Estimating the size of online social networks

Shaozhi Ye
Department of Computer Science
University of California, Davis
sye@ucdavis.edu

Felix Wu
Department of Computer Science
University of California, Davis
wu@cs.ucdavis.edu

***Abstract*—The huge size of online social networks (OSNs) makes it prohibitively expensive to precisely measure any properties which require the knowledge of the entire graph. To estimate the size of an OSN, i.e., the number of users an OSN has, this paper introduces two estimators using widely available OSN functionalities/services. The first estimator is a maximum likelihood estimator (MLE) based on uniform sampling. An $O(\log n)$ algorithm is developed to solve the estimator, which is 70 times faster than the naive linear probing algorithm in our experiments. The second estimator is based on random walkers and we generalize it to estimate other graph properties. In-depth evaluations are conducted on six real OSNs to show the bias and variance of these two estimators. Our analysis addresses the challenges and pitfalls when developing and implementing such estimators for OSNs.**

I. Introduction

As extensive studies are being conducted on OSNs, various OSN properties become the quantity of interest, such as size, diameter and clustering coefficient. Many properties, however, are challenging to measure because they require the knowledge of the entire graph and it is expensive to crawl OSNs with millions of users. Moreover, OSNs are not crawler friendly. Most of them enforce rate limiting and block clients which send lots of requests within a short period of time. It is also nontrivial to parse the dynamic pages generated by OSNs efficiently. Thus sampling and estimation have to be employed in many cases. In this paper, we examine the problem of estimating the size of an OSN, i.e., the number of users an OSN has, which is fundamental to various analysis. For instance, to see how representative a sample is, one important metric is the sampling rate, the number of users in the sample versus the entire population the OSN has.

Currently there are limited ways to get the size of an OSN without crawling the entire network.

- The OSN provider may disclose this number but outsiders are unable to verify it. Meanwhile more and more OSNs give the number of *active users* but its definition varies from one to another, sometimes even unavailable to the public.
- For outsiders, one commonly used approach is estimating the size with the largest valid user ID, which fails if the OSN assigns user IDs non-sequentially (such as Facebook).
- We may estimate the used portion of the non-sequential ID space by probing the ID space. This approach becomes expensive when IDs are assigned non-uniformly.
- To make things worse, numeric user IDs are unavailable on some OSNs (such as YouTube[1]), which makes the ID space both non-uniform (some strings are more likely to be chosen by users) and huge (larger alphabets).

This paper introduces two estimators which do not rely on how OSNs assign user IDs.

- *MLE*: Maximum likelihood estimation based on uniform sampling.
- *RW*: Unbiased estimator based on random walkers.

Both estimators require that any two OSN users are distinguishable, which can be accomplished by comparing user IDs, names, profiles, etc. Besides, they also have their own assumptions.

- MLE requires the capability to uniformly sample a user from the OSN. Many OSNs provide a service to suggest a random user, which may serve as a pseudo uniform sample generator.
- RW requires the friend lists of users to be available to the random walker. Although OSNs allow users to hide their friend lists from strangers, most users keep these lists public [1].

Hence these assumptions are likely to be satisfied in real world.

Instead of using synthetic data generated by social network models, this paper evaluates these two estimators with real OSNs. Our contributions are summarized as follows.

- Two estimators are introduced to estimate the size of OSNs by using widely available OSN services. (Section III and IV)
- An $O(\log n)$ algorithm is developed to solve the MLE problem quickly (70 times faster in our experiments). (Section III-B)
- In-depth evaluations are conducted over real OSNs to address their bias and variance. (Section III-C, III-D and IV-B)
- The RW estimator is generalized to estimate graph properties other than the size of the graph. A biased estimator for clustering coefficient is proposed and evaluated. (Section V)
- We also discuss the challenges and pitfalls when developing these estimators. (Section VI)

We believe that this paper presents the first systematic analysis and case study on estimating the size of OSNs and provides insight into developing estimators on large OSNs.

[1] http://www.youtube.com

II. Preliminaries

This section briefly introduces the terminologies and notations used in this paper.

By treating a user as a node (vertex) and the *friend* relationship between two users as an link (edge), an OSN can be modeled as a graph $G(V, E)$, where V denotes the set of nodes and E denotes the set of links. Let $n = |V|$ denote the size of G, i.e., the total number of nodes on G, which is also commonly known as the *population* of G.

Different OSNs have different definitions of *friends*. Symmetrical (or mutual) friendship corresponds to undirected edges (or bi-directed edges). Asymmetrical friendship leads to directed edges. e_{uv} may represent an undirected edge between u and v or a directed edge from u to v. There is no ambiguity when the specific G is given.

Given a node u, its *neighbors* are the set of nodes $\{v|\forall v, \exists e_{uv} \in E\}$. When a node is *crawled* (or *visited*), its neighbors are known, but the neighbors of its neighbors are not. In reality, crawling an OSN users may include crawling his/her profiles, messages, etc., but in this paper we only need the friend lists.

The main performance metrics we use to evaluate the estimators are *bias* and *variance*. The bias quantifies the difference between the estimated size ($\hat{n}$) and the population (n) of the graph. Larger n usually leads to larger bias. To evaluate the bias without being skewed by n, we define the estimation error as follows.

$$\frac{|n - \hat{n}|}{n} \tag{1}$$

The variance of $\hat{n}$ characterizes how consistent the estimated results are. This paper uses its squared root, standard deviation.

