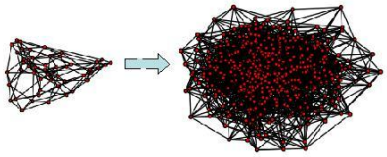


Figure 1. An example of the available G^* and the underlying graph G.

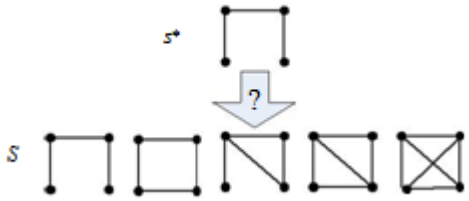


Figure 2. s^* is a 4-node induced subgraph in the RESampled graph G^*, and s is the original induced subgraph of s^* in the original graph G.

II. PRELIMINARIES AND PROBLEM

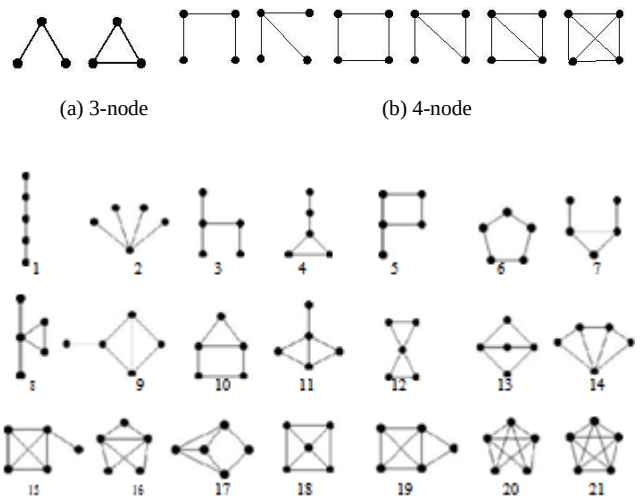
Notation. Denote the underlying graph of interest as a labeled undirected graph $G = (V, E, L)$, where V is a set of nodes, E is a set of *undirected* edges, $E \in V \times V$, and L is a set of labels $l_{u;v}$ associated with edges $(u, v) \in E$. For example, we attach a label $l_{u;v} \in \{\rightarrow, \leftarrow, \leftrightarrow\}$ to indicate the direction of the edge $(u, v) \in E$ for a directed network. Edges may have other labels too, for instance, in a signed network, edges have positive or negative labels to represent *friend* or *foe* relationship. If L is empty, then G is an unlabeled undirected graph, which is equivalent to an undirected graph.

Motif concentrations. Motif concentration is a metric that represents the distribution of various subgraph patterns that appear in G. To illustrate, we show the 3-, 4- and 5-node motifs in Figs. 3 and 4. In Fig. 4, we only show 3-node directed/signed motifs due to the large number of directed/signed motifs with 4 and 5 nodes (there exist 199 and 9,364 4- and 5-node directed motifs respectively). To define

motif concentration formally, we first introduce some notation. An induced subgraph of G, $G' = (V', E', L')$, $V' \subset V$, $E' \subset E$ and $L' \subset L$, is a subgraph whose edges and associated labels are all in G, i.e. $E' = \{(u, v) : u, v \in V', (u, v) \in E\}$, $L' = \{l_{u;v} : u, v \in V', (u, v) \in E\}$. We define $C^{(k)}$ as the set of all CISes with k nodes in G, and denote $n^{(k)} = |C^{(k)}|$. Let T_k denote the number of k-node motifs and $M_i^{(k)}$ denote the i^{th} k-node motif. For example, $T_4 = 6$ and $M_1^{(4)}, \ldots, M_6^{(4)}$ are the six 4-node undirected motifs depicted in Fig. 3(b). Then we partition $C^{(k)}$ into Tk equivalence classes, or $C_1^{(k)}, \ldots, C_{T_k}^{(k)}$, where 1 T_k CISes within $C_i^{(k)}$ are isomorphic to $M_i^{(k)}$. Note that in this paper node labels are not taken into account when checking the isomorphism. When there exist multiple isomorphisms between a CIS and a motif, we only count one of them. Let $n_i^{(k)}$ denote the frequency of motif $M_i^{(k)}$, i.e., the number of CISes in G isomorphic to $M_i^{(k)}$. Formally, we have $n_i^{(k)} = |C_i^{(k)}|$, which is the number of CISes in $C_i^{(k)}$.

(k) Then the concentration of $M_i^{(k)}$ is defined as $\omega_i^{(k)} = \frac{n_i}{n(k)}$, $1 \le i \le T_k$. Thus, $\omega_i^{(k)}$ is the *fraction* of k-node CISes isomorphic to motif $M_i^{(k)}$ among all k-node CISes.

1



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

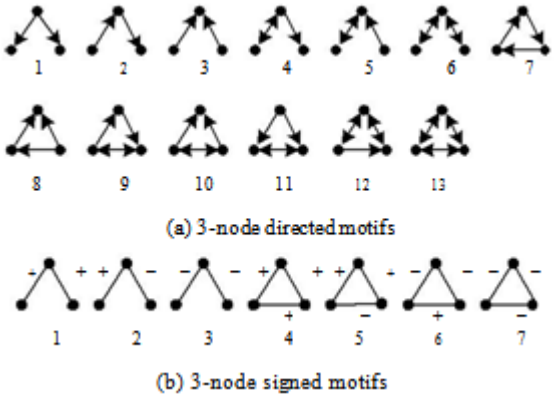
(c) 5-node

Figure 3. All classes of 3-, 4-, and 5-node undirected and connected motifs (The numbers are the motif IDs).

Assumptions. We make the follow assumptions: assumption 1: The complete G is not available to us, but a RESampled graph $G^* = (V^*, E^*, L^*)$ of G is given, where $V^* \in V$, $E^* \in E$, and L^* are node, edge, and edge label sets of G^* respectively. G^* is obtained by random edge sampling, i.e., each edge in E is independently sampled with the same probability $0 < p < 1$, where p is known in advance; assumption 2: The label of a sampled edge $(u, v) \in G^*$ is the same as that of (u, v) in G, i.e., $l_{u;v^*} = l_{u;v}$. These two assumptions are satisfied by many applications' data collection procedures. For instance, the data generated by an application such as network traffic monitoring is given as a stream of directed edges. The following simple method is computationally and memory efficient for collecting edges and generating a small RE-Sampled graph that can be sent to remote network traffic analysis center: Each incoming directed edge $u \rightarrow v$ is sampled when $\tau_{u;v} \le \rho p$, where ρ is an integer (e.g., 10,000) and τ is a hash function satisfying $\tau_{u;v} = \tau_{v;u}$ and mapping edges into integers $0, 1, \ldots, \rho - 1$ uniformly. The property $\tau_{u;v} = \tau_{v;u}$ guarantees that edges $u \rightarrow v$ and $u \leftarrow v$ are either both sampled or discarded. Hence the label of a sampled edge $(u, v) \in E^*$ is the same as that of (u, v) in G. Using universal hashing [16], a simple instance of $\tau_{u;v}$ is given as the following function when each $v \in V$ is an integer smaller than Δ $\tau_{u;v} = a(\min\{u, v\}\Delta + \max\{u, v\}) + b \mod \gamma \mod \rho$,

where γ is a prime larger than Δ^2, a and b are any integers with $a \in \{1, \ldots, \rho - 1\}$ and $b \in \{0, \ldots, \rho - 1\}$. We can easily find that $\tau_{u;v} = \tau_{v;u}$ and τ maps edges into integers $0, \ldots, \rho - 1$ uniformly. The computational and space complexities of the above sampling method are both $O(1)$, which make it practical for data collection.

Problem. We aim to accurately infer the motif concentrations of G based on the given RESampled graph G^*.



(a) 3-node directed motifs

(b) 3-node signed motifs

Figure 4. All classes of three-node directed and signed motifs (The numbers are the motif IDs).

III. MOTIF STATISTICAL INFERENCE

The motif statistics of RESampled graph G^* and original graph G can be quite different. In this section, we introduce a probabilistic model to bridge the gap between the motif statistics of G^* and G. Using this model, we will establish a simple and concise relationship between the motif statistics of G and G^*. We then propose an efficient method to infer the motif concentration of G from G^*. Finally, we also give a method to construct confidence intervals of our estimates of motif concentrations.

1) Probabilistic Model of Motifs in G^* and G

To estimate the motif statistics of G based on G^*, we

develop a probabilistic method to model the relationship between the motifs in G^* and G. Define $P = [P_{i;j}]$ where $P_{i;j}$ is the probability that a CIS s^* in G^* is isomorphic to motif $M_i^{(k)}$ given that its original CIS s in G is isomorphic to motif $M_j^{(k)}$, i.e., $P_{i;j} = P\,(M(s^*) = M_i^{(k)} | M(s) = M_j^{(k)})$.

