

Fast crawling methods of exploring content distributed over large graphs

Pinghui Wang^{1,2} · Junzhou Zhao³ · John C. S. Lui⁴ · Don Towsley⁵ · Xiaohong Guan^{1,6}

Received: 2 June 2017 / Revised: 27 February 2018 / Accepted: 9 March 2018 /
Published online: 15 March 2018
© Springer-Verlag London Ltd., part of Springer Nature 2018

Abstract Despite recent effort to estimate topology characteristics of large graphs (e.g., online social networks and peer-to-peer networks), little attention has been given to develop a formal crawling methodology to characterize the vast amount of content distributed over these networks. Due to the large-scale nature of these networks and a limited query rate imposed by network service providers, exhaustively crawling and enumerating content maintained by each vertex is computationally prohibitive. In this paper, we show how one can obtain content properties by crawling only a small fraction of vertices and collecting their content. We first show that when sampling is naively applied, this can produce a huge bias in content statistics (i.e., average number of content replicas). To remove this bias, one may use maximum likelihood estimation to estimate content characteristics. However, our experimen-

✉ Junzhou Zhao
junzhouzhao@gmail.com

Pinghui Wang
phwang@mail.xjtu.edu.cn

John C. S. Lui
cslui@cse.cuhk.edu

Don Towsley
towsley@cs.umass.edu

Xiaohong Guan
xhguan@mail.xjtu.edu.cn

¹ MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, China

² Shenzhen Research Institute of Xi'an Jiaotong University, Shenzhen, China

³ Division of Computer, Electrical and Mathematical Sciences and Engineering, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

⁴ Department of Computer Science and Engineering, The Chinese University of Hong Kong, Sha Tin, Hong Kong

⁵ Department of Computer Science, University of Massachusetts Amherst, Amherst, MA, US

⁶ Center for Intelligent and Networked Systems, Tsinghua University, Beijing, China

tal results show that this straightforward method requires to sample most vertices to obtain accurate estimates. To address this challenge, we propose two efficient estimators: special copy estimator (SCE) and weighted copy estimator (WCE) to estimate content characteristics using available information in sampled content. SCE uses the special content copy indicator to compute the estimate, while WCE derives the estimate based on meta-information in sampled vertices. We conduct experiments on a variety of real-word and synthetic datasets, and the results show that WCE and SCE are cost effective and also “asymptotically unbiased”. Our methodology provides a new tool for researchers to efficiently query content distributed in large-scale networks.

Keywords Crawling · Online social networks · Sampling · Random walks

1 Introduction

Nowadays online social networks (OSNs) (e.g., Facebook and Twitter) and content sharing networks (e.g., BitTorrent) are two popular classes of Internet applications. Measuring content characteristics such as file duplication level and information spreading rate on such networks become important since it helps one develop effective advertising strategies [46] and provides valuable information for designing content delivery strategies, e.g., video sharing techniques [28] to increase video pre-fetch accuracy by delivering videos based on users’ social relationships and interests, or to develop information seeding techniques [31] to minimize the peak load of cellular networks by proactively pushing some videos to OSN users. Meanwhile, measuring characteristics of OSNs’ content provided by other networks also helps to understand interactions between different networks. For example, [28] found that 80% of videos in Facebook come from other video service providers such as YouTube.

Due to a limited query (or crawling) rate imposed by network service providers, it is not feasible to collect all content and one must consider using sampling methods to explore content distributed over large networks, because crawling the entire networks is computationally expensive. To solve this problem, one can directly apply sampling methods to content, or it can also collect content from a relatively small fraction of users. Unfortunately, in practice one may not be able to sample content in large networks in a direct manner. For example, for file sharing networks such as BaiduWangPan (in English: Baidu web disk, <https://pan.baidu.com>) where a node (or user) maintains a list of files, they support users to randomly explore nodes but not files. Similarly, directly sampling tweets on Sina microblog (<http://weibo.com>) is not feasible. Even when content has a profile webpage, the profile IDs (numeric numbers or strings) of content might be sparsely populated in the ID space, so it is expensive to generate a random yet valid profile ID. Moreover, there may neither have no links between these profile webpages of content. This disables us to directly use crawling methods such as random walks to sample content at random. For example, suppose we want to study statistics of check-in places on Facebook. A check-in place with string name *placename* has a profile webpage (<https://www.facebook.com/placename>) on Facebook, and there exist no links between profile webpages of check-in places. In the above scenarios, sampling content requires to first sample a fraction of nodes and then collect their content.

Unfortunately, previous graph sampling work developed for estimating degree or workplace distributions [14, 40] cannot be directly applied in our context. This is because content and vertex are intrinsically different since content may be *duplicated*. To illustrate this, consider a simple example of a P2P network as shown in Fig. 1. Assume there are two types of files: video and image, each video file such as F_1 is cached by a large number of users, each

Fig. 1 An example of sampling files in a P2P network

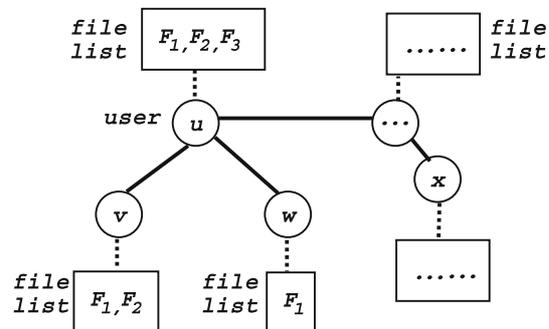


image file such as F_3 is cached by a small number of users, and we want to estimate the fraction of distinct video and image files in the P2P network. Clearly, one will overestimate the fraction of video files because video files are more likely to be sampled than image files even when using uniform node sampling. Therefore, methods for estimating topological metrics such as degree distribution cannot be blindly applied. They need to be modified to deal with biases introduced because of the nature of content characterization.

To the best of our knowledge, our work is the first analytical and qualitative study on the problem of estimating characteristics of content distributed over large graphs. We propose methods to accurately estimate content statistics. Our contributions are summarized as:

- We demonstrate that it is challenging to efficiently explore characteristics of content distributed over large graphs, because: (1) when sampling is naively applied, there can be huge bias in computing content statistics; (2) although one can remove this bias using the maximum likelihood estimation, our experimental results show that one needs to sample most vertices in the graph in order to obtain an accurate estimation.
- To solve the above problem, we present two accurate methods to estimate content statistics using available meta-information in sampled content based on two different assumptions, respectively. The first method assumes that we can determine whether a collected content copy is a source or not, which is true for most OSNs that have a label in each content copy to indicate whether it is a source or a duplicated copy. For example, a tweet in microblog networks can be classified as an original tweet or a retweet. Therefore, one can utilize a special content copy set consisting of all original tweets to characterize tweets in the network. To measure statistics of content in these networks, we propose a special copy estimator (SCE) based on collected source content copies. Moreover, we propose another weighted copy estimator (WCE) for networks (e.g., P2P networks) that content (e.g., video) copies have no difference and no meta-data can be used as a special copy indicator. It assumes that each content copy records the number of copies its content holds. This feature is true for many OSNs and P2P networks [52]. Our experimental results show that WCE and SCE are asymptotically unbiased, and WCE is more accurate than SCE, because SCE uses only a fraction of sampled content copies.
- We also use WCE to estimate graph structure statistics for OSNs such as Sina Microblog and Xiami (<http://www.xiami.com>) where users maintain graph property summaries of their neighbors. For example, by crawling the profile of a user in Sina Microblog, we can obtain its neighbors' properties such as the number of followers, the number of following, the number of tweets. This allows us to collect more information than previous graph sampling methods under the same sampling cost. Since a user's graph property summary can be viewed as content maintained by itself and its neighbors, we apply WCE to estimate

graph statistics. Our experimental results show that WCE obtains the same accuracy with a *much less* sampling cost as compared with previous works.

This paper is organized as follows. Section 2 introduces preliminaries. Section 3 presents our crawling methods for estimating characteristics of content in large networks. The performance evaluation and testing results are presented in Sect. 4. Section 5 presents real applications on Sina Microblog Web sites. Related work is given in Sect. 6, and conclusion is given in Sect. 7.

2 Preliminaries

In this section, we briefly review popular graph sampling methods, which are underlying techniques for sampling vertices and content in large networks discussed in Sect. 3. For ease of presentation, we assume the underlying graph is undirected. One way to convert a directed graph into an undirected graph is by ignoring the direction of edges. Consider an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of undirected edges. Breadth-first search (BFS) is one of most popular graph sampling techniques. However, it introduces a large bias toward high-degree vertices that is unknown and difficult to remove in general graphs [1, 25]. Therefore, we do not consider BFS in this paper. In what follows, we present four popular graph sampling methods: *uniform vertex sampling* (UNI), *random walk* (RW), *Metropolis–Hastings RW* (MHRW) [45], and *Frontier sampling* (FS) [40]. Unless we state otherwise, we denote $\pi = (\pi_v : v \in V)$ as the probability distribution for the underlying sampling method, where π_v is defined as the probability of sampling vertex v at each step.

2.1 Sampling method I: uniform vertex sampling

UNI randomly samples vertices from V with replacement. Not all networks support UNI but some do. For example, one can view Wikipedia as a graph and Wikipedia provides a query API to obtain a random vertex (wiki page) sampled from its entire vertices. At each step, UNI samples each vertex v with the same probability, so we have

$$\pi_v^{\text{UNI}} = \frac{1}{|V|}, \quad v \in V.$$

For OSNs such as Facebook, MySpace, Flickr, Renren (<http://www.renren.com>), Sina Microblog, and Xiami, one can sample users (vertices) as users have numeric IDs. Unfortunately, ID values of users in many networks (e.g., Flickr, Facebook, Sina Microblog, and MySpace) are not sequentially assigned, and the ID space is sparsely populated [39, 41]. Hence, a randomly generated ID may not correspond to a valid user, so considerable computational effort in generating a random ID will be wasted. Therefore, UNI should only be applied to those graphs whose user ID values are densely packed.