III. Estimating with uniform sampling

Assuming we are able to sample uniformly over an OSN (with replacement), as more and more nodes being sampled, the probability of sampling the same node multiple times increases. Based on this information, we can develop an MLE for the size of the OSN. In this section, we first formulate this MLE problem in Section III-A and then develop a fast solution in Section III-B. Section III-C gives the simulation results for its bias and variance and Section III-D presents our experiment on Twitter.com.

A. Maximum likelihood estimator

Formally, given the number of nodes we have sampled and the unique nodes we have seen, denoted as s and k respectively, the population, n, can be estimated as the $\hat{n}$ which maximizes the probability of getting k unique nodes in s samples, i.e.,

$$\hat{n} = \arg\max_{n} \mathrm{P}(k|n, s). \tag{2}$$

This is actually the similar to the *coupon collector's problem* [2], where n and k are known and s is the quantity of question.

Although Driml and Ullrich [3] proved that there exists exactly one $\hat{n}$ which maximizes $\mathrm{P}(k|n, s)$, a brute-force solution needs to compute $\mathrm{P}(k|n, s)$ with lots of n, which is expensive when n and s are large.

A simpler solution is developed by Finkelstein *et al.* [4], shown as Theorem 1.

Theorem 1: If $k < n$, $\hat{n}$ is unique and is the smallest integer $j \geq k$, which satisfies $\frac{j+1}{j+1-k}(\frac{j}{j+1})^s < 1$.

B. Finding $\hat{n}$ quickly

Finkelstein *et al.* [4] does not provide an efficient way to find $\hat{n}$. First of all, the expression $\frac{j+1}{j+1-k}(\frac{j}{j+1})^s$ is unfeasible to compute for large s as it easily causes integer overflows. When dealing with real OSNs, large sample size is inevitable. Therefore we use its equivalent form here, i.e., finding the smallest integer $j \geq k$, which satisfies

$$f(j) = \log(\frac{j+1}{j+1-k}) + s\log(\frac{j}{j+1}) < 0 \tag{3}$$

Secondly, for large OSNs, we expect that $s \ll n$, which leads to a huge search space. Thus a naive linear probing implementation is inefficient.

To find $\hat{n}$ quickly, we need to reduce the search space, which is given by the following theorem.

Theorem 2: $\hat{n} \in [k, \lceil j_c \rceil]$, where $j_c = s(k-1)/(s-k)$. Furthermore, $f(j)$ is monotonically decreasing within the interval $[k, \lfloor j_c \rfloor]$.

Proof: See the Appendix. ∎

Because $f(j)$ is monotonically decreasing within the interval $[k, \lfloor j_c \rfloor]$, a binary search (or Newton's method) can be applied to find j quickly thus the runtime complexity is $O(\log(j_c - k))$. The runtime for the naive linear probing solution is $\Theta(\hat{n} - k)$.

To evaluate the speedup, we perform $1,000$ runs with $s = 100,000$ and $n = 10,000,000$ on a dual core Intel Pentium 4 3.2Ghz PC. Our solution, combined with binary search, costs 32.8 seconds to complete $1,000$ runs whereas the naive linear proving solution needs $2,306$ seconds! In other words, our algorithm is 70 times faster.

In reality, j_c can be very large, especially when s and k are close. Heuristics may give us a better upper bound of j. In our implementation, we set the upper bound of j to be the smaller one between j_c and 4 billion thus a 32 bit unsigned integer is sufficient for j.

C. Simulation results

MLE actually does not need any graph information at all. To set up the simulation, we may simply sample s IDs uniformly with replacement from a given user ID space of size n and count the number of duplicated samples (k). Then we can evaluate the bias and variance with $\hat{n}$ and n.

Multiple tests $(1,000)$ are performed to report reliable results. Shown as Fig. 1, the estimation error diminishes quickly as more nodes are sampled. When $s = 100$K and $n = 10$M, i.e., 1% sampling, the maximum error is 0.2% for all $1,000$ tests. Meanwhile, shown as Fig.2, the standard

deviation of the estimated $\hat{n}$ exhibits the same trend. Both Fig. 1 and 2 indicate that MLE is a reliable estimator. Another observation is that the standard deviation is small comparing to the true population size. For instance, the standard deviation is 4.5% of the population when the sample size is 1%. It is a good indicator for the worst case performance of MLE.