To obtain $P_{i;j}$, we first compute $\phi_{i;j}$, which is the number of subgraphs of $M_j^{(k)}$ isomorphic to $M_i^{(k)}$. For example, $M_2^{(3)}$ (i.e., the triangle) includes three subgraphs isomorphic to $M_1^{(3)}$ (i.e., the wedge) for the undirected graph shown in Fig. 3(a). Thus, we have $\phi_{1;2} = 3$ for 3-node undirected motifs. When $i = j$, $\phi_{i;j} = 1$. Note that it is not easy to compute $\phi_{i;j}$ manually for 4- and 5-node motifs. Hence we provide a simple method to compute $\phi_{i;j}$ in Algorithm 1, where we enumerate each bijection that maps nodes in $M_i^{(k)}$ to nodes in $M_j^{(k)}$ to check whether the bijection also maps edges of $M_i^{(k)}$ to edges in $M_j^{(k)}$ with the same labels. In Step 2 of Algorithm 1, we use a counter $y_{i;j}$ to record the number of bijections that meet this condition. There may exist multiple isomorphisms between a subgraph and a motif, but $\phi_{i;j}$ only counts one of them. Therefore, in Step 3 of Algorithm 1, we use a counter z_i to record the number of automorphisms of $M_i^{(k)}$. At last, we have $\phi_{i;j} = y_{i;j} / z_i$. The computational complexity is $O(k^2 k!)$. We want to emphasize that the cost of computing $\phi_{i;j}$ is not a big concern, because these values are static and independent of the input graph, and they can be computed once and for all and stored in a static table. Denote by $V(s)$ and $E(s)$ the sets of nodes and edges in subgraph s respectively. We have the following equation

$$P_{i;j} = \phi_{i;j}\, p^{|E(M_i^{(k)})|}\, q^{(|E(M_j^{(k)})| - |E(M_i^{(k)})|)},$$

where $q = 1 - p$. The above model implies that in expectation, the fraction of these CISes that appear as CISes isomorphic to $M_i^{(k)}$ in G^* is $P_{i;j}$.

2) Motif Concentration Estimation

Using the above probabilistic model, we propose a method Minfer to estimate motif statistics of G from G^*. Denote by $m_i^{(k)}$, $1 \le i \le T_k$, $k = 3, 4, \ldots$, the number of CISes in G^* isomorphic to motif $M_i^{(k)}$. The method to compute $m_i^{(k)}$ is presented in the next section. Then, the expectation of $m_i^{(k)}$ is computed as

$$\mathrm{E}[m_i^{(k)}] = \sum_{1 \le j \le T_k} n_j^{(k)} P_{i;j}. \qquad (1)$$

In matrix notation, Equation (1) can be expressed as $\mathrm{E}[\mathbf{m}^{(k)}] = P\,\mathbf{n}^{(k)}$, where $P = [P_{ij}]_{1 \le i;j \le T_k}$, $\mathbf{n}^{(k)} = (n_1^{(k)}, \ldots, n_{T_k}^{(k)})'$, and $\mathbf{m}^{(k)} = (m_1^{(k)}, \ldots, m_{T_k}^{(k)})'$. Then, we have $\mathbf{n}^{(k)} = P^{-1}\mathrm{E}[\mathbf{m}^{(k)}]$. Thus, we estimate $\mathbf{n}^{(k)}$ as

$$\hat{\mathbf{n}}^{(k)} = P^{-1}\mathbf{m}^{(k)},$$

where $\hat{\mathbf{n}}^{(k)} = (\hat{n}_1^{(k)}, \ldots, \hat{n}_{T_k}^{(k)})'$. We easily have $\mathrm{E}[\hat{\mathbf{n}}^{(k)}] = \mathrm{E}[P^{-1}\mathbf{m}^{(k)}] = P^{-1}\mathrm{E}[\mathbf{m}^{(k)}] = \mathbf{n}^{(k)}$, therefore $\hat{\mathbf{n}}^{(k)}$ is an unbiased estimator of $\mathbf{n}^{(k)}$. Finally, we estimate $\omega_i^{(k)}$ as

$$\hat{\omega}_i^{(k)} = \frac{\hat{n}_i^{(k)}}{\sum_{j=1}^{T_k} \hat{n}_j^{(k)}}, \qquad 1 \le i \le T_k. \qquad (2)$$

Denote by $\rho_i^{(k)}$ the concentration of motif $M_i^{(k)}$ in G^*, i.e., $\rho_i^{(k)} = \frac{m_i^{(k)}}{m^{(k)}}$. We observe that Equation (2) is equivalent to the following equation, which directly describes the relationship between motif concentrations of G and G^*. Let $\hat{\boldsymbol{\omega}} = [\hat{\omega}_1^{(k)}, \ldots, \hat{\omega}_{T_k}^{(k)}]'$ and $\boldsymbol{\rho} = [\rho_1^{(k)}, \ldots, \rho_{T_k}^{(k)}]'$, then we have

$$\hat{\boldsymbol{\omega}} = \frac{P^{-1}}{W}\boldsymbol{\rho}, \qquad (3)$$

where $W = [1, \ldots, 1]P^{-1}$ is a normalizer. For 3-node undirected motifs, $P = \begin{pmatrix} p^2 & 3qp^2 \\ 0 & p^3 \end{pmatrix}$, and the inverse of P is $P^{-1} = \begin{pmatrix} p^{-2} & -3qp^{-3} \\ 0 & p^{-3} \end{pmatrix}$. Due to limited space, we omit the expressions for P and P^{-1} for other motifs such as 3-node signed/directed motifs, 4-, and 5-node undirected/directed/signed motifs.

Algorithm 1: Computing $\phi_{i;j}$, i.e., the number of subgraphs of $M_j^{(k)}$ that are isomorphic to $M_i^{(k)}$.

Step 1: Generate two graphs $G = (\{v_1, \ldots, v_k\}, E, L)$ and $G = (\{u_1, \ldots, u_k\}, E, L)$, isomorphic to motifs $M_i^{(k)}$ and $M_j^{(k)}$ respectively, where E and L are the edges and edge labels of G with nodes $v_1, \ldots, v_k$, and E and L are the edges and edge labels of G with nodes $u_1, \ldots, u_k$.

Step 2: Initialize a counter $y_{i;j} = 0$. For each permutation $(x_1, \ldots, x_k)$ of integers $1, \ldots, k$, $y_{i;j}$ stays unchanged when there exists an edge $(v_a, v_b) \in E$ satisfying $(u_{x_a}, u_{x_b}) \notin E$ or $l_{v_a;v_b} \neq l_{u_{x_a};u_{x_b}}$, and $y_{i;j} = y_{i;j} + 1$ otherwise.

Step 3: Initialize a counter $z_i = 0$. For each permutation $(x_1, \ldots, x_k)$ of integers $1, \ldots, k$, z_i stays unchanged when there exists an edge $(v_a, v_b) \in E$ satisfying $(v_{x_a}, v_{x_b}) \notin E$ or $l_{v_a;v_b} \neq l_{v_{x_a};v_{x_b}}$, $z_i = z_i + 1$ otherwise.

Step 4: Finally, $\phi_{i;j} = y_{i;j} / z_i$.

3) Estimated Lower Bound on Estimation Errors

It is difficult to directly analyze the errors of our estimate $\hat{\boldsymbol{\omega}}$, because it is complex to model the dependence of sampled CISes due to their shared edges and nodes. Instead, we derive a lower bound on the mean squared error (MSE) of $\hat{\boldsymbol{\omega}}$ using the Cramér-Rao lower bound (CRLB) of $\hat{\boldsymbol{\omega}}$, which gives the smallest MSE that any unbiased estimator of $\boldsymbol{\omega}$ can achieve. For a k-node CIS s selected from k-node CISes of G at random, the probability that s is isomorphic to the j^{th} k-node motif is $P\,(M(s) = M_j^{(k)}) = \omega_j^{(k)}$.

Let s^* be the induced subgraph of the node set $V(s)$ in

RESampled graph G^*. Clearly, s^* may not be connected.

Furthermore, there may exist nodes in $V(s)$ that are not present in G^*. We say s^* is *evaporated* in G^* for these two scenarios. Let $P_{0;j}$ denote the probability that s^* is evaporated given that its original CIS s is isomorphic to the j^{th} k-node motif. Then, we have $P_{0;j} = 1 - \sum_{l=1}^{T_k} P_{l;j}$. For a random k-node CIS s of G, the probability that its original k-node CIS s^* in G^* is isomorphic to the i^{th} k-node motif is

$$\xi_i = P\,(M(s^*) = M_i^{(k)}) = \sum_{j=1}^{T_k} P_{i;j}\,\omega_j^{(k)}, \quad 1 \le i \le T_k,$$

and the probability that s^* is evaporated is $\xi_0 = \sum_{j=1}^{T_k} P_{0;j}\,\omega_j^{(k)}$. When s^* is evaporated, we denote $M(s^*) = 0$. Then, the likelihood function of $M(s^*)$ with respect to $!^{(k)}$ is

$$f(i|!^{(k)}) = \xi_i, \qquad 0 \le i \le T_k.$$

The Fisher information of $M(s^*)$ with respect to $!^{(k)}$ is defined as a matrix $J = [J_{i;j}]_{1\le i,j\le T_k}$, where

$$J_{i;j} = \mathrm{E}\left[\frac{\partial \ln f(l\,!^{(k)}}{\partial \omega_i^{(k)}}\,\frac{\partial \ln f(l\,!^{(k)}}{\partial \omega_j^{(k)}}\right] = \sum_{l=0}^{T_k} \frac{P_{l;i}\,P_{l;j}}{\xi_l}.$$

J is an effective metric to measure the amount of information that a random k-node CIS s^* in G^* carries about an unknown parameter $!^{(k)}$. For simplicity, we assume that the CISes of G^* are independent (i.e., no overlapping edges). Then the Fisher information matrix of all k-node CISes is $n^{(k)}J$. The Cramer´-Rao Theorem states that the MSE of any unbiased estimator is lower bounded by the inverse of the Fisher information matrix, i.e.,

$$\mathrm{MSE}(\hat{\omega}_i^{(k)}) = \mathrm{E}\left[(\hat{\omega}_i^{(k)} - \omega_i^{(k)})^2\right] \ge \frac{(J^{-1})_{i;i} - !^{(k)}(!^{(k)})^{\mathrm{T}}}{n^{(k)}}$$

provided some weak regularity conditions hold [17]. Here the term $!^{(k)}(!^{(k)})^{\mathrm{T}}$ corresponds to the accuracy gain obtained by accounting for the constraint $\sum_{i=1}^{T_k} \omega_i^{(k)} = 1$.