2.2 Sampling method II: random walk

RW has been extensively studied in the graph theory literature [30]. From an initial vertex, a walker selects a neighbor at random as the next-hop vertex. The walker moves to this neighbor and repeats the process. Denote $\mathcal{N}(u)$ as the set of neighbors of any vertex u , $\text{deg}(u) = |\mathcal{N}(u)|$ is the degree of u . Formally, RW can be viewed as a Markov chain with transition matrix $P^{\text{RW}} = [P_{u,v}^{\text{RW}}]$, $u, v \in V$, where $P_{u,v}^{\text{RW}}$ is defined as the probability of vertex v being selected as the next-hop vertex given that its current vertex is u , we have:

$$P_{u,v}^{\text{RW}} = \begin{cases} \frac{1}{\deg(u)} & \text{if } v \in \mathcal{N}(u), \\ 0 & \text{otherwise.} \end{cases}$$

The stationary distribution π^{RW} of this Markov chain is

$$\pi_v^{\text{RW}} = \frac{\deg(v)}{2|E|}, \quad v \in V.$$

For a connected and non-bipartite graph G , the probability of being at a vertex $v \in V$ converges to the above stationary distribution [30]. We can see that π^{RW} is biased toward high-degree vertices, but this sampling bias can be removed when we estimate statistics of graph topologies such as node degree distribution [17, 42].

2.3 Sampling method III: Metropolis–Hastings random walk

MHRW [14, 45, 54] provides another way to modify RW using Metropolis–Hastings technique [10, 16, 32], which aims to collect vertices uniformly. To generate a sequence of random samples from a desired stationary distribution π^{MHRW} , the Metropolis–Hastings technique is a Markov chain Monte Carlo method based on modifying the transition matrix of RW as

$$P_{u,v}^{\text{MHRW}} = \begin{cases} P_{u,v}^{\text{RW}} \min\left(\frac{\pi_v^{\text{MHRW}} P_{v,u}^{\text{RW}}}{\pi_u^{\text{MHRW}} P_{u,v}^{\text{RW}}}, 1\right) & \text{if } v \in \mathcal{N}(u), \\ 1 - \sum_{w \neq u} P_{u,w}^{\text{MHRW}} & \text{if } v = u, \\ 0 & \text{otherwise.} \end{cases}$$

For a MHRW with target distribution $\pi^{\text{MHRW}} = \pi^{\text{UNI}}$, it works as follows: at each step, MHRW selects a neighbor v of current vertex u at random and then accepts the move randomly with probability $\min\left(\frac{\deg(u)}{\deg(v)}, 1\right)$. Otherwise, MHRW still remains at u . Essentially, MHRW removes the bias of RW at each step by rejecting moves toward high-degree vertices with a certain probability.

2.4 Sampling method IV: Frontier sampling

FS [40] is a centrally coordinated sampling which performs $T \geq 1$ dependent RWs in the graph G of interest. Compared to a single RW, FS is less likely to get stuck in a loosely connected component of G . Denote $\mathbf{L} = (v_1, \dots, v_T)$ as a vector with T vertices. Each v_i ($1 \leq i \leq T$) is initialized with a random vertex uniformly selected from V . At each step, FS selects a vertex $u \in \mathbf{L}$ with probability $\frac{\deg(u)}{\sum_{v \in \mathbf{L}} \deg(v)}$ and randomly selects a vertex w from $\mathcal{N}(u)$, i.e., the set of neighbors of u . One can easily find that w is uniformly selected from vertices that are connected to the vertices in $\mathbf{L} = (v_1, \dots, v_T)$. Then, FS replaces u by w in \mathbf{L} and adds w to the sequence of sampled vertices. If G is a connected and non-bipartite graph, the probability of FS sampling a vertex v converges to the following distribution

$$\pi_v^{\text{FS}} = \frac{\deg(v)}{2|E|}, \quad v \in V.$$

2.5 Horvitz–Thompson estimator

Previous work has considered how to estimate topology properties (e.g., degree distribution) via sampling methods. Let $L'(v)$ denote the vertex label of vertex $v \in V$ under study, with range $L' = \{l'_0, \dots, l'_{K'}\}$. Denote vertex label density $\tau = (\tau_0, \dots, \tau_{K'})$, where τ_k

($0 \leq k \leq K'$) is the fraction of vertices with label l'_k . For example, τ is the degree distribution when $L'(v)$ is defined as the degree of v . To estimate τ , RW- and FS-based sampling methods can be viewed as extensions of Horvitz–Thompson estimator [18], which applies inverse probability weighting to account for samples with different probabilities. Therefore, the stationary distribution π is needed to correct the bias introduced by the underlying sampling method. Since $|V|$ and $|E|$ are usually unknown, unbiasing the error is not straightforward. Instead, one may use a non-normalized stationary distribution $\hat{\pi} = (\hat{\pi}_v : v \in V)$ to reweight sampled vertices s_i ($1 \leq i \leq n$), where $\hat{\pi}_v$ is computed as

$$\hat{\pi}_v = \begin{cases} 1 & \text{for UNI and MHRW,} \\ \text{deg}(v) & \text{for RW and FS.} \end{cases} \quad (1)$$

Let $\mathbf{1}(\mathbf{P})$ define the indicator function that equals one when predicate \mathbf{P} is true, and zero otherwise. Finally, τ_k is estimated as

$$\hat{\tau}_k = \frac{1}{S} \sum_{i=1}^n \frac{\mathbf{1}(L'(s_i) = l'_k)}{\hat{\pi}_{s_i}}, \quad 0 \leq k \leq K',$$

where $S = \sum_{i=1}^n \hat{\pi}_{s_i}^{-1}$.

In summary, RW and FS are biased to sample high-degree vertices. Their biases can be later corrected when we compute topology properties [14, 38]. Compare the performances of RW and MHRW, and observe that RW is consistently more accurate than MHRW for exploring large networks. Compared to RW and MHRW, Ribeiro and Towsley [40] reveal that FS is more accurate for characterizing loosely connected and disconnect graphs.

3 Content sampling

Our problem To describe our methods formally, we first introduce some notations. For ease of reading, we list notations used throughout the paper in Table 1. Define $L(\mathbf{c})$ as a generic labeling function of content \mathbf{c} , with range $\mathbf{L} = \{l_0, \dots, l_K\}$. We focus on developing crawling methods for estimating the content distribution $\omega = (\omega_0, \dots, \omega_K)$, where ω_k ($0 \leq k \leq K$) is the fraction of content with label l_k . For example, $L(\mathbf{c})$ can be defined as the number of comments of post \mathbf{c} in OSNs, and then ω_k is the fraction of posts with k comments, where $\mathbf{L} = \{0, 1, 2, \dots\}$. In P2P networks, $L(\mathbf{c})$ can denote the number of replicas of file \mathbf{c} ; then ω_k is the fraction of files having k replicas, where $\mathbf{L} = \{1, 2, \dots\}$. Similarly $L(\mathbf{c})$ can also be the file type of \mathbf{c} in P2P networks, with $\mathbf{L} = \{l_0 = \text{"video"}, l_1 = \text{"music"}, l_2 = \text{"text"}, l_3 = \text{"others"}\}$. Then ω_k ($k = 0, 1, 2, 3$) is the fraction of files of type l_k . Denote by $\mathbf{C} = \{\mathbf{c}_1, \dots, \mathbf{c}_H\}$ the set of all content distributed over graph G , where H is the total number of distinct content in G . Then, the content distribution $\omega = (\omega_0, \dots, \omega_K)$ is defined as

$$\omega_k = \frac{\sum_{\mathbf{c} \in \mathbf{C}} \mathbf{1}(L(\mathbf{c}) = l_k)}{H}, \quad k = 0, \dots, K,$$

where $L(\mathbf{c})$ is the label of content \mathbf{c} with range $\{l_0, \dots, l_K\}$. We assume that n vertices s_1, \dots, s_n are obtained by a graph sampling method (e.g., UNI, RW, MHRW, or FS) with vertex sampling probability distribution π . Denote by $C(v)$ the set of the content copies maintained by a vertex $v \in V$. Later in Sect. 3.1, we will present two methods DCE and MLE and show that it is difficult to estimate ω solely from sampled content copies $C(s_1), \dots, C(s_n)$.

Table 1 A list of notations

$s_i, 1 \leq i \leq n$	Sampled vertices
$\pi = (\pi_v : v \in V)$	Vertex sampling probability distribution
$\mathbf{C} = \{\mathbf{c}_1, \dots, \mathbf{c}_H\}$	Set of all content appearing in graph G
$L(\mathbf{c})$	Label of content $\mathbf{c} \in \mathbf{C}$
$\{l_0, \dots, l_K\}$	Range of label function $L(\mathbf{c})$
$\omega = (\omega_0, \dots, \omega_K)$	Distribution of content by the content label
$f(\mathbf{c})$	Number of copies that content \mathbf{c} possesses
$\{c^{(1)}, \dots, c^{(f(\mathbf{c}))}\}$	All copies of content \mathbf{c}
c^*	The special copy of content \mathbf{c}
$C_S = \{c_1^*, \dots, c_H^*\}$	Special content copy set
c	A copy of content \mathbf{c}
$v(c)$	Vertex that maintains content copy c
$L(c), f(c)$	$L(c) = L(\mathbf{c}), f(c) = f(\mathbf{c})$
$C(v), v \in V$	Content copies maintained by vertex v
\mathbf{C}_D	Set of content that has a copy maintained by sampled vertices s_i

To solve this challenge, we develop two estimators SCE and WCE utilizing the following two kinds of content meta-information, respectively:

Meta-information 1: special copy indicator We let c^* denote a **special copy**, such as the original source of content \mathbf{c} . Note that each content \mathbf{c} has one and only one special copy c^* . Unless we state otherwise, in what follows the notation \mathbf{c} is used to depict content and c is used to depict a copy of content \mathbf{c} . Let $C_S = \{c_1^*, \dots, c_H^*\}$ be a **special content copy set**. For some graphs, sampling methods can check whether a sampled content copy is special or not and generate such a set C_S . For example, a tweet in the Sina Microblog can be classified into an original tweet or a retweet; therefore, we can generate C_S consisting of all original tweets.