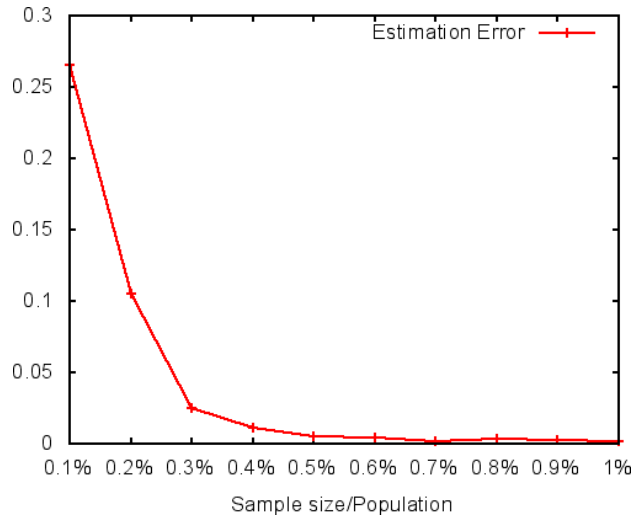


Fig. 1. Estimation error versus sample size

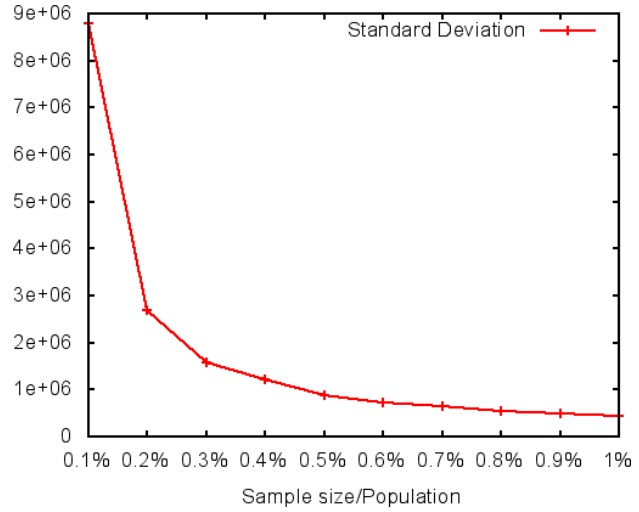


Fig. 2. Standard deviation versus sample size

D. Experiment on Twitter.com

Many OSNs provide random users upon request, therefore MLE is widely applicable. Here we apply it to Twitter.com. Twitter provides 20 most recent status messages (tweets) every 60 seconds via a service called *public timeline* (http://twitter.com/public_timeline). The statuses are chosen from non-protected users who have set a custom user icon according to Twitter API (http://apiwiki.twitter.com). We collected a sample of $9,716,838$ user IDs (s) from Feb. 2009 to Feb. 2010, among which there are $6,929,343$ unique ones (k). With MLE, we have $\hat{n} = 13,512,532$.

Moore [5] estimated that as of Jan. 2010 Twitter has 70 million users, among which only 10% are protected and 38% have never posted a single status message. Assuming these two numbers are independent, we have an estimation of non-protected users who have posted at least one status as $70 \times 90\% \times 62\% = 37.8$ millions. There are several possible causes to explain this difference.

- The messages are sampled from users who have set their user icons according to Twitter API therefore our result suggests that 35.7% of users set their customized user icons.
- Twitter is a fast changing website. Its definition of *public timeline* may have changed over the year. For example, as we have noticed, users with default icons are also shown in the public timeline.
- The sample may not be uniform across all users. If some users get higher probability to be selected, underestimation is likely to happen.
- The network grows during the sampling period. In the past 12 months, Twitter has almost doubled its size [5].

Nevertheless, the estimation here gives us some insight into the number of active twitter users and verifies that there are a large portion of users who actually never post any status messages [5].

IV. Estimating with Random Walk

When uniform sampling is unavailable or we can not get a lot of uniform samples, random walk can be applied. Section IV-A introduces an unbiased estimator (named as RW in this paper) proposed by Marchetti-Spaccamela [6] to estimate the number of nodes being connected to a certain node. The experimental results on five real OSNs are presented in Section IV-B.

A. Estimating the number of nodes connected to a node

Given a node v_0, Marchetti-Spaccamela [6] estimates the number of nodes connecting to v_0 with the following two-phase procedure.

1) **Forward walking**: Find a random acyclic path $\mathcal{P}$ starting from v_0. A random walker starts from v_0 and uniformly selects a neighbor of v_0 to proceed. It terminates when it reaches a node without any outlinks (dead ends) or a node it has previously visited (acyclic path). Then $\mathcal{P}$ consists of all the nodes the random walker has visited, except the terminating node.
2) **Back tracing**: For each node $v \in \mathcal{P}$, we try to find an acyclic path between v_0 and v in the reverse graph. Given a graph $G(V, E)$, its reverse graph is defined as $G'(V, E')$ where $\forall \overrightarrow{ij} \in E$, we have $\overrightarrow{ji} \in E'$ and vice versa. More specifically, in the reverse graph a random walker starts at v and proceeds as what we do in the forward walking phase. When the random walker reaches a dead end or any previously seen node, it restarts from v. The back tracing phase ends when the random walker reaches v_0.

To generalize this estimator in Section V, we present its proof in detail, which is initially provided by Marchetti-Spaccamela [6].

Given a random acyclic path $\mathcal{P}$ starting from v_0, let $\mathcal{P}_v$ be the prefix of $\mathcal{P}$ finishing at v, i.e., $\mathcal{P}_v =< v_0, v_1, \ldots, v >$. $\Pi_{\text{out}}(\mathcal{P}_v)$ denotes the product of the out degrees of all the nodes in $\mathcal{P}_v$ except the last node v. Similarly $\Pi_{\text{in}}(\mathcal{P}_v)$ denotes the product of the in degrees of all the nodes in $\mathcal{P}_v$ except the first node v_0. Finally, we define $b_{\mathcal{P}}(v) = \Pi_{\text{out}}(\mathcal{P}_v)/\Pi_{\text{in}}(\mathcal{P}_v)$ and $c_{\mathcal{P}}(v)$ to be the number of acyclic paths the random walker generates in the back tracing phase. b_v and c_v are used as shorthands when there is no confusion of $\mathcal{P}$ given the context. We have the following theorem.