The CRLB method provides us a way to set p properly, i.e., we can perform a pilot study to estimate/guess the original graph's statistics, and then use the CRLB method to evaluate the estimation errors for different p.

IV. ENUMERATE AND CLASSIFY CISES

Function $M(s)$ used in previous sections, which determines the motif ID of a k-node CIS s, can be achieved by graph isomorphism testing tools such as NAUTY [18]. In this section, we introduce a simple yet quick method for small values of k = 3, 4, 5, which only requires only one hash table lookup for classifying CISes with up to 5 nodes. Similarly, existing generalized graph enumeration methods such as [12] can be used for enumerating all k-node CISes in RESampled graph G^*, while it is not easy to apply and is (computationally and memory) inefficient for small values of k = 3, 4, 5. In this section, we also present a method (an extension of the NodeIterator++ method in [19]) to enumerate and count 3-node CISes in G^* and new two methods to enumerate and count 4 and 5-node CISes in G^* respectively. In what follows, we denote $N^*(u)$ as the neighbors of u in G^*. Note that in this section G^* is the default graph when we define a function. For example, the CIS with nodes u, v, and w refers to the CIS with nodes u, v, and w in G^*.

1) Classify 3-, 4-, and 5-node CISes

Let d be the number of different edge labels for the graph of interest. For example, edges in directed graphs have d = 3 labels →, ←, and ↔, and edges in signed graphs have d = 2 labels + and −. We order the edge labels. For the j^{th} edge label, we define its label string as a bit string consisting of d bits, where the j^{th} bit is 1 and the other bits are 0. For example, we let "100", "010", and "001" be label string for directed edge labels →, ←, and ↔ respectively.

For a k-node CIS s, we quickly compute a bit string, and then retrieve its motif ID $M(s)$ by looking up the string from a hash table H_T. More specifically,

Step 1: compute a string matrix A where $A_{i;j}$ records the edge label string of edge $(V_i(s), V_j(s)) \in E(s)$, where $V_i(s)$ and $V_j(s)$ is the i^{th} and j^{th} nodes in s. $A_{i;j}$ is a bit string consisting of d zeros when $(V_i(s), V_j(s)) \notin E(s)$. Step 2: concatenate bit strings of all elements in A, that is,

$$str = A_{1;1}||\ldots||A_{1;k}||A_{2;1}\ldots||A_{2;k}||A_{k;1}\ldots||A_{k;k}.$$

Step 3: get $M(s)$ by looking up key str in hash table H_T. Now, we discuss how to generate H_T in advance. For each motif $M_j^{(k)}$ with k nodes, we use the above step 1 and step 2 to compute a str for each permutation of nodes in $M_j^{(k)}$, and then update hash table H_T (str) = j. There exist $k!$ different permutations for k nodes. Thus, the computational and memory complexities of generating H_T are $O(k!T_k)$ and $O(k!dT_k)$ respectively. The compu-tational complexity of classifying a single k-node CIS is $O(\log(k!T_k))$.

2) Enumerate 3-node CISes

Similar to the NodeIterator++ method in [19], we "pivot" (the associated operation is discussed later) each node $u \in V^*$ to enumerate CISes including u. For any two neighbors v and w of u, we can easily find that the induced graph s with nodes u, v and w is a 3-node CIS. Thus, we enumerate each pair of two nodes in $N^*(u)$, and update the 3-node CIS consisting of u and the pair of nodes. We call this process "pivoting" u for 3-node CISes. Its pseudo-code is shown in Algorithm 2. Clearly, a 3-node CIS s is counted three times when the associated undirected graph of s by discarding edge labels is isomorphic to a triangle, once by pivoting each node u, v, and w. Let $\succ$ be

an arbitrary total order on all of the nodes, which can be easily defined and obtained, e.g. from array position or pointer addresses. To ensure each CIS is enumerated once and only once, we let one and only one node in a CIS be the *leader* of the CIS, which is "responsible" for making sure the CIS gets counted. When we "pivot" u and enumerate a CIS s, s is counted if u is the leader of s. Otherwise, s is discarded and not counted. We use the same method in [19], [20], i.e., let the node with lowest order in a CIS whose associated undirected graph is isomorphic to a triangle be the leader. For the other classes of CISes, their associated undirected graphs are isomorphic to an unclosed wedge, i.e., the 1st motif in Fig. 3(a). For each of these CISes, we let the node in the center of its associated undirected graph (e.g., the node with degree 2 in the unclosed wedge) be the leader.

Algorithm 2: Pivot3CIS(G^*, u). Function IND(Γ) returns the CIS with the node set Γ, and $M(s)$ returns the motif class ID of s.

```
input: G* = (V*, E*, L*) and u ∈ V*
output: m(3)_u = (m(3)_{u;1}, . . . , m(3)_{u;T3})^T
for v ∈ N*(u) do
    for w ∈ N*(u) and w ≻ v do
        s ← IND({u, v, w});
        if (w, v) ∈ E* and u ≻ v then
            continue();
        end
        i ← M(s);
        m(3)_{u;i} ← m(3)_{u;i} + 1;
    end
end
```

3) Enumerate 4-node CISes

To enumerate 4-node CISes, we "pivot" each node u as follows: For each pair of u's neighbors v and w where $w \succ v$, we compute the neighborhood of u, v, and w defined as $\Gamma = N^*(u) \cup N^*(v) \cup N^*(w) - \{u, v, w\}$. For any node $x \in \Gamma$, we observe that the induced graph s consisting of nodes u, v, w, and x is a 4-node CIS. Thus, we enumerate each node x in Γ, and update the 4-node CIS consisting of u, v, w, and x. We repeat this process until all pairs of u's neighbors v and w are enumerated and processed. Algorithm 3 shows the pseudo-code of "pivoting" u for 4-node CISes. Similar to 3-node CISes, some 4-node CISes may be enumerated and counted more than once when we "pivot" each node u as above. To solve this problem, we propose the following methods for making sure each 4-node CIS s is enumerated and gets counted once and only once: When $(u, x) \in E^*$ and $w \succ x$, we discard x. Otherwise, denote by $s\hat{}$ the associated undirected graph of s by discarding edge labels. When $s\hat{}$ includes one and only one node u having at least 2 neighbors in $V(\hat{}s)$, we let u be the *leader* of s. When $s\hat{}$ includes more than one node having at least 2 neighbors in $V(\hat{}s)$, we let the node with lowest order among the nodes having at least 2 neighbors in $V(\hat{}s)$ be the leader of s. For example, the nodes 4, 6, and 3 are the leaders of the 1st, 2nd, and 3rd subgraphs in Fig. 5.

4) Enumerate 5-node CISes

Algorithm 4 shows the pseudo-code of "pivoting" u for 5-node CISes. For a 5-node CIS s, we classify it into two types according to its associated undirected graph $s\hat{}$: type-1 5-node CIS s: $s\hat{}$ includes at least one node with more than two neighbors in $V(\hat{}s)$; type-2 5-node CIS s: All nodes in $s\hat{}$ have no more than two neighbors in $V(\hat{}s)$, i.e., $s\hat{}$ is isomorphic to a 5-node line or a circle, i.e., the 1st or 6th motifs in Fig. 3(c). We propose two different methods to enumerate these two types of 5-node CISes respectively.

To enumerate Type-1 5-node CISes, we "pivot" each node u as follows: When u has at least three neighbors, we enumerate each combination of three nodes v, w, $x \in N^*(u)$ where $x \succ w \succ v$, and then compute the neighbor-hood of u, v, w, and x, defined as $\Gamma \leftarrow N^*(u) \cup N^*(v) \cup N^*(w) \cup N^*(x) - \{u, v, w, x\}$. For any node $y \in \Gamma$, we observe that the induced graph s consisting of nodes u, v, w, x, and y is a 5-node CIS. Thus, we enumerate each node y in Γ, and update the 5-node CIS consisting of u, v, w,

Algorithm 3: Pivot4CIS(G^*, u). Functions IND(Γ) and $M(s)$ are the same as those in Algorithm 2, UND(s) returns the associated undirected graph of s by discarding edge labels, minN(Λ) returns the node with the lowest order in $V(\hat{}s)$, FN($\hat{}s$, t) returns the set of nodes in $V(\hat{}s)$ having at least t neighbors in $V(\hat{}s)$, and $N^*(S)$ returns a set consisting of the neighbors of nodes in set S, i.e., $N^*(S) = \cup_{v \in S} N(v) - S$.

```
input: G* = (V*, E*, L*) and u ∈ V*
output: m(4)_u = (m(4)_{u;1}, . . . , m(4)_{u;T4})^T
for v ∈ N*(u) do
    for w ∈ N*(u) and w ≻ v do
        Γ = N*({u, v, w});
        for x ∈ Γ do
            if (u, x) ∈ E* and w ≻ x then
                continue();
            end
            s ← IND({u, v, w, x});
            s^ ← UND(s), Λ ← FN(^s, 2);
            if |Λ| ≥ 2 then
                if u ≻ minN(Λ) then
                    continue();
                end
            end
            i ← M(s), m(4)_{u;i} ← m(4)_{u;i} + 1;
        end
    end
end
```

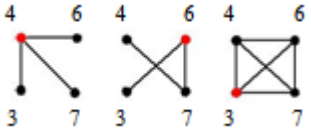


Figure 5. Examples of the leaders of 4-node CISs. Graphs shown are CISes' associated undirected graphs, and the number near to a node represents the node order. Red nodes are the leaders.