Meta-information 2: content copy count For content $\mathbf{c} \in \mathbf{C}$, let $f(\mathbf{c})$ denote the number of its copies, and $\{c^{(1)}, \dots, c^{(f(\mathbf{c}))}\}$ denote the set of its copies appearing in graph G . For networks such as Sina Microblog and Renren, $f(\mathbf{c})$ is available in each content copy $c^{(i)}$, $1 \leq i \leq f(\mathbf{c})$.

Unlike WCE using each and every content copy sampled, SCE only uses a fraction of sampled content copies. Therefore, WCE tends to exhibit smaller estimation errors than SCE. Table 2 summarizes all content sampling methods studied in this paper.

3.1 Challenges

Challenge 1: directly using sampled content copies is biased for characterizing content in large graphs Let \mathbf{C}_D be the set of content that has at least one copy maintained by sampled vertices. One may directly estimate ω using all distinct sampled content \mathbf{C}_D as

Table 2 A comparison of four estimators DCE, MLE, SCE, and WCE

	DCE	MLE	SCE	WCE
Unbiased or not	×	✓	✓	✓
Meta-information 1 is required or not	×	×	✓	×
Meta-information 2 is required or not	×	×	✓	×
A small number of samples are needed or not	×	×	✓	✓

$$\hat{\omega}_k^{\text{DCE}} = \frac{1}{|\mathbf{C}_D|} \sum_{\mathbf{c} \in \mathbf{C}_D} \mathbf{1}(L(\mathbf{c}) = l_k), \quad 0 \leq k \leq K.$$

We call this straightforward method as Distinct Content Estimator (DCE). For a content copy c , let $v(c)$ be the vertex that maintains c . As alluded, content $\mathbf{c} \in \mathbf{C}$ may be maintained by multiple vertices $v(c^{(j)})$ ($1 \leq j \leq f(\mathbf{c})$) and each vertex $v(c^{(j)})$ is sampled with probability $\pi_{v(c^{(j)})}$, and therefore the probability that one copy of content \mathbf{c} is collected by randomly sampling a vertex is $\sum_{j=0}^{f(\mathbf{c})} \pi_{v(c^{(j)})}$. Note that this probability depends both on the graph sampling method and the number of copies of \mathbf{c} ; therefore, content in \mathbf{C}_D is *not uniformly* sampled from \mathbf{C} . Even when the UNI sampling method is used, where each vertex u is sampled with the same probability $\pi_u = \frac{1}{|V|}$, the probability that \mathbf{C}_D contains \mathbf{c} is proportional to $f(\mathbf{c})$, i.e., popular content is more likely to be sampled than rare ones. This clearly shows that $\hat{\omega}_k^{\text{DCE}}$ using UNI is still biased.

Challenge 2: it is not easy to correct the sampling bias for characterizing content in large graphs To correct the bias of DCE, we use the maximum likelihood estimation method, which is widely used for inferring statistics of interest from partially observed samples. The details of MLE is given in Appendix. In our experiments, however, we will show that one has to sample a large number of vertices in order to accurately calculate $\hat{\omega}_k^{\text{MLE}}$. It is consistent with results observed in [35].

In summary, it is challenging to characterize content distributed over large networks solely based on regular graph sampling methods. To solve this problem, next, we present two methods utilizing two kinds of meta-information (i.e., an special copy indicator and the number of copies) of content, which are available for each content copy in many OSNs.

3.2 Our unbiased estimator I: special copy estimator (SCE)

SCE estimates ω only using collected special content copies, i.e., content copies in \mathbf{C}_S . For content \mathbf{c} , $v(c^*)$ is the vertex maintained its special copy c^* . Then the probability that c^* is collected by sampling a random vertex is $\pi_{v(c^*)}$. Similar to the Horvitz–Thompson estimator introduced in Sect. 2, we use $\hat{\pi}$ defined in Eq. (1) to estimate ω_k ($0 \leq k \leq K$),

$$\hat{\omega}_k^{\text{SCE}} = \frac{1}{S^{\text{SCE}}} \sum_{i=1}^n \sum_{c \in C(s_i)} \frac{\mathbf{1}(L(c) = l_k) \mathbf{1}(c \in C_S)}{\hat{\pi}_{s_i}}, \quad (2)$$

where $S^{\text{SCE}} = \sum_{i=1}^n \sum_{c \in C(s_i)} \frac{\mathbf{1}(c \in C_S)}{\hat{\pi}_{s_i}}$. The pseudo-code of SCE is shown in Algorithm 1. The average time complexity of SCE is $O\left(\frac{n \sum_{v \in V} \pi_v |C(v)|}{|V|}\right)$, where $(\pi_v)_{v \in V}$ is the sampling

probability distribution of the underlying graph sampling method, which can be any sampling method presented in Sect. 2.

Algorithm 1: The pseudo-code of SCE.

```

input : sampling budget  $n$ .
output:  $\hat{\omega}_0^{\text{SCE}}, \dots, \hat{\omega}_K^{\text{SCE}}$ .

 $i = 1$ ;  $S^{\text{SCE}} \leftarrow 0$ ;
foreach  $k \in \{0, 1, \dots, K\}$  do
  |  $\hat{\omega}_k^{\text{SCE}} \leftarrow 0$ ;
end
/* Crawler can be any sampling method in Sect. 2 */
Crawler.initialize();
for  $i \leq n$  do
  |  $s \leftarrow \text{Crawler.sample\_a\_node}()$ ;
  |  $i \leftarrow i + 1$ ;
  | foreach  $c \in C(s)$  do
  | | if  $c$  is a special copy then
  | | | /*  $L(c).id$  is the label ID of content  $c$ , i.e.,  $L(c) = l_{L(c).id}$ . */
  | | |  $k \leftarrow L(c).id$ ;
  | | |  $\hat{\omega}_k^{\text{SCE}} \leftarrow \hat{\omega}_k^{\text{SCE}} + \frac{1}{\hat{\pi}_s}$ ;
  | | |  $S^{\text{SCE}} \leftarrow S^{\text{SCE}} + \frac{1}{\hat{\pi}_s}$ ;
  | | end
  | end
end
foreach  $k \in \{0, 1, \dots, K\}$  do
  |  $\hat{\omega}_k^{\text{SCE}} \leftarrow \frac{\hat{\omega}_k^{\text{SCE}}}{S^{\text{SCE}}}$ ;
end

```

It is important to point out that $\hat{\omega}_k^{\text{SCE}}$ is an *asymptotically unbiased estimator* of ω_k . For each vertex $v \in V$, Eq. (1) shows that $\pi_v/\hat{\pi}_v$ has the same value, denoted as S_π . We have the following equation for each $k = 0, \dots, K$ and $i = 1, \dots, n$

$$\begin{aligned}
 & \mathbb{E} \left[\sum_{c \in C(s_i)} \frac{\mathbf{1}(L(c) = l_k) \mathbf{1}(c \in C_S)}{\hat{\pi}_{s_i}} \right] \\
 &= \sum_{v \in V} \pi_v \sum_{c \in C(v)} \frac{\mathbf{1}(L(c) = l_k) \mathbf{1}(c \in C_S)}{\hat{\pi}_v} \\
 &= S_\pi \sum_{v \in V} \sum_{c \in C(v)} \mathbf{1}(L(c) = l_k) \mathbf{1}(c \in C_S) \\
 &= S_\pi \sum_{c \in C_S} \mathbf{1}(L(c) = l_k) \\
 &= S_\pi H \omega_k.
 \end{aligned}$$

Applying the law of large numbers, we have

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \sum_{c \in C(s_i)} \frac{\mathbf{1}(L(c) = l_k) \mathbf{1}(c \in C_S)}{\hat{\pi}_{s_i}} \xrightarrow{a.s.} S_\pi H \omega_k,$$

where ‘‘a.s.’’ denotes ‘‘almost sure’’ converge, i.e., the event happens with probability one. Similarly, we have $\lim_{n \rightarrow \infty} \frac{S^{\text{SCE}}}{n} \xrightarrow{\text{a.s.}} S_{\pi} H$. Therefore, $\hat{\omega}_k^{\text{SCE}}$ is an asymptotically unbiased estimator of ω_k .

3.3 Our unbiased estimator II: weighted copy estimator (WCE)

WCE estimates ω using all collected content copies $C(s_i)$ ($1 \leq i \leq n$). This estimator is useful for networks (e.g., Sina Microblog and Renren) in which each copy of any content \mathbf{c} records the value of $f(\mathbf{c})$, the number of copies \mathbf{c} has in the network. Vertex $v(c^{(j)})$ maintains the copy $c^{(j)}$ ($1 \leq j \leq f(\mathbf{c})$) of content \mathbf{c} , and it is sampled with probability $\pi_{v(c^{(j)})}$. Meanwhile, a random vertex maintains a copy of \mathbf{c} with probability proportional to $f(\mathbf{c})$. Therefore, we assign a weight $\frac{1}{\pi_{v(c^{(j)})} f(\mathbf{c})}$ for $c^{(j)}$ to remove the sampling bias. Finally, we estimate ω_k ($0 \leq k \leq K$) as

$$\hat{\omega}_k^{\text{WCE}} = \frac{1}{S^{\text{WCE}}} \sum_{i=1}^n \sum_{c \in C(s_i)} \frac{\mathbf{1}(L(c) = l_k)}{\hat{\pi}_{s_i} f(c)}, \quad (3)$$

where $S^{\text{WCE}} = \sum_{i=1}^n \sum_{c \in C(s_i)} \frac{1}{\hat{\pi}_{s_i} f(c)}$. The pseudo-code of WCE is shown in Algorithm 2. Similar to SCE, the average time complexity of WCE is $O\left(\frac{n \sum_{v \in V} \pi_v |C(v)|}{|V|}\right)$.