Theorem 3: $\sum_{v\in\mathcal{P}} b_v c_v$ is an unbiased estimator of the number of nodes connected to v_0.

Proof: To see this estimator is unbiased, we just need to show that its expected value is the number of all the nodes connected to v_0. The key to the proof is to rewrite the expectation, a summation over all possible paths ($\mathcal{P}$), as a summation over all the nodes connected to v_0, i.e.,

$$E(\sum_{v\in\mathcal{P}} b_v c_v) = \sum_{\mathcal{P}}[(\sum_{v\in\mathcal{P}} b_v c_v)\text{P}(\mathcal{P})] \quad (4)$$

$$= \sum_{v}\sum_{q\in Q_v} b_q(v)c_q(v)\text{P}(q) \quad (5)$$

$$= \sum_{v} E(b_q(v)c_q(v)) \quad (6)$$

where $\text{P}(\mathcal{P})$ is the probability that $\mathcal{P}$ is chosen by the random walker, Q_v is the set of all the acyclic paths from v_0 to v, and $\text{P}(q)$ is the probability for a random walker to walk through path q. To get the total number of nodes connected to v_0, we just need to show that $E(b_q(v)c_q(v)) = 1$.

$b_q(v)$ is determined by the random path q and $c_q(v)$ is determined by the number of random walks needed to back trace v from v_0. They are independent of each other because the random walk is memoryless. Therefore we have

$$E(b_q(v)c_q(v)) = E(b_q(v))E(c_q(v)) \quad (7)$$

The probability for the random walker to walk through a path $q \in Q_v$ is $\frac{1}{\Pi_{\text{out}}q}$. Hence we have

$$E(b_q(v)) = \sum_{\forall q\in Q_v} b_q(v)\text{P}(q) \quad (8)$$

$$= \sum_{\forall q\in Q_v} \frac{\Pi_{\text{out}}(q)}{\Pi_{\text{in}}(q)}\text{P}(q) \quad (9)$$

$$= \sum_{\forall q\in Q_v} \frac{1}{\Pi_{\text{in}}(q)} \quad (10)$$

Let Q'_v be the set of all acyclic paths in the reverse graph G' starting from v to v_0. The probability for the random walker to walk through a path $q \in Q'_v$ is

$$\sum_{\forall q\in Q'_v} \frac{1}{\Pi_{\text{out}}(q)} \quad (11)$$

Notice that a path in G' from v to v_0 actually corresponds to a path in G from v_0 to v and the out degree of v in the G' is the in degree of v in G. Therefore we have

$$\sum_{\forall q\in Q_v} \frac{1}{\Pi_{\text{in}}(q)} = \sum_{\forall q\in Q'_v} \frac{1}{\Pi_{\text{out}}(q)} \quad (12)$$

Combining (10) and (12), $E(b_q(v))$ is the probability for the random walker to find an acyclic path from v to v_0 in the reverse graph, which we denote as $\text{P}(v \to v_0)$.

$E(c_q(v))$ is the number of random walks needed to get a path in Q'_v. As the random walks are independent of each other, $c_q(v)$ follows a geometric distribution with parameter $\text{P}(v \to v_0)$ thus $E(c_q(v)) = \frac{1}{\text{P}(v\to v_0)}$.

Therefore we have $E(b_q(v))E(c_q(v)) = 1$. ■

If $G(V, E)$ is a connected graph, i.e., there exists a path between any two nodes in G, then the number of nodes connected to any node v is $n - 1$, where n is the size of G. As a special case, for a node on an undirected graph, its in degree is equal to its out degree thus we have $b_v = v_0/v$, i.e., given a path $\mathcal{P}$ from node v_0 to v, b_v is solely decided by v_0 and v_i.

B. Experiments on real OSNs

To evaluate the RW estimator, we use four real OSN graphs collected by Mislove *et al.* [7]. Being published in 2007, these four graphs have been widely used in OSN studies. These four OSNs have different focuses. Flickr[2] specializes in photo sharing, YouTube focuses on video sharing, and LiveJournal[3] and Orkut[4] are general social websites. In addition, we crawled the entire Buzznet.com, another general OSN website. Therefore the results reported here may apply to a variety of OSNs.

To avoid being trapped in small regions which are not connected to the majority of the graph, we use the largest connected components (LCC). Since these graphs are highly symmetrical and well connected, their LCCs covers a large portion of the original graphs, shown as Table I. Orkut is supposed to be a bi-directed graph but the data we got from Mislove *et al.* [7] do have some asymmetrical links.

TABLE I
LARGEST CONNECTED COMPONENTS VERSUS THEIR ORIGINAL GRAPHS

Graph	Nodes in LCC	Links in LCC
Buzznet	85.6%	72.6%
Flickr	79.4%	60.6%
LiveJournal	87.7%	72.7%
Orkut	99.9%	95.2%
YouTube	86.9%	78.3%

Unless explicitly specified, for the rest of the paper we use LCCs instead of the whole graphs. Table II summaries the basic properties of these five LCCs.