Algorithm 4: Pivot5CIS(G^*, u). For a node set S, we define function $N_+(S) = N(S) \cup S$, and other functions are the same as those in Algorithms 2 and 3.

```
input: G* = (V*, E*, L*) and u ∈ V*
output: m(5)_u = (m(5)_u;1, ..., m(5)_u;T5)^T
for v ∈ N(u) do
    for w ∈ N(u) and w ≻ v do for x ∈ N(u)
        and x ≻ w do
            Γ ← N(u, v, w, x);
            for y ∈ Γ do
                if (v, u) ∈ E* and x ≻ y then
                    continue();
                end
                s ← IND(u, v, w, x, y);
                ŝ ← UND(s), Λ ← FN(ŝ, 3);
                if |Λ| ≥ 2 and u ≻ minN(Λ) then
                    continue();
                end
                i ← M(s), m(5)_u;i ← m(5)_u;i + 1;
            end
        end
        if ∈ then
            Γ_w ← N(v) − N+(u, w);
            for x ∈ Γ_w do
                Γ_w ← N(w) − N+(u, v);
                for y ∈ Γ_w do
                    if (x, y) ∈ E* and
                    u ≻ minN(u, v, w, x, y) then
                        continue();
                    end
                    s ← IND(u, v, w, x, y),
                    i ← M(s), m(5)_u;i ← m(5)_u;i + 1;
                end
            end
        end
    end
end
```

x, and y. We repeat this process until all combinations of three nodes v, w, $x \in N(u)$ are enumerated and processed. Similar to 4-node CISes, we propose the following method to make sure each 5-node s is enumerated and gets counted once and only once: When $(y, u) \in E^*$ and $y \succ x$, we discard y. Otherwise, let $\hat{s}$ be the associated undirected graph of s, and we then pick the node with lowest order among the nodes having more than two neighbors in $V(\hat{s})$ be the leader. The 3rd and 4th subgraphs in Fig. 6 are two corresponding examples.

To enumerate Type-2 5-node CISes, we "pivot" each node u as follows: When u has at least two neighbors, we first enumerate each pair of u's neighbors v and w where $(v, w) \notin E^*$. Then, we compute Γ_v defined as the set of v's neighbors not including u and w and not connected to u and w, that is, $\Gamma_v \leftarrow N(v) - \{u, w\} - N(u) - N(w)$.

Similarly, we compute Γ_w defined as the set of w's neigh-bors not including u and v and not connected to u and v, i.e., $\Gamma_w \leftarrow N(w) - \{u, v\} - N(u) - N(v)$. Clearly, $\Gamma_v \cap \Gamma_w = \emptyset$. For any $x \in \Gamma_v$ and $y \in \Gamma_w$, we observe that the induced graph s consisting of nodes u, v, w, x, and y is a Type-2 5-node CIS. Thus, we enumerate each pair $(x, y) \in \Gamma_v \times \Gamma_w$, and update the 5-node CIS consisting of u, v, w, x, and y. We repeat this process until all pairs of u's neighbors v and w are enumerated and processed. To make sure each CIS s is enumerated and gets counted once and only once, we let the node with lowest order be the leader when the associated undirected graph $\hat{s}$ of s isomorphic to a 5-node circle. When $\hat{s}$ is isomorphic to a 5-node line, we let the node in the center of the line be the leader. The 1st and 2nd subgraphs in Fig. 6 are two examples.

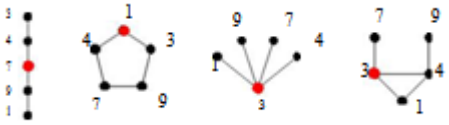


Figure 6. Examples of the leaders of 5-node CISs. Graphs shown are CISes' associated undirected graphs, and the number near to a node represents the node order. Red nodes are the leaders.

5) Distributed Implementation on GraphLab

Algorithms in Sections 4.2, 4.3, and 4.4 can be easily parallelized on a single machine. In this section, we further implement these algorithms for enumerating 3-, 4- and 5-node CISes on a popular distributed graph computing platform GraphLab [21]. To the best of our knowledge, this is the first to count 3-, 4- and 5-node motifs that go beyond undirected graphs under a distributed setting. We adopt the Gather-Apply-Scatter (GAS) framework of GraphLab [21]. GAS follows the vertex-centric programming paradigm, which stores data acted upon on every node and edge, and is executed in parallel on each node. More specifically, GAS mainly consists of three phases: Gather, Apply and Scatter. In the "Gather" phase, each node $u \in V^*$ collects information (e.g., neighbor list $N(u)$ and labels of edges between u and nodes in $N(u)$) about its adjacent nodes and edges. In the Apply phase, each node takes computation from the information returned by the "Gather" phase, and updates its own data. In the "Scatter" phase, each node updates data on its adjacent edges.

The key idea behind our distributed algorithms is to *"gather"* the minimal but necessary information for piv-oting each node $u \in V$, which is discussed in previous Sections 4.2, 4.3, and 4.4, and then pivot u by *"apply-ing"* functions Pivot3CIS(G^*, u), Pivot4CIS(G^*, u), and Pivot5CIS(G^*, u) in Algorithms 2, 3, and 4. We start by introducing the whole

framework for our distributed system. Let k-hop neighbors of $u \in V^*$ be the nodes that can reach u with no more than k steps on the undirected graph of G^*. For example, $N^*(u)$ is the set of 1-hop neighbors of u. Let $G^*_{u;k}$ denote the k-hop ego-network of u, which is defined as the inducted subgraph consisting of nodes u and its k-hop neighbors. We can easily find that $G^*_{u;2}$ is necessary and enough for pivoting 3- and 4-node CISes including u, and the $G^*_{u;3}$ is required for pivoting 5-node CISes including u. By iteratively collecting information from neighbors, each node's 2- and 3-hop ego-networks can be efficiently obtained by *"gathering"* twice and three times respectively. Our algorithms are shown in Algorithms 5 and 6. Note that we do not change data (i.e., edge labels in this work) stored on edges, so the "Scatter" stage is not required in our algorithms.

Algorithm 5: Counting 3-/4-node motifs on Graph Lab.

Input : $G^* = (V^*, E^*, L^*)$.
Output: 3-/4-node motif frequency $m^{(3)}/m^{(4)}$.
Gather : Gather adjacent nodes and edges of each node $u \in V^*$.
Apply : Compute 1-hop ego-network $G^*_{u:1}$ and store it on u.
Gather : Gather data stored on the neighbors of each node $u \in V^*$.
Apply : Compute 2-hop ego-network $G^*_{u;2}$, and then call function $m_u^{(3)} = \text{Pivot3CIS}(G^*_{u;2}, u)$, or $m_u^{(4)} = \text{Pivot4CIS}(G^*_{u;2}, u)$.
Reduce: Compute $m^{(3)} = \sum_{u \in V} m_u^{(3)}$ or $m^{(4)} = \sum_{u \in V} m_u^{(4)}$.

Algorithm 6: Counting 5-node motifs on GraphLab

Input : $G^* = (V^*, E^*, L^*)$.
Output: 5-node motif frequency $m^{(5)}$.
Gather : Gather adjacent nodes and edges of each node $u \in V^*$.
Apply : Compute 1-hop ego-network $G^*_{u:1}$ and store it on u.
Gather : Gather data stored on the neighbors of each node $u \in V^*$.
Apply : Compute 2-hop ego-network $G^*_{u;2}$, and then call function $m_u^{(5)} = \text{Pivot5CIS}(G^*_{u;3}, u)$.
Reduce: Compute $m^{(5)} = \sum_{u \in V} m_u^{(5)}$.

V. EVALUATION

1) Datasets

We evaluate the performance of our methods on publicly available datasets taken from the Stanford Network Analysis Platform (SNAP) (www.snap.stanford.edu), which are summarized in Table 1. We start by evaluating the performance of our methods in characterizing 3- and 4-node CISes over million-edge graphs: Flickr, Pokec, LiveJournal, YouTube, Web Google, and Wiki talk, contrasting our re-sults with the ground truth computed through an exhaustive method. It is computationally intensive to calculate the ground-truth of 5-node CIS classes in large graphs. For example, we easily observe that a node with degree $d > 4$ is included in at least $\frac{1}{24} d(d-1)(d-2)(d-3)$ 5-node CISes, therefore it requires more than $O(10^{19})$ operations to enumerate the 5-node CISes within the Wiki-talk graph, which contains one node with 100,029 neighbors. To solve this problem, the experiments for 5-node CISes are per-formed on two relatively small graphs com-DBLP, and com-Amazon, where computing the ground-truth is feasi-ble. We also evaluate the performance of our methods for characterizing signed CIS classes in graphs sign-Epinions, sign-Slashdot08, and sign-Slashdot09.

Table 1 - Graph datasets used in our experiments.