Algorithm 2: The pseudo-code of WCE.

```

input : sampling budget  $n$ .
output:  $\hat{\omega}_0^{\text{WCE}}, \dots, \hat{\omega}_K^{\text{WCE}}$ .

 $i = 1$ ;  $S^{\text{WCE}} \leftarrow 0$ ;
foreach  $k \in \{0, 1, \dots, K\}$  do
  |  $\hat{\omega}_k^{\text{WCE}} \leftarrow 0$ ;
end
/* Crawler can be any sampling method in Sect. 2 */
Crawler.initialize();
for  $i \leq n$  do
  |  $s \leftarrow \text{Crawler.sample\_a\_node}()$ ;
  |  $i \leftarrow i + 1$ ;
  | foreach  $c \in C(s)$  do
    | /*  $L(c).id$  is the label ID of content  $c$ , i.e.,  $L(c) = l_{L(c).id}$ . */
    |  $k \leftarrow L(c).id$ ;
    |  $\hat{\omega}_k^{\text{WCE}} \leftarrow \hat{\omega}_k^{\text{WCE}} + \frac{1}{\hat{\pi}_s f(c)}$ ;
    |  $S^{\text{WCE}} \leftarrow S^{\text{WCE}} + \frac{1}{\hat{\pi}_s f(c)}$ ;
  | end
end
foreach  $k \in \{0, 1, \dots, K\}$  do
  |  $\hat{\omega}_k^{\text{WCE}} \leftarrow \frac{\hat{\omega}_k^{\text{WCE}}}{S^{\text{WCE}}}$ ;
end

```

$\hat{\omega}_k^{\text{WCE}}$ is an *asymptotically unbiased estimator* of ω_k . To see that, we have the following equation for each $k = 0, \dots, K$ and $i = 1, \dots, n$

$$\begin{aligned} & \mathbb{E} \left[\sum_{c \in C(s_i)} \frac{\mathbf{1}(L(c) = l_k)}{\hat{\pi}_{s_i} f(c)} \right] \\ &= \sum_{v \in V} \pi_v \sum_{c \in C(v)} \frac{\mathbf{1}(L(c) = l_k)}{\hat{\pi}_v f(c)} \\ &= S_\pi \sum_{v \in V} \sum_{c \in C(v)} \frac{\mathbf{1}(L(c) = l_k)}{f(c)} \\ &= S_\pi \sum_{\mathbf{c} \in \mathbf{C}} \sum_{j=1}^{f(\mathbf{c})} \frac{\mathbf{1}(L(c^{(j)}) = l_k)}{f(c^{(j)})} \\ &= S_\pi \sum_{\mathbf{c} \in \mathbf{C}} \mathbf{1}(L(\mathbf{c}) = l_k) \\ &= S_\pi H \omega_k. \end{aligned}$$

Then, we have

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \sum_{c \in C(s_i)} \frac{\mathbf{1}(L(c) = l_k)}{\hat{\pi}_{s_i} f(c)} \xrightarrow{a.s.} S_\pi H \omega_k.$$

Similarly, we have $\lim_{n \rightarrow \infty} \frac{S^{\text{WCE}}}{n} \xrightarrow{a.s.} S_\pi H$. Therefore, $\hat{\omega}_k^{\text{WCE}}$ is an asymptotically unbiased estimator of ω_k .

Extend WCE to estimate graph structure statistics We also note that, compared with previous sampling methods [14,40], WCE is a more *cost effective method* to estimate graph structure statistics for OSNs (e.g., Sina Microblog and Xiami) which carry such meta-information. As shown in Fig. 2, the webpage of a user in Sina Microblog maintains a summary for each of its neighbors (both followers and following), which includes graph properties such as the number of followers, the number of following, and the number of posts. Hence, one can obtain properties of any vertex v and all its neighbors by simply sampling v . Compared with previous works for measuring structure characteristics, we can obtain more accurate estimates by utilizing the meta-information of sampled vertices. It is important to point that when we use this meta-information, we are biased toward vertices with a large number of neighbors (even when using UNI). Therefore, we need a way to unbiased this error. Denote $\text{outdeg}(v)$ as the number of vertices that vertex v follows, and $\text{indeg}(v)$ as the number of vertices that follow v . To remove the sampling bias for observing high-degree vertices' graph properties, we use WCE to estimate vertex label density $\boldsymbol{\tau} = (\tau_0, \dots, \tau_{K'})$ defined in Sect. 2, where τ_k ($0 \leq k \leq K'$) is the fraction of vertices with vertex label l'_k . The property summary of each vertex v can be viewed as content with $f'(v) = \text{indeg}(v) + \text{outdeg}(v) + 1$ copies maintained by followers, following of v , and v itself. For a collected vertex v , define its associated vertices $C'(v)$ as the collection of its following, followers, and itself. Note that $C'(v)$ might contain duplicated elements because a vertex can be both a following and follower of v . We use WCE to estimate τ_k ($0 \leq k \leq K'$) as follows

$$\hat{\tau}_k^{\text{WCE}} = \frac{1}{S'} \sum_{i=1}^n \sum_{v \in C'(s_i)} \frac{\mathbf{1}(L'(v) = l'_k)}{\hat{\pi}_{s_i} f'(v)}, \quad (4)$$

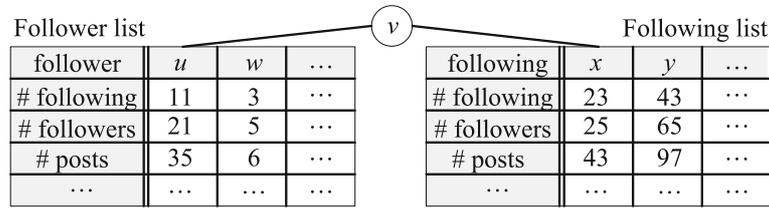


Fig. 2 Graph properties maintained by a vertex v

where $S' = \sum_{i=1}^n \sum_{v \in C'(s_i)} \frac{1}{\hat{\pi}_{s_i} f'(v)}$. Note that $\hat{\tau}_k^{\text{WCE}}$ is an asymptotically unbiased estimator of τ_k . To see that, we have the following equation for each $k = 0, \dots, K'$ and $i = 1, \dots, n$

$$\begin{aligned}
 & \mathbb{E} \left[\sum_{v \in C'(s_i)} \frac{\mathbf{1}(L'(v) = l'_k)}{\hat{\pi}_{s_i} f'(v)} \right] \\
 &= \sum_{u \in V} \pi_u \sum_{v \in C'(u)} \frac{\mathbf{1}(L'(v) = l'_k)}{\hat{\pi}_v f'(v)} \\
 &= S_\pi \sum_{u \in V} \sum_{v \in C'(u)} \frac{\mathbf{1}(L'(v) = l'_k)}{f'(v)} \\
 &= S_\pi \sum_{u \in V} \mathbf{1}(L'(v) = l'_k) \\
 &= S_\pi |V| \tau_k.
 \end{aligned}$$

Then, we have

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \sum_{v \in C'(s_i)} \frac{\mathbf{1}(L'(v) = l'_k)}{\hat{\pi}_{s_i} f'(v)} \xrightarrow{a.s.} S_\pi |V| \tau_k.$$

Similarly, we have $\lim_{n \rightarrow \infty} \frac{S'}{n} \xrightarrow{a.s.} S_\pi |V|$. Therefore, $\hat{\tau}_k^{\text{WCE}}$ is an asymptotically unbiased estimator of τ_k .

4 Data evaluation

Our experiments are performed on a variety of real-world networks, which are summarized in Table 3. Xiami is a popular Web site devoted to music streaming and recommendations. Similar to Twitter, Xiami builds a social network based on follower and following relationships. Each user has a numeric ID that is sequentially assigned. We crawled its entire network graph and have made the dataset publicly available (<http://www.cse.cuhk.edu.hk/%7ecslui/data>). Flickr and YouTube are popular photograph and video sharing Web sites. In these Web sites, a user can subscribe to other user updates such as blogs and photographs. These networks can be represented by a direct graph, with vertices representing users and a directed edge from u to v represents that user u subscribes to user v . Further details of these datasets can be found in [33]. Our experiments are conducted on a Dell Precision T1650 workstation with an Intel Core i7-3770 CPU 3.40 GHz processor and 8 GB DRAM memory.

Table 3 Overview of directed graph datasets used in our simulations

Graph	Xiami	YouTube	Flickr
Vertices (LCC)	1,748,010	1,134,890	1,624,992
Edges (LCC)	16,015,779	2,987,624	15,476,835
Directed edges (LCC)	16,568,449	4,942,035	22,477,015

“Directed edges” refers to the number of directed edges in a directed graph, “edges” refers to the number of edges in an undirected graph, and “LCC” refers to the largest connected component of a given graph

Using real entire graph topologies which are publicly available, we generate benchmark datasets for our simulation experiments by manually generating content and distributing them over these graphs. In the following experiments, we generate 10^7 distinct content and distribute each content \mathbf{c} using four different content distribution schemes (CDSes): CDS I to CDS IV, which model information distribution mechanisms for undirected and directed graphs. CDSes I and II distribute content with a target content distribution by the number of copies. Define the truncated Pareto distribution as $\phi_k = \frac{\alpha}{\gamma k^{\alpha+1}}$, $k = 1, \dots, W$, where $\alpha > 0$, $\gamma = \sum_{k=1}^W \frac{\alpha}{k^{\alpha+1}}$, and W is the maximum number of copies. The number of copies for each content \mathbf{c} is randomly selected from set $\{1, \dots, W\}$ according to the truncated Pareto distribution with parameter α and W for CDSes I and II. Then, copies of \mathbf{c} are distributed as follows

- *CDS I* We distribute each content copy to a randomly selected vertex in G_d (one of the directed graph in Table 3).
- *CDS II* When content has k copies, we first randomly select a vertex v that can reach at least $k - 1$ other vertices. Here, two vertices are reachable if there is at least one path between them in the undirected graph G , which is derived from G_d by ignoring the direction of edges. Then, we assign the special (or original) copy of this content to v , and assign $k - 1$ duplicated copies to the top $k - 1$ nearest vertices in G which are reachable from v .
CDSes III and IV both distribute each content \mathbf{c} using the independent cascade model [15], that is
- *CDS III* We distribute \mathbf{c} over the associated undirected graph G . We first distribute the special copy of \mathbf{c} to a randomly selected vertex v . Then we distribute copies of \mathbf{c} to other vertices iteratively. When a new vertex first receives a copy of \mathbf{c} , it is given a single chance to distribute a copy of \mathbf{c} to each of its neighbors currently without \mathbf{c} with probability p_S .
- *CDS IV* We distribute \mathbf{c} similar to CDS III but on the *direct graph* G_d . The difference is: when a new vertex first receives a copy of \mathbf{c} , it is given a single chance to distribute a copy of \mathbf{c} to each of its incoming neighbors (followers) currently without \mathbf{c} with probability p_S .