[2]http://www.flickr.com
[3]http://www.livejournal.com
[4]http://www.orkut.com

TABLE II
GRAPHS USED IN OUR EXPERIMENTS

Graph (LCC)	Total Nodes	Total Links	Mean Degree	Clustering Coefficient
Buzznet	423,020	6,616,264	15.6	0.221
Flickr	1,144,940	13,709,764	12.0	0.136
LiveJournal	4,033,137	56,296,041	14.0	0.317
Orkut	2,997,166	212,698,418	71.0	0.170
YouTube	495,957	3,873,496	7.8	0.110

With LCCs, the number of nodes connected to any node is $n-1$. We randomly select $2,000$ nodes as v_0 from each of the five graphs and the results given by the RW estimator are shown as Fig. 3.

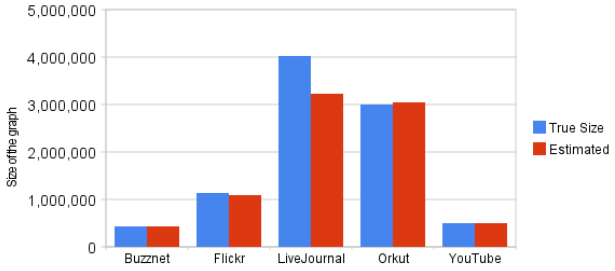


Fig. 3. Estimating the size of five OSNs with RW. Left bars represent the true sizes and right bars represent the estimated sizes.

The small gaps between the true sizes and the estimated sizes suggest that RW gives a good estimation to all these graphs. The large estimation error over LiveJournal indicates that more tests are needed for getting reliable results on LiveJournal.

It might be tempting to understand why some graphs are overestimated whereas others are underestimated. Examining the results indicates that this difference is probably the result of variance. According to the law of large numbers, the more tests we have, the closer the mean of the tests is to the expected value. Shown as Fig. 4, the estimation on Buzznet approaches its true size quickly as the number of tests increases while the estimation on Flickr fluctuates.

To see if the estimation is sensitive to the selection of v_0, we select v_0 with different degrees, i.e., $k(v_0)$. For $k(v_0) = 10, 20, \ldots, 100$, the results do not show significant difference between each other, in terms of bias and variance. One interesting observation is that the number of nodes crawled during the estimation becomes smaller when high degree nodes are chosen for v_0, shown as Fig. 5. To compare the results on graphs with different size, the number of nodes crawled is normalized by the number of nodes crawled when $k(v_0) = 10$. Identifying the graph structure which causes this effect is an interesting direction for future work.

V. GENERALIZING THE RW ESTIMATOR

The proof of Theorem 3 actually provides a way to estimate other quantities besides the number of nodes in a graph.

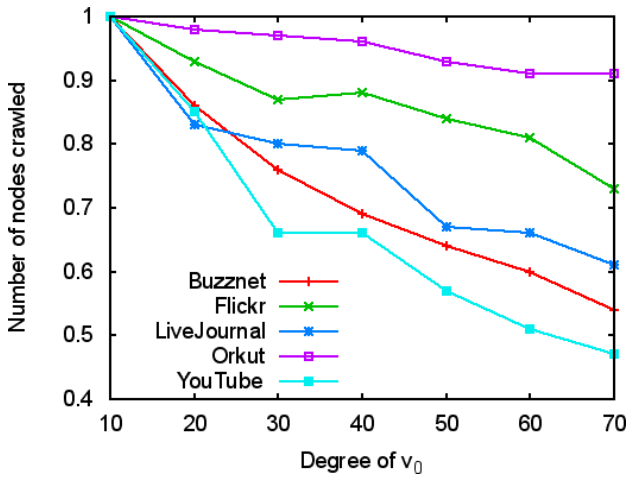


Fig. 5. Number of nodes crawled (normalized by the number of nodes crawled when $k(v_0) = 10$) versus the degree of v_0.

Corollary 1: If $\omega(v)$ is not a random variable given node v, $\sum_{v\in\mathcal{P}} \omega(v) b_v c_v$ is an unbiased estimator of $\sum_v \omega(v)$.

Proof: Following Theorem 3 we have

$$E(\sum_{\forall v\in\mathcal{P}} \omega(v) b_v c_v) = \sum_v E(\omega(v) b_v c_v) \quad (13)$$