Graph	nodes	edges	max-degree
Flickr [22]	1,715,255	15,555,041	27,236
Pokec [23]	1,632,803	22,301,964	14,854
LiveJournal [22]	5,189,809	48,688,097	15,017
YouTube [22]	1,138,499	2,990,443	28,754
Wiki-Talk [24]	2,394,385	4,659,565	100,029
Web-Google [25]	875,713	4,322,051	6,332
sign-Epinions [26]	119,130	704,267	3,558
sign-Slashdot08 [26]	77,350	416,695	2,537
sign-Slashdot09 [26]	82,144	504,230	2,552
com-DBLP [27]	317,080	1,049,866	343
com-Amazon [27]	334,863	925,872	549
p2p-Gnutella08 [28]	6,301	20,777	97
ca-GrQc [29]	5,241	14,484	81
ca-CondMat [29]	23,133	93,439	279
ca-HepTh [29]	9,875	25,937	65

2) Error Metric

In our experiments, we focus on the normalized root mean square error (NRMSE) to measure the relative error of the estimator $\hat{\omega}_i$ of the subgraph class concentration ω_i, $i = 1, 2, \ldots$. NRMSE($\hat{\omega}_i$) is defined as: NRMSE($\hat{\omega}_i$) = $\frac{\text{MSE}(\hat{\omega}_i)}{\omega_i}$, $i = 1, 2, \ldots$, where MSE($\hat{\omega}_i$) is defined as the mean square error (MSE) of an estimate $\hat{\omega}_i$ with respect to its true value $\omega_i > 0$, that is MSE($\hat{\omega}_i$) = $E[(\hat{\omega}_i - \omega_i)^2]$ = Var($\hat{\omega}_i$) + $(E[\hat{\omega}_i] - \omega_i)^2$. We note that MSE($\hat{\omega}_i$) decomposes into a sum of the variance and bias of the estimator $\hat{\omega}_i$. Both quantities are important and need to be as small as possible to achieve good estimation performance. When $\hat{\omega}_i$ is an unbiased

estimator of ω_i, then we have MSE($\hat{\omega}_i$) = Var($\hat{\omega}_i$) and thus NRMSE($\hat{\omega}_i$) is equivalent to the normalized standard error of $\hat{\omega}$, i.e.,

$$\sqrt{}^{i}\text{NRMSE}(\hat{\omega}_i) = \text{Var}(\hat{\omega}_i)/\omega_i.$$ Note that our metric uses the relative error. Thus, when ω_i is small, we consider values as large as NRMSE($\hat{\omega}_i$) = 1 to be acceptable. In all our experiments, we average the estimates and calculate their NRMSEs over 1,000 runs.

3) Accuracy Results

Results of inferring 3-node motif concentrations: Table 2 shows the real values of the 3-node undirected and directed

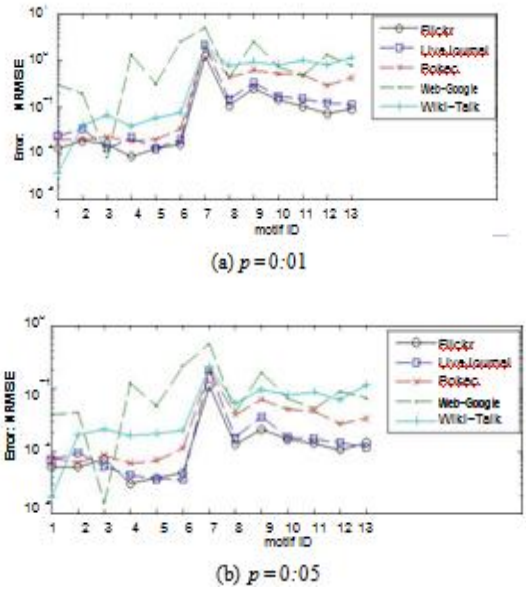


Figure 7. NRMSEs of $\hat{\omega}_i^{(3)}$, the concentration estimates of 3-node directed motifs for p = 0.01, 0.05.

Table 2 - Values of $\omega_i^{(3)}$, the concentrations of 3-node undirected and directed motifs. (*i* is the motif ID.)

i	Flickr	Pokec	Live-Journal	Wiki-Talk	Web-Google
3-node undirected motifs					
1	9.60e-01	9.84e-01	9.55e-01	9.99e-01	9.81e-01
2	4.04e-02	1.60e-02	4.50e-02	7.18e-04	1.91e-02
3-node directed motifs					
1	2.17e-01	1.77e-01	7.62e-02	8.91e-01	1.27e-02
2	6.04e-02	1.11e-01	4.83e-02	4.04e-02	1.60e-02
3	1.28e-01	1.60e-01	3.28e-01	3.91e-03	9.28e-01
4	2.44e-01	1.74e-01	1.14e-01	5.43e-02	3.09e-03
5	1.31e-01	1.91e-01	1.73e-01	5.48e-03	1.92e-02
6	1.80e-01	1.71e-01	2.15e-01	3.88e-03	1.92e-03
7	5.69e-05	7.06e-05	2.74e-05	1.37e-05	4.91e-05
8	6.52e-03	2.49e-03	8.66e-03	1.81e-04	6.82e-03
9	1.58e-03	1.03e-03	1.06e-03	8.42e-05	2.84e-04
10	5.19e-03	1.91e-03	6.63e-03	1.28e-04	2.77e-03
11	6.46e-03	2.03e-03	6.27e-03	8.03e-05	5.98e-03
12	1.07e-02	5.13e-03	9.82e-03	1.78e-04	1.21e-03
13	9.86e-03	3.45e-03	1.26e-02	6.65e-05	2.00e-03

Table 3 - NRMSEs of $\hat{\omega}_i^{(3)}$, the concentration estimates of 3-node undirected motifs for p = 0.01, 0.05

i	Flickr	Pokec	Live-Journal	Wiki-Talk	Web-Google
p = 0:01					
1	1.92e-03	3.26e-03	2.69e-03	5.21e-03	2.93e-04
2	4.56e-02	6.92e-02	1.64e-01	2.67e-01	4.00e-01
p = 0:05					
1	2.90e-04	4.10e-04	2.64e-04	6.06e-04	2.92e-05
2	6.90e-03	8.68e-03	1.61e-02	3.11e-02	3.99e-02

Table 4 - Values of $\omega_i^{(3)}$, the concentrations of 3-node signed motifs. (*i* is the motif ID.)

i	sign-Epinions	sign-Slashdot08	sign-Slashdot09
1	6.69e-01	6.58e-01	6.68e-01
2	2.12e-01	2.32e-01	2.25e-01
3	9.09e-02	1.02e-01	9.96e-02
4	2.29e-02	5.86e-03	5.75e-03
5	2.76e-03	9.74e-04	9.34e-04
6	2.49e-03	1.14e-03	1.13e-03
7	3.81e-04	1.80e-04	1.76e-04

motifs' concentrations for the undirected graphs and directed graphs of Flickr, Pokec, LiveJournal, Wiki-Talk, and Web-Google, which have 1.35×10^{10}, 2.02×10^{9}, 6.90×10^{9}, 1.2×10^{10}, and 7.00×10^{8} 3-node CISes respectively. Among all 3-node directed motifs, the 7th motif exhibits the smallest concentration for all these five directed graphs. Here the undirected graphs are obtained by discarding the edge directions of directed graphs. Table 3 shows the NRMSEs of our estimates of 3-node undirected motifs' concentrations for p = 0.01 and p = 0.05 respectively. On average $p \times 100\%$ of edges in the original graph are kept in the RESampled graph. We observe that the NRMSEs associated with the sampling probability p = 0.05 is about ten times smaller than the NRMSEs when p = 0.01. The NRMSEs are smaller than 0.04 when p = 0.05 for all five graphs. Fig. 7 shows the NRMSEs of our estimates of 3-node directed motifs' concentrations for p = 0.01 and p = 0.05 respectively. Similarly, we observe the NRMSEs when p = 0.05 are nearly ten times smaller than the NRM-SEs when p = 0.01. The NRMSE of our estimates of $\omega_7^{(3)}$ (i.e., the 7th 3-node directed motif concentration) exhibits the largest error. Except for $\omega_7^{(3)}$, the NRMSEs of the other motif concentrations' estimates are smaller than 0.2 when p = 0.05. Web-Google exhibits larger errors than the other graphs, because it has less 3-node CISes. Table 4 shows the real values of 3-node signed motifs' concentrations for sign-Epinions, sign-Slashdot08, and sign-Slashdot09, which have 1.72×10^{8}, 6.72×10^{7}, and 7.25×10^{7} 3-node CISes respectively. Fig. 8 shows the NRMSEs of our estimates of 3-node signed and undirected motifs' concentrations for p = 0.05 and p = 0.1 respectively. For all these three signed graphs, the NRMSEs are smaller than 0.9 and 0.2 when p = 0.05 and p = 0.1 respectively.