We would like to point out that α and p_S are parameters for generating our simulation datasets and they are irrelevant to content sampling methods. In the following experiments, we evaluate the performance of our methods for estimating ω , the content distribution by the number of copies. Let

$$\text{NMSE}(\hat{\omega}_j) = \frac{\sqrt{E[(\hat{\omega}_j - \omega_j)^2]}}{\omega_j}, \quad j = 1, 2, \dots,$$

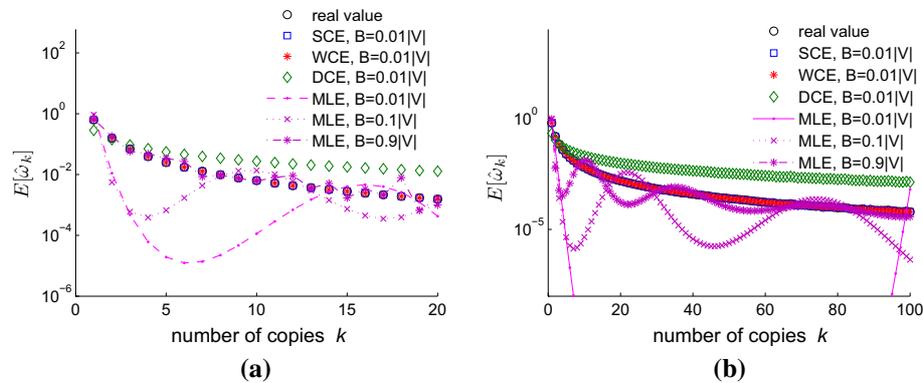


Fig. 3 (Xiami) Average of content distribution estimates for different estimators. **a** $W = 20$. **b** $W = 100$

be a metric that measures the relative error of the estimate $\hat{\omega}_j$ with respect to its true value ω_j . In our experiment, we average the estimates and calculate their NMSE over 1000 runs. Let B denote the sampling budget, which is the number of distinct sampled vertices per run. We set default parameters as: sampling budget $B = 0.01|V|$ and the number of random walkers $T = 1000$ for FS. To simplify notations, graph sampling method A combined with content estimator B is denoted as method A_B.

Figure 3 shows the average of content distribution estimates of 1000 runs for methods DCE, MLE, SCE, and WCE, where the graph sampling method is UNI, and the content distribution scheme is CDS I with $\alpha = 1$ and $W = \{20, 50\}$. We observe that DCE is *highly biased*, while SCE and WCE are unbiased. MLE needs to sample most vertices to reduce biases, especially for large W . Note that SCE and WCE practically coincide with the correct values.

In the following experiments, we set $\alpha = 1$ and $W = 10^5$ for CDSes I and II, and $p_S = 0.01$ for CDSes III and IV. We evaluate the performance of SCE and WCE combined with different graph sampling methods based on the datasets generated by four different CDSes. Figure 4a–d shows the complementary cumulative distribution function (CCDF) of the expectation of content distribution estimates provided by DCE, SCE, and WCE, where the graph is Xiami and the graph sampling method is UNI. We find that DCE exhibits large errors, and SCE and WCE are quite accurate. Similar results are obtained when we use the other four graph sampling methods described in Sect. 2. Figure 4e–t shows the NMSE of SCE and WCE combined with different graph sampling methods for measuring content statistics. The results show that WCE is significantly more accurate than SCE over most points. In particular, WCE is almost an order of magnitude more accurate than SCE for the number of copies larger than 100, and nearly two orders of magnitude more accurate than SCE for the number of copies larger than 1000. Figure 5 shows the compared results for different graph sampling methods where the graph used is Xiami. The results show that UNI is quite accurate and MHRW exhibits large errors for content with a small number of copies. The compared results for WCE show that MHRW is much worse than the other graph sampling methods, while RW and FS have almost the same accuracy. The results for graph YouTube are similar. We show them in Appendix. The time cost of sampling B vertices consists of two parts: (1) computational time and (2) the query response time. In practice, the query response time is usually the bottleneck, because most network service providers impose a query rate limit on crawlers. We can easily find that four graph sampling methods UNI, RW, MHRW, and

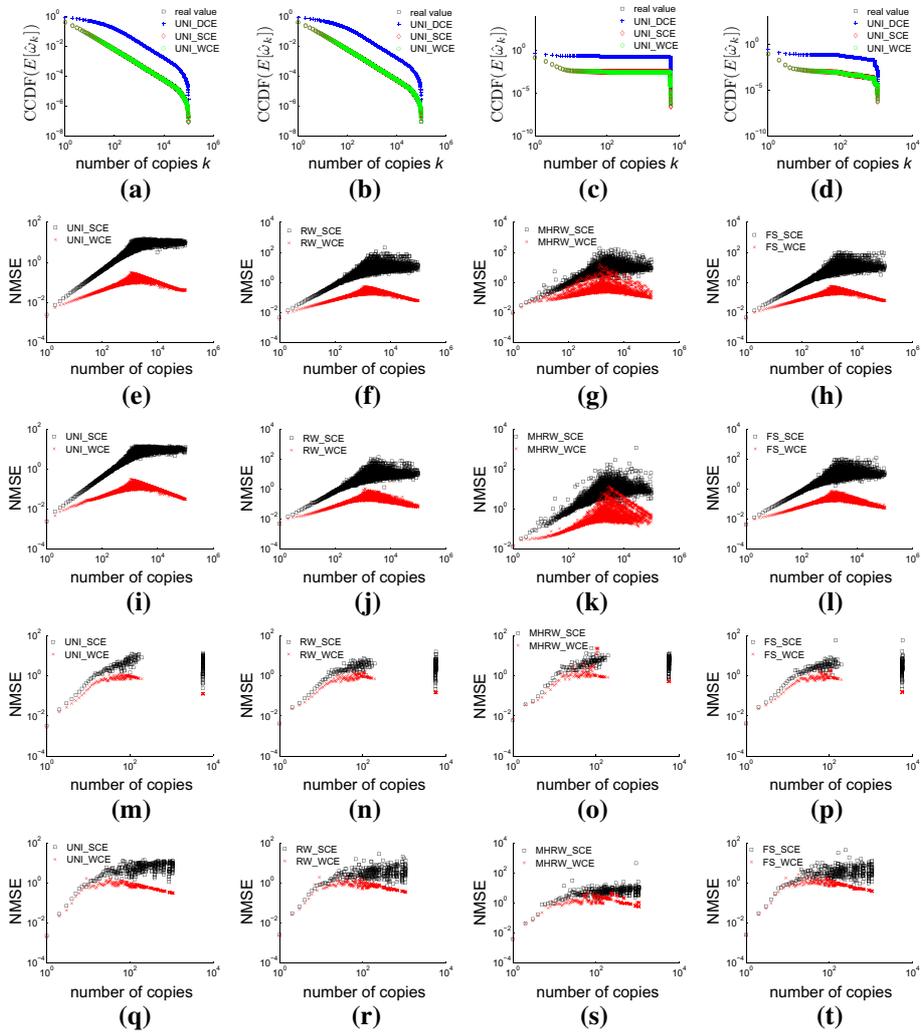


Fig. 4 (Xiami) NMSEs of content distribution estimates for different estimators and graph sampling methods. **a** CDS I. **b** CDS II. **c** CDS III. **d** CDS IV. **e** UNI, CDS I. **f** RW, CDS I. **g** MHRW, CDS I. **h** FS, CDS I. **i** UNI, CDS II. **j** RW, CDS II. **k** MHRW, CDS II. **l** FS, CDS II. **m** UNI, CDS III. **n** RW, CDS III. **o** MHRW, CDS III. **p** FS, CDS III. **q** UNI, CDS IV. **r** RW, CDS IV. **s** MHRW, CDS IV. **t** FS, CDS IV

FS have the similar computational complexity. Moreover, we observe that four estimators DCE, MLE, SCE, and WCE almost exhibit the same computational time. Table 4 shows the computational time for different sampling budget B , where the query response time is not considered. We can see that the computational time increases linearly with B .

Figure 6 shows the distributions of users in Xiami using different labels, where the province numbers and corresponding names are shown in Table 5. The fraction of users with more than 10^4 followers, following, or recommendations is smaller than 2×10^{-6} . The top three popular provinces are Guangdong, Beijing, and Shanghai. Figures 7, 8, 9 and 10 show the results of our methods for estimating vertex label densities. The results show that WCE

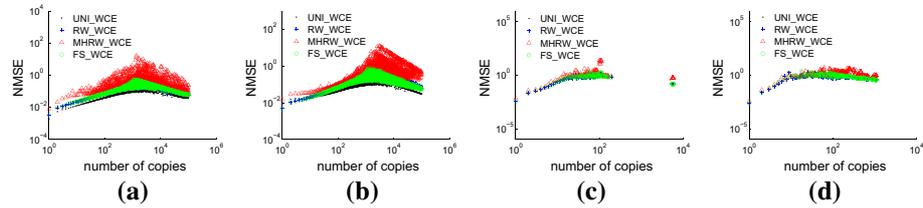


Fig. 5 (Xiami) Compared NMSEs of content distribution estimates for WCE using different graph sampling methods. **a** CDS I. **b** CDS II. **c** CDS III. **d** CDS IV

Table 4 Computational time (seconds) for different sampling budgets and graphs

$B =$	Xiami	YouTube	Flickr
$0.01 V $	0.035	0.024	0.039
$0.1 V $	0.349	0.238	0.389
$0.9 V $	3.146	2.145	3.51