$$= \sum_v \omega(v) E(b_v c_v) \quad (14)$$

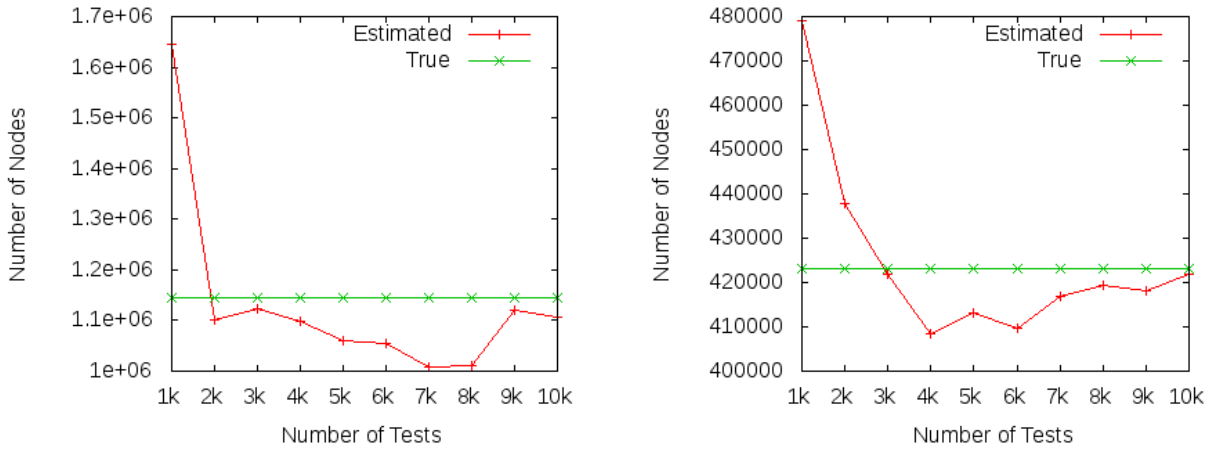
$$= \sum_v \omega(v) \quad (15)$$

■

For example, if we set $\omega(v)$ to be the degree of v, then $\sum_v \omega(v)$ is the total number of links of the graph.

More generally, we have the following corollary:

Corollary 2: If $\omega(v)$ is a random variable independent of $b(v)$ and $c(v)$, $\sum_{\forall v\in\mathcal{P}} \omega(v) b_v c_v$ is an unbiased estimator of $\sum_v E(\omega(v))$.



(a) Flickr (b) Buzznet

Fig. 4. Estimation versus number of tests

Proof: Following Theorem 3 we have

$$E(\sum_{\forall v\in\mathcal{P}} \omega(v)b_v c_v) = \sum_v E(\omega(v)b_v c_v) \quad (16)$$

$$= \sum_v E(\omega(v))E(b_v)E(c_v) \quad (17)$$

$$= \sum_v E(\omega(v)) \quad (18)$$

■

Similar results as Section IV-B are got when applying RW to estimate the number of links a graph has, thus are omitted here due to the page limit.

As an important metric for small world graphs, the clustering coefficient is computed and analyzed in many social network studies [8]. With Corollary 1, we may estimate the total clustering coefficient of a network ($\sum cc$) if we set $\omega(v)$ to be the clustering coefficient of node v. Then an estimator for the mean clustering coefficient of the network is given by

$$\frac{\widehat{\sum cc}}{\hat{n}} \quad (19)$$

This is a biased estimator because in general we have

$$E(cc) = E(\frac{\sum cc}{n}) \neq \frac{E(\sum cc)}{E(n)} \quad (20)$$

If we know the size of the graph, we can have an unbiased estimator as follows.

$$E(cc) = E(\frac{\sum cc}{n}) = \frac{E(\sum cc)}{n} \quad (21)$$

The difference is that n in (19) is a random variable thus needs to be estimated whereas the n in (21) is a known constant. The results given by these two estimators (average of $1,000$ tests) are shown as Fig. 6. To our surprise, the biased estimator actually is not much worse than the unbiased one. On the YouTube graph, it even outperforms the unbiased estimator. A possible resolution is that the error of $\hat{n}$ introduced by the random walker can be compensated by the error of $\widehat{\sum cc}$ in the same run. Meanwhile, our analysis on the variance suggests more tests for the unbiased estimator to further approach the true value.

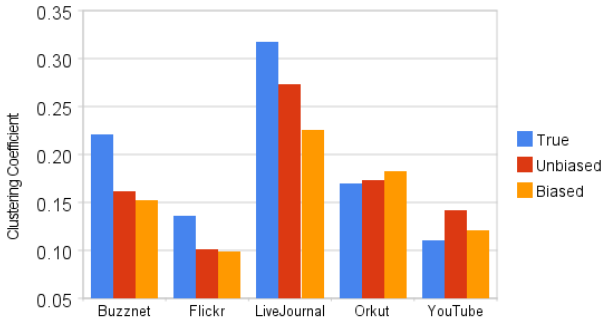


Fig. 6. Estimating the clustering coefficient of five OSNs with RW. Left bars represent the true value, middle bars represent the unbiased estimation given by (21) and right bars represent the biased estimation given by (19).

VI. Pitfalls

This section presents the pitfalls and challenges we encountered when developing estimators for OSNs. We believe that the discussions here provide valuable insights for future work.

A. Combining MLE with MHRW

Gjoka *et al.* [9] proposed an unbiased sampling method, Metropolis-Hastings Random Walk (MHRW), to sample OSNs. We tried to combined MLE with MHRW. This approach would have two advantages.

- It does not rely on the OSN to provide uniform samples.
- It does not need to crawl lots of nodes.

This idea, however, does not work. Let $\Psi = \{v_1, v_2, \ldots, v\}$ be the set of nodes crawled by MHRW. MHRW generates a uniform sampling *within* Ψ, therefore combining MLE with MHRW, we are only able to estimate the size of Ψ.