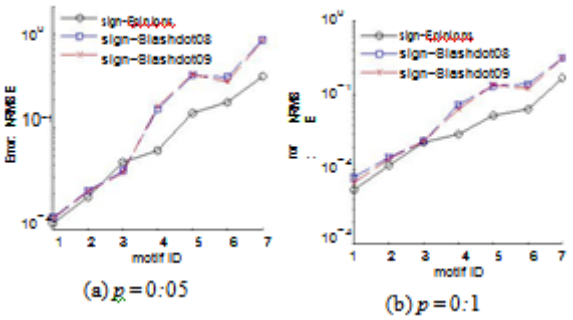


Figure 8. NRMSEs of $\omega_i^{(3)}$, the concentration estimates of 3-node signed motifs for p = 0.05, 0.1

Results of inferring 4-node motif concentrations: Table 5 shows the real values of $\omega_i^{(4)}$, i.e., the concentrations of 4-node undirected motifs for Flickr, Pokec, and LiveJournal, which have 4.52×10^{13}, 1.22×10^{12}, and 7.93×10^{12} 4-node CISes respectively. Fig. 9 shows the NRMSEs of $\hat{\omega}_i^{(4)}$, the concentration estimates of 4-node undirected motifs for p = 0.05 and p = 0.2 respectively. We observe that motifs with smaller $\omega_i^{(4)}$ exhibit larger NRMSEs. The NRMSEs of all the motif concentration estimates are smaller than 0.1 for p = 0.2.

Table 5 - Values of $\omega_i^{(4)}$, the concentrations of 4-node undirected motifs. (i is the motif ID.)

i	Flickr	Pokec	LiveJournal
1	1.45e-01	1.45e-01	2.92e-01
2	8.46e-01	8.35e-01	6.19e-01
3	2.95e-04	6.30e-04	4.84e-03
4	7.78e-03	1.57e-02	7.55e-02
5	3.84e-04	2.10e-03	8.23e-03
6	3.53e-05	1.26e-03	8.85e-04

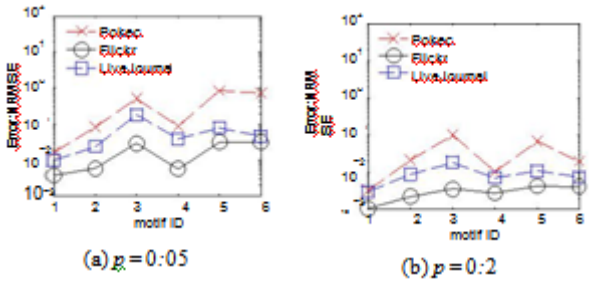


Figure 9. NRMSEs of $\hat{\omega}_i^{(4)}$, the concentration estimates of 4-node undirected motifs for p = 0.05, 0.2.

Results of inferring 5-node motif concentrations: Table 6 shows the real values of $\omega_i^{(5)}$, i.e., the concentrations of 5-node undirected motifs for com-Amazon, com-DBLP, p2p-Gnutella08, ca-GrQc, ca-CondMat, and ca-HepTh, which have 8.50×10^9, 3.34×10^{10}, 3.92×10^8, 3.64×10^7, 3.32×10^9, and 8.73×10^7 5-node CISes respectively. Fig. 10 shows the NRMSEs of $\hat{\omega}_i^{(5)}$, the concentration estimates of 5-node undirected motifs for p = 0.1, p = 0.2, and p = 0.3 respectively. We observe that NRMSE decreases as P increases, and the 6th, 10th, 13th, 17th, 18th 5-node motifs with small $\omega_i^{(5)}$ exhibit large NRMSEs. We generate a large graph G consisting of R soc-Amazon graphs, i.e., G has R components, and each component is an instance of the soc-Amazon graph. Clearly G has the same motif distributions as the soc-Amazon graph. Fig. 11 shows that the NRMSEs decreases as R increases, which indicates that our methods may exhibit small errors for characterizing 5-node motif of large graphs.

4) Error Bounds

Figure 12 shows the root CRLBs (RCRLBs) and the root MSEs (RMSEs) of our estimates of 3-node directed motifs' concentrations, 4-, and 5-node undirected motifs' concentrations, where graphs LiveJournal, soc-Epinions, and com-DBLP are used for studying 3-node directed motifs, 4-, and 5-node undirected motifs respectively. We observe that the RCRLBs are smaller than the RMSEs, and fairly close to the RMSEs. The RMSEs and RCRLBs are almost indistinguishable for 3-node directed motifs, where p = 0.01 and LiveJournal is used. It indicates that the RCRLBs can efficiently bound the errors of our motif concentration estimations.

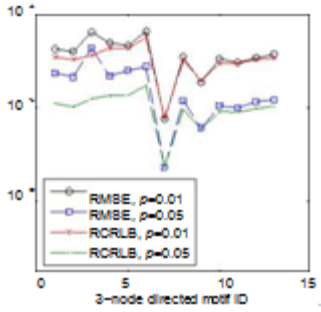
Table 6 - Values of $\omega_i^{(5)}$, concentrations of 5-node undirected motifs. (i is the motif ID.)

i	com-Amazon	com-DBLP	p2p-Gnutella08	ca-GrQc	ca-CondMat	ca-HepTh
1	2.9e-2	1.4e-1	2.6e-1	9.8e-2	1.4e-1	2.6e-1
2	7.5e-1	1.8e-1	1.8e-1	5.2e-2	2.2e-1	8.2e-2
3	1.6e-1	4.4e-1	4.6e-1	2.1e-1	4.3e-1	4.4e-1
4	6.0e-3	4.8e-2	1.1e-2	1.0e-1	4.9e-2	6.0e-2
5	2.3e-3	1.1e-3	2.7e-2	1.4e-3	2.1e-3	5.4e-3
6	3.6e-5	5.0e-5	1.4e-3	9.2e-5	1.1e-4	4.1e-4
7	1.5e-2	5.6e-2	2.7e-2	1.1e-1	5.5e-2	6.4e-2
8	3.5e-2	7.9e-2	2.2e-2	1.2e-1	8.0e-2	5.2e-2
9	1.4e-3	4.2e-3	1.4e-3	1.5e-2	7.0e-3	8.4e-3
10	1.7e-4	1.4e-4	1.0e-3	6.5e-4	3.0e-4	8.0e-4
11	7.3e-3	8.1e-3	4.3e-3	2.3e-2	9.9e-3	1.0e-2
12	5.3e-4	6.4e-3	2.8e-4	2.3e-2	4.5e-3	3.6e-3
13	8.2e-5	3.5e-6	7.4e-4	4.5e-6	6.4e-6	3.5e-5
14	3.9e-4	5.2e-4	1.7e-4	2.8e-3	6.6e-4	1.0e-3
15	6.7e-4	2.6e-2	7.6e-5	1.5e-1	5.9e-3	5.3e-3
16	7.1e-4	3.4e-4	1.4e-4	1.4e-3	9.2e-4	4.4e-4
17	3.9e-5	1.1e-5	8.0e-5	4.3e-5	2.9e-5	8.4e-5
18	2.3e-5	4.9e-6	6.0e-6	2.3e-5	8.5e-6	3.0e-5
19	2.4e-4	2.8e-3	1.5e-5	1.9e-2	9.8e-4	5.8e-4
20	5.8e-5	4.2e-4	7.0e-7	8.0e-3	1.4e-4	8.2e-5
21	7.2e-6	7.9e-3	1.5e-8	6.1e-2	1.5e-4	3.2e-3

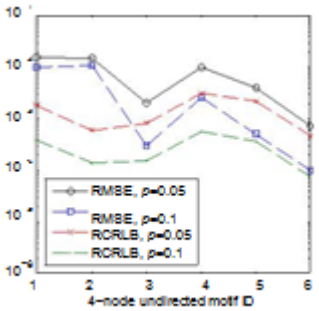
5) Running Time

In this subsection, we describe the experimental results of our 3-, 4-, and 5-node CIS enumeration methods in Section 4. In order to evaluate the performance of our methods, we present results on both synthetic and real-world graphs. Our experiments are implemented on an AWS EC2 cluster consisting of up to 12 r3.4xlarge ma-chines, where each machine has 122GB RAM and 16 virtual CPUs. To the best of our knowledge, we are the first to enumerate directed/signed CISes in a distributed manner. Thus, we only compare our methods with the state-of-the-art *multicore* method gtrieScanner [30] for counting directed/signed motifs. The results demonstrate that our methods outperform gtrieScanner on a single machine with 16 cores in terms of computational time and memory usage. Comparison to gtrieSanner on a single machine: We implement our method and gtrieScanner

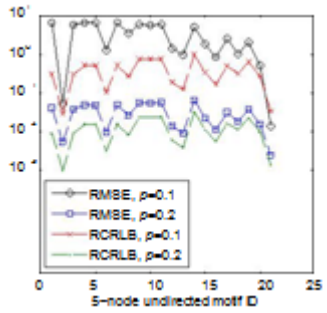
both in a single AWS EC2 machine. We generate graphs with different sizes using the Erdos¨ Renyi´ model $G(n, p\hat{})$ [31] , where n is the number of nodes. On average, a directed graph in $G(n, p\hat{})$ has $n(n-1)p$ edges. In our experiments, we set $np\hat{} = 5$. gtrieScanner fails to process graphs with $n \geq 400,000$ nodes because it stores graphs as regular matrices, which is prohibited for large graphs. Therefore, we compare our method and gtrieScanner on graphs with 50,000 to 400,000



(a) (LiveJournal) 3-node directed motifs

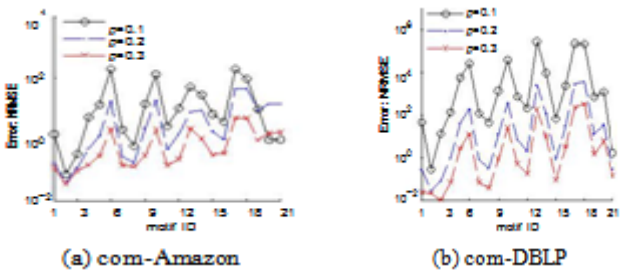


(b) (soc-Epinions) 4-node undirected motifs



(c) (com-DBLP) 5-node undirected motifs

Figure 10. RCRLBs and RMSEs of concentration estimates of 3, 4, and 5-node directed motifs.