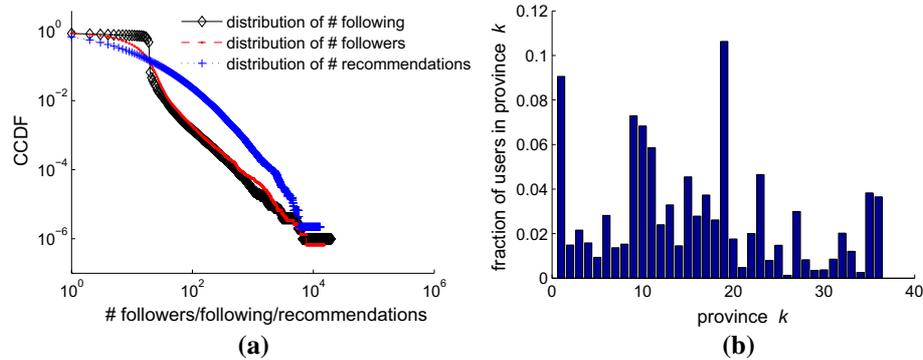


Fig. 6 (Xiami) The distributions of users by different labels. **a** # followers, # following, and # recommendations. **b** Location

significantly outperforms previous methods over almost all points. This is because WCE uses neighbors' graph property summaries of sampled vertices. In particular for UNI_WCE, which is an order of magnitude more accurate than UNI for follower/following counts larger than 100. Figure 11 shows the compared results for different graph sampling methods where the graph used is Xiami. The results show that MHRW is quite accurate. RW and FS almost have the same accuracy. For the follower and recommendation count distributions, UNI is more accurate for follower and recommendation counts with small values. Figures 12 and 13 show the results for estimating out-degree distribution for YouTube and Flickr, respectively. We observe that WCE is better than previous methods over almost all points.

5 Applications

We now apply our methodology to a real OSN, Sina Microblog network, to characterize various content, e.g., the average number of retweets or replies per tweet, types of tweet messages, as well as the associated top rank statistics. By crawling webpages of 148,313 random accounts selected by UNI, we obtain 19.7 million tweets and retweets. Note that in

Table 5 (Xiami) Province numbers and corresponding names

1. Beijing	2. Tianjin	3. Hebei
4. Shanxi	5. Inner Mongolia	6. Liaoning
7. Jilin	8. Heilongjiang	9. Shanghai
10. Jiangsu	11. Zhejiang	12. Anhui
13. Fujian	14. Jiangxi	15. Shandong
16. Henan	17. Hubei	18. Hunan
19. Guangdong	20. Guangxi	21. Hainan
22. Chongqing	23. Sichuan	24. Guizhou
25. Yunnan	26. Tibet	27. Shannxi
28. Gansu	29. Qinghai	30. Ningxia
31. Xinjiang	32. Taiwan	33. Hong Kong
34. Macao	35. Null	36. Overseas

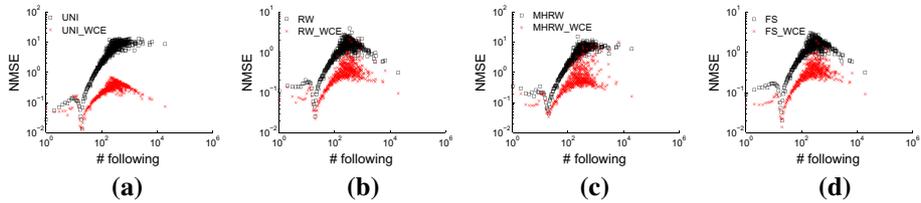


Fig. 7 NMSEs of following count distribution estimates for different graph sampling methods and estimators. **a** UNI. **b** RW. **c** MHRW. **d** FS

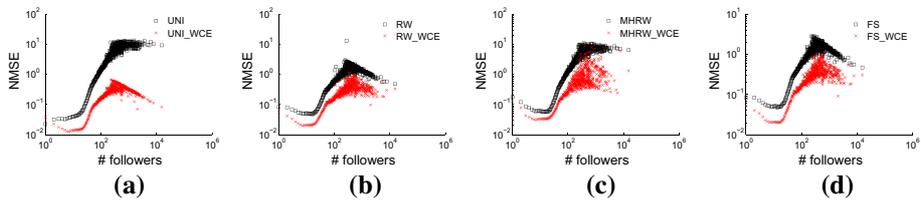


Fig. 8 (Xiami) NMSEs of follower count distribution estimates for different graph sampling methods and estimators. **a** UNI. **b** RW. **c** MHRW. **d** FS

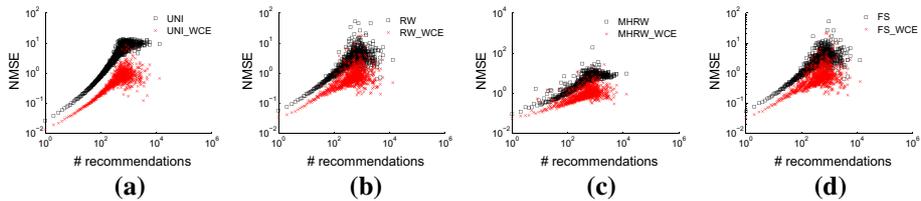


Fig. 9 (Xiami) NMSEs of recommendation count distribution estimates for different graph sampling methods and estimators. **a** UNI. **b** RW. **c** MHRW. **d** FS

the following analysis, tweets refer to the original tweets. Each tweet or retweet records its original tweet's information such as the number of retweets and replies. Figure 14 shows the results of estimating the distribution of tweets from retweets and replies using DCE, SCE, and WCE, where the special content is defined as the original tweet for SCE. The estimates

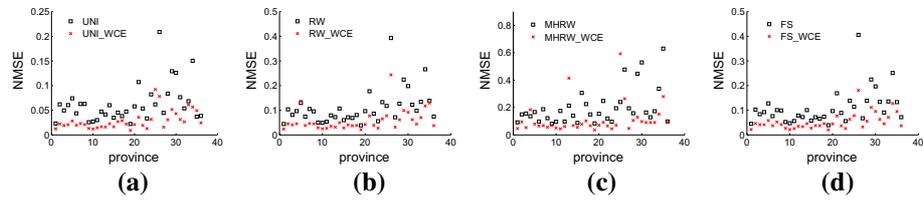


Fig. 10 (Xiami) NMSEs of location distribution estimates for different graph sampling methods and estimators. **a** UNI. **b** RW. **c** MHRW. **d** FS

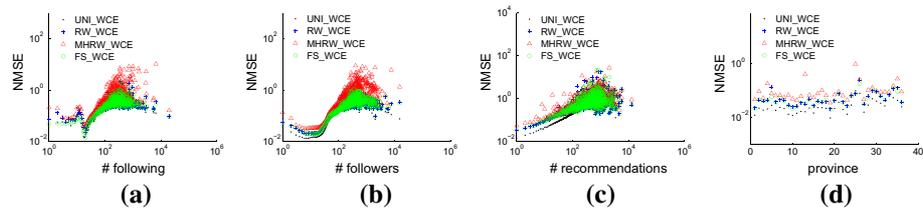


Fig. 11 (Xiami) Compared NMSEs of graph label density estimates for WCE using different graph sampling methods. **a** # following. **b** # follower. **c** # recommendation. **d** Location

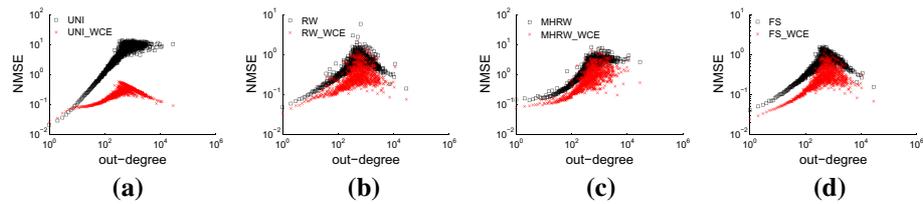


Fig. 12 (YouTube) NMSEs of out-degree distribution estimates for different graph sampling methods and estimators. **a** UNI. **b** RW. **c** MHRW. **d** FS_WCE versus FS

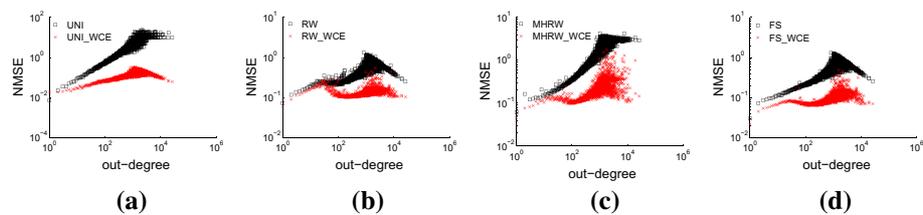


Fig. 13 (Flickr) NMSEs of out-degree distribution estimates for different graph sampling methods and estimators. **a** UNI. **b** RW. **c** MHRW. **d** FS

for the average number of retweets and replies per tweet are shown in Table 6. We observe that the estimates of SCE and WCE are close to each other, but the estimates obtained by DCE significantly deviate from SCE and WCE. This is consistent with previous experimental results which show that DCE exhibits a large bias. Furthermore, as shown in Fig. 14, we can see that the maximum number of retweets or replies given by SCE is the smallest because it only uses information of sampled original tweets, and the original tweets of popular tweets are not always sampled.