B. Large variance and expensive back tracing

There are two problems when applying RW to large OSNs. First of all, RW generates large variances in its estimation. For example, among $30,000$ tests on Buzznet, only 24% of them give have estimation error within 50%. To get reliable estimations, we need the mean of multiple runs.

Secondly, during back tracing, RW may need lots of random walks before it reaches v_0. For instance, in rare cases, more than 90% of the graph are crawled to complete the back tracing.

It is possible to replace the back tracing with a breadth first search (BFS) and estimate a lower bound for the number of random walks needed to reach v_0, but due to the small world property of OSNs, BFS is also likely to cover a large portion of the graph, as shown by Ye *et al.* [10].

To finish the back tracing quickly, parallel random walkers can be used while careful coordination needs to be implemented. Once a random walker reaches v_0, all the random walkers need to be stopped (to save the crawling costs) and the random walkers starting after the one which reaches v_0 should be discarded (otherwise the number of random walks needed to reach v_0 will be overestimated). Caching visited nodes across multiple random walkers also needs to be employed to avoid duplicated crawlings.

C. Poor lower and upper bounds for RW

Marchetti-Spaccamela [6] proposed an estimator for the lower bound of RW, which performs up to k random walks in the back tracing phase. If after k random walks, v_0 is still not reached, k is returned as the minimum number of random walks needed to reach v_0. Without prior knowledge of the graph size, however, it is nontrivial to set a proper k: Large ks do not serve the purpose of accelerating the back tracing whereas small ks produce poor lower bounds. In our experiments, with $k = 1$M, the lower bounds given by this estimator are still poor for large graphs such as LiveJournal and Orkut.

An upper bound is also proposed in [6] which simply assumes that G is a tree instead of a graph. It greatly overestimates the size when G is large. On our graphs the upper bound estimator overestimates by several magnitudes.

D. Expensive computation costs in simulations

Simulations on large graphs such as OSNs are computationally expensive. In our tests, we used a cluster of 36 PCs with two AMD Opteron 2.6GHz CPU/4GB memory per node. It took us a month to finish various tests presented here. To improve the performance, we have the following optimizations.

- *Tight data structures*: An adjacency list (array) is used to represent the neighbors of each node. A large array is used to keep pointers to these lists therefore random access to any list is simple and fast. Initially we used the IDs coming with the datasets, which are not continuous especially after we extracted LCCs. To make the array as small as possible, we reassign the IDs for the nodes such that they are continuous. We also sort the IDs within each adjacency list such that when computing clustering coefficients, a binary search can be applied to check if the list has a certain node.
- *Accelerating frequently executed routines*: As a lot of similar operations are performed multiple times, making such components fast greatly reduces the simulation time. For example, we need to set/check whether a node has been visited when trying to find an acyclic path. This was initially implemented with a *set* from the C++ Standard Template Library. Set is implemented with trees but we actually do not need to keep the order of the items. Then we changed to *unordered_set*, which is a hash table. Still not satisfied with its performance, we finally replaced it with a bit field where each bit corresponds to a node, then the check is done by simply examining whether the bit is set or not.
- *Improving data locality*: The random walker incurs a lot of cache misses, which become the main bottleneck of our simulation. More specifically, after the random walker visits a node v, it needs to check v's neighbors, say N_v, then it moves to $u \in N_v$ and checks u's neighbors, say N_u. If the two adjacency lists corresponding to N_v and N_u are far away from each other, a cache miss is likely to happen. To alleviate this problem, we malloc a large continuous memory pool for the adjacency lists and rearrange them to keep adjacency lists of neighbors close to each other in the memory layout.

With all the optimizations applied, the simulation runs $10-20$ times faster.

VII. Related work

Mark and recapture is a method commonly used in ecology to estimate the population of a certain species [11]. The basic idea is that m nodes are sampled uniformly from the population without replacement, and returned to the population after they are marked. Then C nodes are sampled from the population without replacement. Assuming R nodes of the second sample set are marked (a.k.a. they are in the prior sample set), the population size N can be estimated as MC/R. This estimator, named as the Lincoin-Petersen method, has been extended with various assumptions and settings, such as using multiple sample sets instead of two and deciding m and C adaptively according to the previous R. This family of approaches can be combined with our estimating with uniform sampling method. On the other hand, few OSNs supports the functionality to provide a sample set directly. Multiple single node samplings have to be employed therefore proper m and C need to be chosen, which complicates the estimation.

Knuth [12] proposed an unbiased estimator to estimate the size of a tree with random walkers. Pitt [13] extended this estimator to estimate the size of directed acyclic graphs (DAGs). These two pieces of work inspire the RW estimator, which we have discussed in detail in Section IV-A.

Besides estimating the size of graphs, there are studies on estimating other graph properties but few of them focuses on

OSNs. Tsonis *et al.* [14] proposed an estimator for clustering coefficient in scale free networks. Magnien *et al.* [15] developed a fast algorithm to find tight bounds for diameters. Becchetti *et al.* [16] proposed two approximation algorithms to count the number of triangles incident to every node in a graph. Buchsbaum *et al.* [17] considered the problem of find common neighbors between two nodes. It would be interesting to see how these estimators work on OSNs.