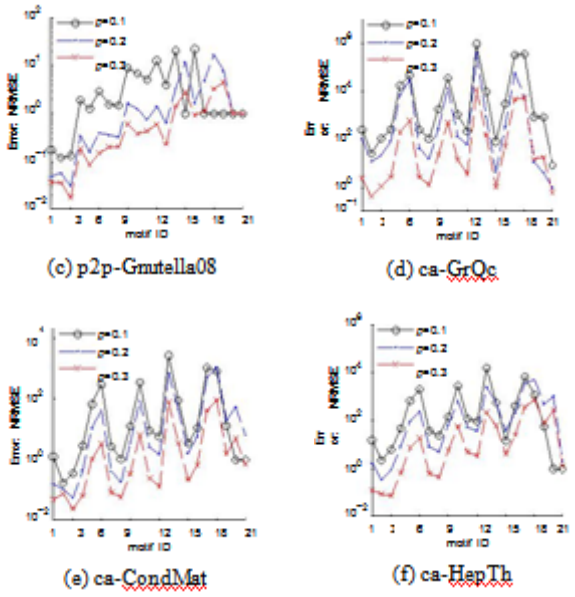


Figure 11. NRMSEs of $\omega_i\hat{}^{(5)}$, the concentration esti-mates of 5-node undirected motifs for $p = 0.1, 0.2, 0.3$.

nodes. Figs. 13(a)-(c) show the scalability of our method for processing graphs with different sizes. We can see that its time complexity almost increases linearly with the graph size. Compared to gtrieScanner, our method is 15× and 5× faster of counting 3- and 4-node directed motifs reflectively, and almost have the same speed for counting 5-node directed motifs. Fig. 13(d) shows that the memory usage of our method increases linearly with the graph size, while the memory usage of gtrieScanner grows quadratically with the graph size and cannot load in a graph with more than 400,000 nodes in a single AWS EC2 machine. Notice that our method uses $\frac{1}{120}$, $\frac{1}{30}$, and $\frac{1}{3}$ memory to gtrieScanner for counting 3-, 4-, and 5-node directed motifs reflectively, which is more suitable for large graphs. We omit the similar results for enumerating signed motifs.

Running time on a cluster of machines: Next, we evaluate the scalability of our method on a large graph LiveJournal with millions of nodes and edges for different number of machines and sampling rate p. Fig. 14 shows the running time for counting 3-, 4- and 5-node CISes as a function of sample rate and number of machines. We can see the running time decreases when the number of machines grows. Benefitting from the edge sampling, we speed up by orders of magnitude for estimating 3-, 4-, and 5-node motifs without significantly sacrificing accuracy, which has been discussed previously. By setting different p, we study the relationship between the running time and the estimation error of our method. The result is shown in Fig. 15. We can see that the average NRMSE grows as p decreases, while the running time decreases dramatically as p decreases.

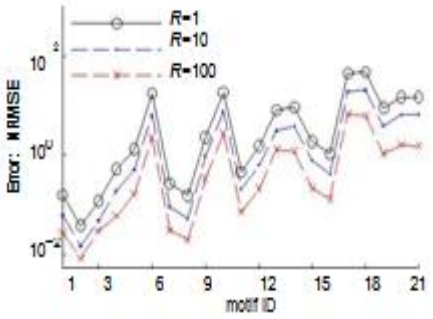


Figure 12. NRMSEs of $\omega^{\wedge}_{i}{}^{(5)}$, the concentration esti-mates of 5-node undirected motifs for a graph consist-ing of R soc-Amazon graphs, where p = 0.2.

VI. RELATED WORK

Motif Sampling Methods: There has been considerable interest in designing efficient sampling methods for count-ing specific subgraph patterns such as triangles [15], [32]– [35], cliques [36], [37], and cycles [32], [38], because it is computationally intensive to compute the number of the

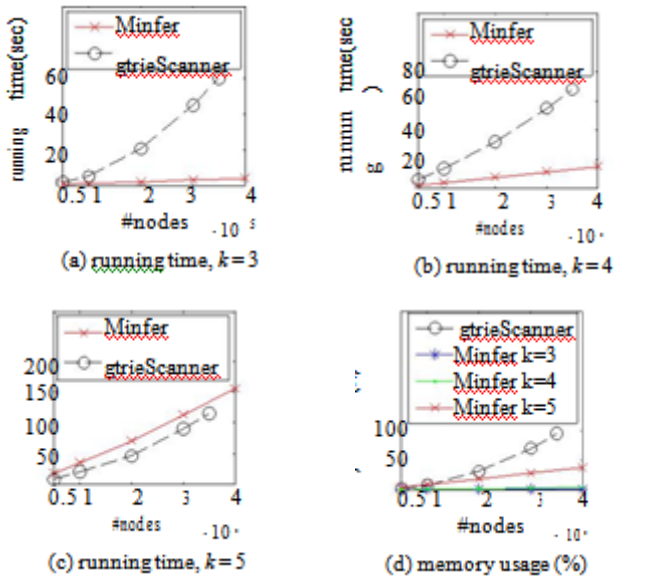
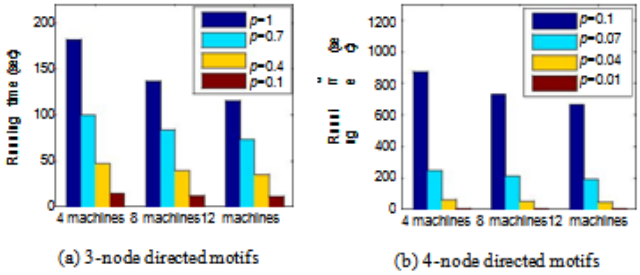


Figure 13. Compared results of our methods and gtrieScanner for enumerating the k-node directed CISes of graphs with different sizes in a single machine with 122GB memory. gtrieScanner fails to process graphs with 400, 000 nodes.



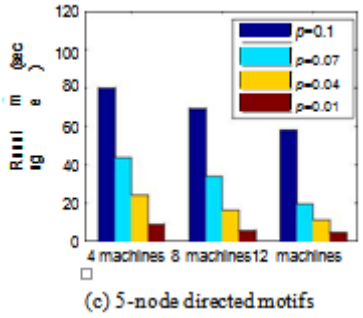


Figure 14. (LiveJournal) Running time of enumerating 3-, 4- and 5-node motifs in directed graph for different number of machines and sample rate p.

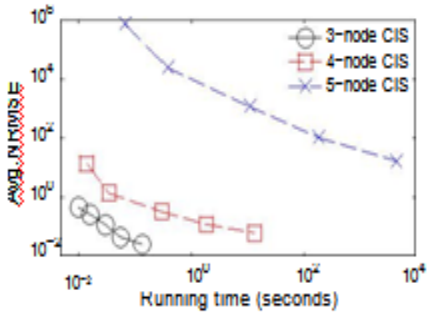


Figure 15. (soc-Epinions1) Running time vs. average NRMSE of concentration estimates of all 3-, 4-, and 5-node motifs with $p \in \{0.01, 0.02, 0.05, 0.1, 0.2\}$.

subgraph pattern's appearances in a large graph. Similar to the problem studied in [10]–[13], [39], in this work we focus on characterizing 3-, 4-, and 5-node CISes in *a single large graph*, which differs from the problem of estimating the number of subgraph patterns appearing in *a large set of graphs* studied in [40]. OmidiGenes et al. [39] proposed a subgraph enumeration and counting method using sampling. However this method suffers from an unknown sampling bias. To estimate subgraph class concentrations, Kashtan et al. [11] proposed a subgraph sampling method, but their method is computationally expensive when calculating the weight of each sampled subgraph, which is needed to correct for the bias introduced by sampling. To address this drawback, Wernicke [12] proposed an algorithm, FANMOD, based on enumerating subgraph trees to detect network motifs. Bhuiyan et al. [13] proposed a Metropolis-Hastings based sampling method GUISE to estimate 3-node, 4-node, and 5-node subgraph frequency distribution. Wang et al. [10] proposed an effi-cient crawling method to estimate online social networks' motif concentrations, when the graph's topology is not available in advance and it is costly to crawl the entire topology. Work on graph sparsifiers such as [41] focuses on designing methods to obtain a sparse graph similar to the original graph with respect to a specific graph metric such as triangle count. We are interested in a different problem, to infer the original graph's motif statistics from a given RESampled graph and we assume the original graph is not available or cannot be explored. Triangle sparsifiers is used to estimate the original graph's triangle count. It cannot characterize 4- and 5-node motifs, and

3-node directed motifs, because it does not consider how to remove the uncertainty that a sampled 3-node CIS may differ from its original CISes. In summary, previous methods focus on designing efficient sampling methods and crawling methods for estimating motif statistics when the graph is directly available or indirectly available (i.e., it is not expensive to query a node's neighbors [10]). They cannot be applied to solve the problem studied in this paper, i.e., we assume the graph is not available but a RESampled graph is given and we aim to infer the underlying graph's motif statistics from the RESampled graph. At last, we would like to point out our method of estimating motif statistics and its error bound computation method are inspired by methods of estimating flow size distribution for network traffic measurement and monitoring [42]–[45].