Let us explore the “type” of tweets. We classify tweets into three types: *text tweet*, *image tweet*, and *video tweet*. Table 7 shows their statistics measured by WCE. We find that 60.1%

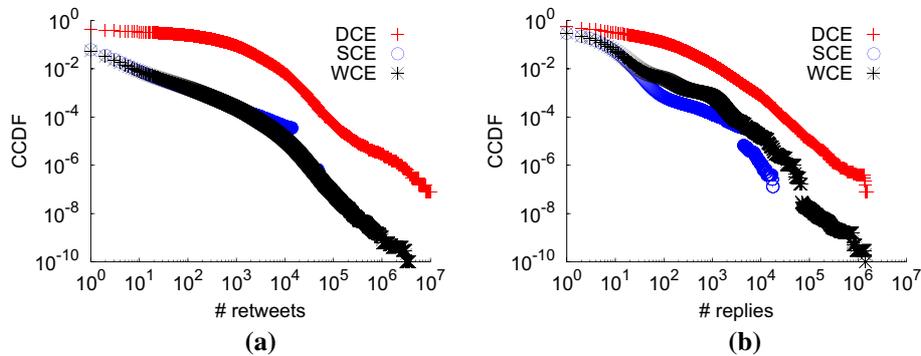


Fig. 14 Distributions of tweets by the number of retweets and replies. **a** # retweets. **b** # replies

Table 6 Our estimates of the average number of retweets and replies per tweet

	Avg. # retweets	Avg. # replies
DCE	423	89.8
SCE	2.01	3.93
WCE	1.60	4.60

Table 7 (Sina Microblog) Statistics of tweets by different categories

	Fraction of tweets (%)	Avg. # retweets per tweet	Avg. # replies per tweet
Text	60.1	0.31	2.30
Image	37.6	3.33	7.91
Video	2.3	7.05	10.91

Table 8 (Sina Microblog) Statistics of video tweets

Source	Fraction of video tweets (%)	Avg. # retweets per tweet	Avg. # replies per tweet
youku.com	42.7	5.57	9.68
tudou.com	26.3	9.87	13.73
sina.com	10.0	9.09	11.32
yinyuetai.com	6.2	5.06	9.36
56.com	4.4	11.29	26.64
ku6.com	2.9	11.29	13.02
sohu.com	1.7	4.56	1.29
kandian.com	1.6	0.13	0.16
baomihua.com	1.6	0.31	0.08
ifeng.com	0.9	5.77	4.36

are text tweets, 37.6% are image tweets, and 2.3% are video tweets. On average, image and video tweets have more retweets and replies than text tweets. Table 8 shows the statistics of video tweets by their associated external video source Web sites. We find that the top five popular video Web sites are youku.com (42.7%), tudou.com (26.3%), sina.com (10.0%), yinyuetai.com (6.2%), and 56.com (4.4%).

6 Related work

Little attention has been given to develop a formal crawling methodology to estimate statistics of content over large networks. Zafar et al. [53] conduct an extensive measurement study on comparing the performance of using expert tweets and random sampled tweets on Twitter for applications such as topical search and breaking news detection. Unlike Twitter, in this paper we assume the network of interest does not support an API to sample content at random. Our problem is closely related to research on crawling and sampling large networks. In this paper, we differ crawling and sampling methods as: (1) crawling methods assume that the network of interest is not given in advance, and they aim to accurately estimate network statistics from as few queried nodes as possible because service providers usually impose a rate limit on crawlers; (2) unlike crawling methods, sampling methods assume that the entire network is given in advance, and they aim to design sampling methods to reduce the computational cost of calculating network statistics such as the number of triangles in the network.

Crawling methods for reducing query cost Previous crawling work focuses on designing accurate and efficient methods for measuring graph characteristics, such as vertex degree distribution [14, 38, 40, 41, 45] and the topology of vertices' groups [24]. Most previous OSN graph crawling and sampling work focuses on undirected graph since each vertex in most OSNs maintains both its incoming and outgoing neighbors, so it is easy to convert these directed OSNs to their associated undirected graphs by ignoring the directions of edges. Breadth-First-Search (BFS) is easy to implement but it introduces a large bias toward high-degree vertices, and it is difficult to remove these biases in general [1, 25, 26]. Random walk (RW) is biased to sample high-degree vertices, however its bias is known and can be corrected [17, 42]. Compared with uniform vertex sampling (UNI), RW has smaller estimation errors for high-degree vertices, and these vertices are quite common for many OSNs like Facebook, Myspace, and Flickr [40]. Furthermore, it is costly to apply UNI in these networks. Metropolis–Hastings RW (MHRW) [14, 45, 54] modifies the RW procedure, and it aims to sample each vertex with the same probability. The accuracy of RW and MHRW is compared in [14, 38]. RW is shown to be consistently more accurate than MHRW. The mixing time of RW determines the efficiency of the sampling, and it is found to be much larger than commonly believed for many OSNs [34]. There are a lot of work on how to decrease the mixing time [3, 7, 11, 13, 23, 40, 55]. Avrachenkov et al. [3] observes that random jumps increase the spectral gap of the random walk, which leads to faster convergence to the steady state distribution. Kurant et al. [23] assigns weights to nodes that are computed using their neighborhood information, and develops a weighted RW-based method to perform stratified sampling on social networks. Dasgupta et al. [11] randomly samples nodes (either uniformly or with a known bias) and then uses neighborhood information to improve its unbiased estimator. Zhou et al. [55] modifies the regular random walk by “rewiring” the network of interest on-the-fly in order to reduce the mixing time of the walk. To sample a directed graph with latent incoming links (e.g., the Web graph and Flickr), Ribeiro et al. [41] use a RW with jumps under the assumption that vertices can be uniformly sampled at random from directed graphs. In addition to node/edge label distribution estimation, Chen et al. and Wang et al. [9, 48] present crawling methods for characterizing graphlets (i.e., small connected subgraph patterns) in large networks.

Sampling methods for reducing computational time Recently, counting triangle in a large graph attracts a lot of attention. Seshadhri et al. and Wu et al. [43, 51] present sampling algorithms for estimating triangles in a large static graph, which is given in advance and

its size can be fitted into memory space. [2,4,8,19,20,37,44,47] develop several streaming algorithms for estimating the global count of triangles in a large graph represented as a stream of edges. Jha et al. [19] develop a wedge sampling-based algorithm to estimate the number of triangles in a graph stream. Tsourakakis et al. [47] present a triangle sparsification method by sampling each and every edge with a fixed probability, which can also be used to estimate triangle counts. Ahmed et al. [2] present a more general edge sampling framework for estimating a variety of graph statistics including the number of triangles. Stefani et al. [44] use a fixed user-specified memory space to sample edges by the reservoir sampling technique. Moreover, [5,27,29,44] develop algorithms for estimating local (i.e., incident to each vertex) counts of triangles in a large graph stream. In addition to triangle counting, a number of algorithms [6,21,36,49,50] have been developed for estimating high-order graphlet concentrations/counts in a large graph, which is more computationally intensive.

7 Conclusions

In this paper, we study the problem of estimating characteristics of content distributed over large graphs. Our analysis and experimental results show that existing graph sampling methods are biased to sample content with a large number of copies, and there can be a huge bias in statistics computed by using collected content in a direct manner. To remove this bias, the MLE method is applied. However, we observe that MLE needs to sample most vertices in the graph to obtain an accurate estimate. To address this challenge, we develop two accurate methods SCE and WCE using available meta-information in sampled content and prove that they are asymptotically unbiased. We perform extensive experiments and demonstrate that WCE is more accurate than SCE. Furthermore, we use WCE to estimate graph characteristics when vertices maintain their neighbors' graph properties and conduct experiments to show that WCE is more accurate than regular sampling methods. Our methods SCE and WCE fail to estimate characteristics of content distributed over large graphs when the meta-data of a content do not provide a special copy indicator or the number of copies that the content holds. In future, we plan to solve this limitation.

Acknowledgements The authors wish to thank the anonymous reviewers for their helpful feedback. This work was supported in part by Army Research Office Contract W911NF-12-1-0385, and ARL under Cooperative Agreement W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the ARL, or the U.S. Government. The work was also supported in part by National Natural Science Foundation of China (61603290, 61602371, U1301254), Ministry of Education and China Mobile Research Fund (MCM20160311), China Postdoctoral Science Foundation (2015M582663), Natural Science Basic Research Plan in Zhejiang Province of China (LGG18F020016), Natural Science Basic Research Plan in Shaanxi Province of China (2016JQ6034, 2017JM6095), Shenzhen Basic Research Grant (JCYJ20160229195940462).

Appendix

MLE method

We present the MLE of ω only for graph sampling method UNI, because it is not easy to derive the MLE of ω for other graph sampling methods such as RW, MHRW, and FS. Suppose that the graph size is known (this can be estimated by sampling methods proposed in [22]), $n < |V|$ vertices are sampled and then each copy of \mathbf{c} is sampled with the same probability

$p = \frac{n}{|V|}$. For simplicity, we assume that content is distributed over networks uniformly at random. Let M be the maximum number of copies that content has. Denote $P_{i,j}$ as the probability of sampling i copies for content totally having j copies, where $1 \leq i \leq j \leq M$. Let $q = 1 - p$, we have $P_{i,j} = \frac{\binom{j}{i} p^i q^{j-i}}{1-q^j}$. We compute the MLE of ω from sampled content copies in respect to the following two cases:

Case 1 When the content label under study is the number of copies associated with content. For randomly sampled content, let α_i ($1 \leq i \leq M$) be the probability that it has i copies sampled. Among sampled content, let x_i be the fraction of content with i copies sampled. We have $\mathbb{E}(x_i) = \alpha_i$. Thus, x_i is an unbiased estimate of α_i . Next, we present a method to estimate ω based on the relationship of α_i and ω . The likelihood function of α_i is

$$\alpha_i = \sum_{j=i}^M \omega_j P_{i,j}. \quad (5)$$

This is similar to packet sampling-based flow size distribution estimation studied in [12], where each packet is sampled with probability p . Here a flow refers to a group of packets with the same source and destination, and the flow size is the number of packets that it contains. In our context, content corresponds to a flow, and its copies correspond to packets in the flow. Therefore, we can develop a maximum likelihood estimate $\hat{\omega}_k^{\text{MLE}}$ of ω_k ($1 \leq k \leq M$) similar to the method proposed in [12].