VIII. Conclusions

In this paper we introduce two existing estimators to estimate the size of OSNs. The first estimator, MLE, relies on uniform sampling and gives accurate estimates with a small sample size. We develop an algorithm to solve the MLE problem with logarithm runtime complexity, which is 70 times faster than the naive linear probing method in our experiment. We extend the second estimator, RW, to estimate other graph properties such as clustering coefficient. Real OSNs are used to evaluate the bias and variance of these estimators. We also discuss the pitfalls we had when developing these estimators.

Evaluating the resource requirement of RW and other random walk based estimators will be an interesting direction to explore. More specifically, we are interested in how these estimators perform given the number of nodes they can crawl and try to reduce such resource requirement without losing too much accuracy.

We believe that this paper for the first time addresses the challenges of estimating on OSNs and the analysis and pitfalls presented here provides us with valuable insight for developing such estimators.

Acknowledgements

The authors would like to thank Alan Mislove for sharing with us the social graphs and Matt Spear for crawling the Buzznet graph. Norm Matloff provided valuable comments to improve the paper, especially in formulating the MLE problem. The authors are also grateful to Jeff Rowe and Prantik Bhattacharyya for helpful references.

Appendix
Proof of Theorem 2

Proof: Notice that

$$f(j) = (1-s)\log(j+1) - \log(j+1-k) + s\log(j) \quad (22)$$

$$\frac{df(j)}{dj} = (1-s)\frac{1}{j+1} - \frac{1}{j+1-k} + s\frac{1}{j} \quad (23)$$

$$= \frac{(s-k)j - s(k-1)}{j(j+1)(j+1-k)} \quad (24)$$

Let $\frac{df(j)}{dj} = 0$ and solve for critical points. Considering $j \geq k$, $s \geq k$, and $k > 0$, there is only one critical point $s(k-1)/(s-k)$, denoted as j_c. Moreover, we have

$$\frac{df(j)}{dj} \begin{cases} < 0 & \text{if } j < j_c \\ > 0 & \text{if } j > j_c \end{cases}$$

In other words, $f(j)$ is initially monotonically decreasing and then monotonically increasing. If $\hat{n} < \lceil j_c \rceil$, there must exists another integer $j < \hat{n}$ which also satisfies $f(j) < 0$. This, however, contradicts the assumption that $\hat{n}$ is the smallest number which satisfies $f(j) < 0$). Therefore, we have $j \leq \lceil j_c \rceil$. We also need to check if $j = \lceil j_c \rceil$ because it is possible that $f(\lfloor j_c \rfloor) > 0$ and $f(\lceil j_c \rceil) < 0$. ■

References

[1] R. Gross, A. Acquisti, and H. J. Heinz, III, "Information revelation and privacy in online social networks," in *WPES '05: Proceedings of the 2005 ACM workshop on privacy in the electronic society*, 2005, pp. 71–80.

[2] W. Feller, *Introduction to probability and its applications*, 2nd ed. John Wiley, 1957, vol. 1.

[3] M. Driml and M. Ullrich, "Maximum likelihood estimate of the number of types," *Acta Technica ČSAV*, pp. 300–303, 1967.

[4] M. Finkelstein, H. G. Tucker, and J. A. Veeh, "Confidence intervals for the number of unseen types," *Statistics and probability letters*, vol. 37, no. 4, pp. 423–430, March 1998.

[5] R. J. Moore, "New data on Twitter's users and engagement," 2010, http://themetricsystem.rjmetrics.com/2010/01/26/new-data-on-twitters-users-and-engagement/.

[6] A. Marchetti-Spaccamela, "On the estimate of the size of a directed graph," *Graph-theoretic concepts in computer science*, vol. 344, pp. 317–326, 1989.

[7] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *IMC'07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007, pp. 29–42.

[8] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[9] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Walking in Facebook: A case study of unbiased sampling of OSNs," in *Proceedings of the 2010 IEEE Infocom conference*, 2010.

[10] S. Ye, J. Lang, and F. Wu, "Crawling online social graphs," in *APWeb '08: Proceeding of the 2010 Asia Pacific Web conference*, 2010.

[11] G. Seber, *The Estimation of Animal Abundance*. The Blackburn Press, 1982.

[12] D. E. Knuth, "Estimating the efficiency of backtrack programs," *Mathematics of computation*, vol. 29, no. 129, pp. 121–136, 1975.

[13] L. Pitt, "A note on extending Knuth's tree estimator to directed acyclic graphs," *Information processing letters*, vol. 24, no. 3, pp. 203–206, 1987.

[14] A. Tsonis, K. Swanson, and G. Wang, "Estimating the clustering coefficient in scale-free networks on lattices with local spatial correlation structure." *Physica A: Statistical mechanics and its applications*, vol. 387, pp. 5287–5294, 2008.

[15] C. Magnien, M. Latapy, and M. Habib, "Fast computation of empirically tight bounds for the diameter of massive graphs," *Journals of experimental algorithmics*, vol. 13, pp. 1.10–1.9, 2009.

[16] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis, "Efficient semi-streaming algorithms for local triangle counting in massive graphs," in *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on knowledge discovery and data mining*, 2008, pp. 16–24.

[17] A. L. Buchsbaum, R. Giancarlo, and B. Racz, "New results for finding common neighborhoods in massive graphs in the data stream model," *Theoretical computer science*, vol. 407, no. 1-3, pp. 302–309, 2008.