Exact Motif Counting Methods: The problem of enumerating and counting all triangles in large undirected graphs has been well studied in both centralized and distributed set-tings [46]–[49]. Recently, many efforts have been devoted to designing efficient algorithms for exactly counting higher order subgraph patterns such as 4- and 5-node graphlets. [49]–[51] developed efficient algorithms for counting 3-and 4-node undirected motifs by utilizing the relationships between 3- and 4-node motif frequencies, which do not hold for labeled graphs such as directed and signed graphs studied in this paper. [30] developed a multi-core algorithm to enumerate both undirected and directed graphlets. For the distributed setting, [52] developed a framework PSgL for listing only undirected subgraphs in a divide-and-conquer fashion. Compared to existing methods, our experiments demonstrate that our methods are much more effective for counting 3-, 4-, and 5-node labeled (e.g., directed) motifs.

VII. CONCLUSIONS

In this paper, we study the problem of inferring the underlying graph's motif statistics when the entire graph topology is not available, and only a RESampled graph is given. We propose a model to bridge the gap between the underlying graph's and its RESampled graph's motif statistics. Based on this probabilistic model, we develop a method Minfer to infer the underlying graph's motif statistics, and give a Fisher information based method to bound the error of our estimates. We further develop new methods for enumerating and classifying 3-, 4-, and 5-node labeled (e.g., directed or signed) CISes under both centralized and distributed settings. Experimental results on a variety of real world datasets demonstrate the efficiency of our methods.

VIII. ACKNOWLEDGMENT

We thank the anonymous reviewers as well as Dr. Wei Fan for helpful suggestions. The research presented in this paper is supported in part by National Natu-ral Science Foundation of China (U1301254, 61603290, 61602371), the Ministry of Education&China Mobile Re-search Fund (MCM20160311), the Natural Science Foun-dation of JiangSu Province (SBK2014021758), 111 Inter-national Collaboration Program of China, the Prospective Joint Research of Industry-Academia-Research Joint Inno-vation Funding of Jiangsu Province (BY2014074), Shen-zhen Basic Research Grant (JCYJ20160229195940462), China Postdoctoral Science Foundation (2015M582663), Natural Science Basic Research Plan in Shaanxi Province of China (2016JQ6034).

References

[1] H. Chun, Y. yeol Ahn, H. Kwak, S. Moon, Y. ho Eom, and H. Jeong, "Comparison of online social relations in terms of volume vs. interaction: A case study of cyworld," in *IMC 2008*, pp. 57–59.

[2] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The slashdot zoo: mining a social network with negative edges," in *WWW 2009*, pp. 741–750.

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

[4] J. Ugander, L. Backstrom, and J. Kleinberg, "Subgraph frequencies: mapping the empirical and extremal geography of large graph collections," in *WWW 2013*, pp. 1307–1318.

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

[6] I. Albert and R. Albert, "Conserved network motifs allow protein–protein interaction prediction," *Bioinformatics*, vol. 4863, no. 13, pp. 3346–3352, 2004.

[7] S. Itzkovitz, R. Levitt, N. Kashtan, R. Milo, M. Itzkovitz, and U. Alon, "Coarse-graining and self-dissimilarity of complex net-works," *Physica Rev.E*, vol. 71, p. 016127, 2005.

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

[9] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, July 2016.

[10] P. Wang, J. C. Lui, J. Zhao, B. Ribeiro, D. Towsley, and X. Guan, "Efficiently estimating motif statistics of large networks," *ACM TKDD*, vol. 9, no. 2, Nov. 2014.

[11] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon, "Efficient sampling algorithm for estimating subgraph concentrations and detecting net-work motifs," *Bioinformatics*, vol. 20, no. 11, pp. 1746–1758, 2004.

[12] S. Wernicke, "Efficient detection of network motifs," *IEEE/ACM TCBB*, vol. 3, no. 4, pp. 347–359, 2006.

[13] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. A. Hasan, "Guise: Uniform sampling of graphlets for large graph analysis," in *ICDM 2012*, pp. 91–100.

[14] B. Ribeiro and D. Towsley, "Estimating and sampling graphs with multidimensional random walks," in *IMC 2010*, pp. 390–403.

[15] N. Ahmed, N. Duffield, J. Neville, and R. Kompella, "Graph sample and hold: A framework for big-graph analytics," in *SIGKDD 2014*, pp. 589–597.

[16] J. Chen, W. Hsu, M.-L. Lee, and S.-K. Ng, "Nemofinder: dissecting genome-wide protein-protein interactions with meso-scale network motifs," in *SIGKDD 2006*, pp. 106–115.

[17] H. L. van Trees, *Estimation and Modulation Theory, Part 1*. New York: Wiley, 2001.

[18] B. D. McKay, "nauty user's guide, version 2.4," Department of Computer Science, Australian National University, Tech. Rep., 2009.

[19] S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," in *WWW 2011*, pp. 607–614.

[20] T. Schank, "Algorithmic aspects of triangle-based network analysis," Ph.D. dissertation, 2007.

[21] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "Powergraph: Distributed graph-parallel computation on natural graphs," in *OSDI 2012*, pp. 17–30.

[22] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *IMC 2007*, pp. 29–42.

[23] L. Takac and M. Zabovsky, "Data analysis in public social networks." in *International Scientific Conference and International Workshop Present Day Trends of Innovations*, 2012, pp. 1–6.

[24] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *WWW 2010*, pp. 641–650.
[25] "Google programming contest," http://www.google.com/ programming-contest/, 2002.
[26] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *CHI 2010*, pp. 1361–1370.
[27] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *ICDM 2012*, pp. 745–754.
[28] M. Ripeanu, I. T. Foster, and A. Iamnitchi, "Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," *IEEE Internet Computing Journal*, vol. 6, no. 1, pp. 50–57, 2002.
[29] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM TKDD*, vol. 1, no. 1, Mar. 2007.
[30] D. Oliveira Aparicio, P. M. Pinto Ribeiro, and F. M. A. D. Silva, "Parallel subgraph counting for multicore architectures," in *IPDPS 2014*, pp. 34–41.
[31] P. Erdos¨ and A. Renyi,´ "On random graphs i," *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959.
[32] N. Alon, R. Yuster, and U. Zwick, "Color-coding," *J. ACM*, vol. 42, no. 4, pp. 844–856, Jul. 1995.
[33] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos, "Doulion: Counting triangles in massive graphs with a coin," in *SIGKDD 2009*, pp. 837–846.
[34] A. Pavany, K. T. S. Tirthapuraz, and K.-L. Wu, "Counting and sampling triangles from a graph stream," in *PVLDB 2013*, pp. 1870– 1881.
[35] M. Jha, C. Seshadhri, and A. Pinar, "A space efficient streaming algorithm for triangle counting using the birthday paradox," in *SIGKDD 2013*, pp. 589–597.
[36] J. Cheng, Y. Ke, A. W.-C. Fu, J. X. Yu, and L. Zhu, "Finding maximal cliques in massive networks," *ACM TODS*, vol. 36, no. 4, 21:1–21:34, Dec. 2011.
[37] M. Gjoka, E. Smith, and C. T. Butts, "Estimating Clique Compo-sition and Size Distributions from Sampled Network Data," *ArXiv e-prints*, Aug. 2013.
[38] M. Manjunath, K. Mehlhorn, K. Panagiotou, and H. Sun, "Approx-imate counting of cycles in streams," in *ESA 2011*, pp. 677–688.
[39] S. Omidi, F. Schreiber, and A. Masoudi-nejad, "Moda: An efficient algorithm for network motif discovery in biological networks," *Genes and Genet systems*, vol. 84, no. 5, pp. 385–395, 2009.
[40] M. A. Hasan and M. J. Zaki, "Output space sampling for graph patterns," in *PVLDB 2009*, pp. 730–741.
[41] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller, "Triangle sparsifiers," *J. Graph Algorithms Appl.*, vol. 15, no. 6, pp. 703–726, 2011.
[42] N. Duffield, C. Lund, and M. Thorup, "Estimating flow distributions from sampled flow statistics," in *SIGCOMM 2003*, pp. 325–336.
[43] B. Ribeiro, D. Towsley, T. Ye, and J. Bolot, "Fisher information of sampled packets: an application to flow size estimation," in *IMC 2006*, pp. 15–26.
[44] P. Tune and D. Veitch, "Towards optimal sampling for flow size estimation," in *IMC 2008*, pp. 243–256.
[45] P. Wang, X. Guan, J. Zhao, J. Tao, and T. Qin, "A new sketch method for measuring host connection degree distribution," *IEEE TIFS*, vol. 9, no. 6, pp. 948–960, 2014.
[46] S. Chu and J. Cheng, "Triangle listing in massive networks and its applications," in *SIGKDD 2011*, pp. 672–680.
[47] S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," in *WWW 2011*, pp. 607–614.
[48] T. Schank, "Algorithmic aspects of triangle-based network analysis," *Phd in Computer Science*, 2007.
[49] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Di-makis, "Beyond triangles: A distributed framework for estimating 3-profiles of large graphs," in *SIGKDD 2015*, pp. 229–238.
[50] D. Marcus and Y. Shavitt, "Rage–A rapid graphlet enumerator for large networks," *Computer Networks*, vol. 56, no. 2, pp. 810–819, 2012.
[51] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Di-makis, "Distributed estimation of graph 4-profiles," in *WWW 2015*, 2–14.
[52] Y. Shao, B. Cui, L. Chen, L. Ma, J. Yao, and N. Xu, "Parallel subgraph listing in a large-scale graph," in *SIGMOD 2014*, pp. 625– 636.