Case 2 When the content label under study is independent with the number of duplicates and it is available in each content copy, which is not a latent property such as the number of copies content has, we use the following approach to derive the MLE. Define $\beta_{k,j}$ ($0 \leq k \leq K$, $1 \leq j \leq M$) as the fraction of the number of content with label l_k and j copies over the number of content with label l_k . For sampled content, let $\alpha_{k,i}$ ($1 \leq i \leq M$) be the probability that its content label is l_k and has i copies sampled. Then, the likelihood function of $\alpha_{k,i}$ is

$$\alpha_{k,i} = \sum_{j=i}^M \beta_{k,j} P_{i,j}.$$

$\alpha_{k,i}$ can be estimated based on sampled content copies. That is, among sampled content, let $x_{k,i}$ be the fraction of content with label l_k that has i copies sampled. We have $\mathbb{E}(x_{k,i}) = \alpha_{k,i}$. Therefore, $x_{k,i}$ is an unbiased estimate of $\alpha_{k,i}$. Similar to (5), we then develop a maximum likelihood estimate $\hat{\beta}_{k,j}$ of $\beta_{k,j}$, $1 \leq j \leq M$. Since

$$\alpha_k = \omega_k \sum_{i=1}^M \sum_{j=i}^M \beta_{k,j} P_{i,j},$$

we have the following estimator of ω_k

$$\hat{\omega}_k^{\text{MLE}} = \frac{\hat{\alpha}_k}{S^{\text{MLE}} \sum_{i=1}^M \sum_{j=i}^M \hat{\beta}_{k,j} P_{i,j}}, \quad 0 \leq k \leq K,$$

where $\hat{\alpha}_k$ is the fraction of sampled content with label l_k , and

$$S^{\text{MLE}} = \sum_{k=0}^K \frac{\hat{\alpha}_k}{\sum_{i=1}^M \sum_{j=i}^M \hat{\beta}_{k,j} P_{i,j}}.$$

References

1. Achlioptas D et al (2005) On the bias of traceroute sampling or, power-law degree distributions in regular graphs. In: STOC, pp 694–703
2. Ahmed N et al (2014) Graph sample and hold: a framework for big-graph analytics. In: SIGKDD, pp 1446–1455
3. Avrachenkov K et al (2010) Improving random walk estimation accuracy with uniform restarts. In: WAW, pp 98–109
4. Bar-Yossef Z et al (2002) Reductions in streaming algorithms, with an application to counting triangles in graphs. In: SODA, pp 623–632
5. Becchetti L et al (2010) Efficient algorithms for large-scale local triangle counting. TKDD 4(3):13:1–13:28
6. Bhuiyan MA et al (2012) Guise: uniform sampling of graphlets for large graph analysis. In: ICDM, pp. 91–100
7. Boyd S et al (2004) Fastest mixing Markov chain on a graph. SIAM Rev 46(4):667–689
8. Buriol LS et al (2006) Counting triangles in data streams. In: PODS, pp 253–262
9. Chen X et al (2017) A general framework for estimating graphlet statistics via random walk. In: PVLDB, pp 253–264
10. Chib S, Greenberg E (1995) Understanding the metropolis-hastings algorithm. Am. Stat. 49(4):327–335
11. Dasgupta A et al (2012) Social sampling. In: SIGKDD, pp 235–243
12. Duffield N et al (2003) Estimating flow distributions from sampled flow statistics. In: SIGCOMM, pp 325–336
13. Gjoka M et al (2011) Multigraph sampling of online social networks. JSAC 29(9):1893–1905
14. Gjoka M et al (2010) Walking in facebook: a case study of unbiased sampling of OSNs. In: INFOCOM, pp 2498–2506
15. Goldenberg J et al (2001) Talk of the network: a complex systems look at the underlying process of word-of-mouth. Mark. Lett. 12(3):211–223
16. Hastings WK (1970) Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57(1):97–109
17. Heckathorn DD (2002) Respondent-driven sampling II: deriving valid population estimates from chain-referral samples of hidden populations. Soc Probl 49(1):11–34
18. Horvitz D, Thompson D (1952) A generalization of sampling without replacement from a finite universe. JASA 47(260):663–685
19. Jha M et al (2013) A space efficient streaming algorithm for triangle counting using the birthday paradox. In: SIGKDD, pp 589–597
20. Jowhari H, Ghodsi M (2005) New streaming algorithms for counting triangles in graphs. In: COCOON, pp 710–716
21. Kashtan N et al (2004) Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. Bioinformatics 20(11):1746–1758
22. Katzir L et al (2011) Estimating sizes of social networks via biased sampling. In: WWW, pp 597–606
23. Kurant M et al (2011) Walking on a graph with a magnifying glass: stratified sampling via weighted random walks. In: SIGMETRICS, pp 281–292
24. Kurant M et al (2012) Coarse-grained topology estimation via graph sampling. In: WOSN, pp. 25–30
25. Kurant M et al (2010) On the bias of bfs (breadth first search) and of other graph sampling techniques. In: ITC, pp 1–9
26. Kurant M et al (2011) Towards unbiased bfs sampling. JSAC 29(9):1799–1809
27. Kutzkov K, Pagh R (2013) On the streaming complexity of computing local clustering coefficients. In: WSDM, pp 677–686
28. Li Z et al (2012) Socialtube: P2P-assisted video sharing in online social networks. In: INFOCOM mini conference, pp 2886–2890
29. Lim Y, Kang U (2015) MASCOT: memory-efficient and accurate sampling for counting local triangles in graph streams. In: SIGKDD, pp 685–694
30. Lovász L (1993) Random walks on graphs: a survey. Combinatorics 2:1–46
31. Malandrino F et al (2012) Proactive seeding for information cascades in cellular networks. In: INFOCOM, pp 2886–2890
32. Metropolis N et al (1953) Equations of state calculations by fast computing machines. JSAC 21(6):1087–1092
33. Mislove A et al (2007) Measurement and analysis of online social networks. In: IMC, pp 29–42
34. Mohaisen A et al (2010) Measuring the mixing time of social graphs. In: IMC, pp 390–403

35. Murai F et al (2012) On set size distribution estimation and the characterization of large networks via sampling. *JSAC* 31(6):1017–1025
36. Omidi S et al (2009) Moda: an efficient algorithm for network Motif discovery in biological networks. *GGs* 84(5):385–395
37. Pavany A et al (2013) Counting and sampling triangles from a graph stream. In: *PVLDB*, pp 1870–1881
38. Rasti AH et al (2009) Respondent-driven sampling for characterizing unstructured overlays. In: *INFOCOM mini-conference*, pp 2701–2705
39. Ribeiro B et al (2010) On MySpace account spans and double Pareto-like distribution of friends. In: *NetSciCom*, pp 1–6
40. Ribeiro B, Towsley D (2010) Estimating and sampling graphs with multidimensional random walks. In: *IMC*, pp 390–403
41. Ribeiro B et al (2012) Sampling directed graphs with random walks. In: *INFOCOM*, pp 1692–1700
42. Salganik MJ, Heckathorn DD (2004) Sampling and estimation in hidden populations using respondent-driven sampling. *Sociol Methodol* 34:193–239
43. Seshadhri C et al (2014) Wedge sampling for computing clustering coefficients and triangle counts on large graphs. *Stat Anal Data Min* 7(4):294–307
44. Stefani LD et al (2016) Trièst: counting local and global triangles in fully-dynamic streams with fixed memory size. In: *SIGKDD*, pp 825–834
45. Stutzbach D et al (2009) On unbiased sampling for unstructured peer-to-peer networks. *TON* 17(2):377–390
46. Suh B et al (2010) Want to be retweeted? large scale analytics on factors impacting retweet in twitter network. In: *SocialCom*, pp 177–184
47. Tsourakakis CE et al (2009) Doulion: counting triangles in massive graphs with a coin. In: *KDD*, pp 837–846
48. Wang P et al (2014) Efficiently estimating motif statistics of large networks. *TKDD* 9(2):8:1–8:27
49. Wang P et al (2016) Minfer: a method of inferring motif statistics from sampled edges. In: *ICDE*, pp 1050–1061
50. Wernicke S (2006) Efficient detection of network motifs. *TCBB* 3(4):347–359
51. Wu B et al (2016) Counting triangles in large graphs by random sampling. *TKDE* 28(8):2013–2026
52. Yang M et al (2004) Deployment of a large-scale peer-to-peer social network. In: *WORLDS*, pp 1–6
53. Zafar MB et al (2015) Sampling content from online social networks: comparing random versus expert sampling of the twitter stream. *TWEB* 9(3):12:1–12:33
54. Zhong M, Shen K (2006) Random walk based node sampling in self-organizing networks. *SIGOPS Oper Syst Rev* 40(3):49–55
55. Zhou Z et al (2013) Faster random walks by rewiring online social networks on-the-fly. *TODS* 40(4):26:1–26:36



Pinghui Wang received the B.S. degree in information engineering and the Ph.D. degree in automatic control from Xi'an Jiaotong University, Xi'an, China, in 2006 and 2012, respectively. He is currently an associate professor in MOE Key Laboratory for Intelligent Networks and Network Security at Xi'an Jiaotong University. His research interests include Internet traffic measurement and modeling, abnormal detection, and online social network measurement.



Junzhou Zhao received the B.S. degree in information engineering and the Ph.D. degree in automatic control from Xi'an Jiaotong University, Xi'an, China, in 2008 and 2015, respectively. He is currently a Post-doc Fellow in the Division of Computer, Electrical and Mathematical Sciences and Engineering, King Abdullah University of Science and Technology, Saudi Arabia. His research interests include online social network measurement and modeling.



John C. S. Lui received the Ph.D. degree in computer science from UCLA. He is currently a professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong. His current research interests include communication networks, network system security, network economics, network sciences, cloud computing, large-scale distributed systems and performance evaluation theory.



Don Towsley holds a B.A. in Physics (1971) and a Ph.D. in Computer Science (1975) from University of Texas. From 1976 to 1985, he was a member of the faculty of the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst. He is currently a Distinguished Professor at the University of Massachusetts in the Department of Computer Science. His research interests include network measurement and modeling.



Xiaohong Guan received the B.S. and M.S. degrees in automatic control from Tsinghua University, Beijing, China, in 1982 and 1985, respectively, and the Ph.D. degree in electrical engineering from the University of Connecticut, Storrs, US, in 1993. He is currently a professor at the Systems Engineering Institute, Xi'an Jiaotong University, Xi'an, China. His research interests include allocation and scheduling of complex networked resources, network security, and sensor networks.