Estimating the Size of Online Social Networks

Shaozhi Ye*

Google Inc.
1600 Amphitheatre Pkwy
Mountain View, CA 94043, USA
Email: yeshao@gmail.com
*Corresponding author

S. Felix Wu

Department of Computer Science
University of California, Davis
One Shields Ave
Davis, CA 95616, USA
Email: 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 three 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. In our experiments it is 70 times faster than the naive linear probing algorithm. The second estimator is mark and recapture (MR), which we employ to estimate the number of Twitter users behind its public timeline service. The third estimator (RW) is based on random walkers and is generalized to estimate other graph properties. In-depth evaluations are conducted on six real OSNs to show the bias and variance of these estimators. Our analysis addresses the challenges and pitfalls when developing and implementing such estimators for OSNs.

Keywords: Online social networks; Maximum likelihood estimation; Mark and recapture; Random walker.

Reference to this paper should be made as follows: Ye, S. and Wu, S.F. (xxxx) ‘Estimating the size of online social networks’, *Int. J. Social Computing and Cyber-Physical Systems*, Vol. x, No. x, pp.xxx–xxx. A preliminary version of this paper appeared in Proceedings of the Second IEEE International Conference on Social Computing (SocialCom2010).

Biographical notes: Shaozhi Ye received his Ph.D in computer science in 2010 from University of California, Davis. He is currently working as a software engineer at Google Inc. His research interests include Web search, online social network analysis, distributed systems and machine learning. This work was conducted while he was a graduate student at UC Davis.
S. Felix Wu received his Ph.D in computer science in 1995 from Columbia University. He is currently a professor at University of

California, Davis. His latest focus is on the Davis Social Links (DSL) project, which is currently sponsored by NSF/FIND, NSF/BBN/GENI, Army/ARO/MURI, Air Force/AFOSR/MURI, and most recently ARL/CTA Network Science.

1 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 are challenging to measure because they require the knowledge of the entire graph. 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 if they issue 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 needed in almost all social network 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 number of users 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 provide the number of *active users* but the definition of an active user 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 (`http://www.facebook.com`).
- It is possible to estimate the used portion of the non-sequential ID space by probing the ID space. For example, for every k IDs, we randomly sample one ID and check if it is valid. Assuming we get c valid IDs out of these samples, the total number of valid user IDs can be simply estimated as ck. This approach fails when IDs are assigned non-uniformly. There are more sophisticated approaches to probe non-uniformly distributed ID space, but they are expensive as they need to examine a large number of IDs.
- To make things worse, numeric user IDs are unavailable on some OSNs, such as YouTube (`http://www.youtube.com`), 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 three estimators which do not rely on how user IDs are assigned.

- *MLE*: Maximum likelihood estimation based on uniform sampling.

- *MR*: Mark and recapture estimator based on uniform sampling.
- *RW*: Unbiased estimator based on random walkers.

All these 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.
- The requirement of MR is very similar to that of MLE. It requires the capability to sample m users uniformly without replacement from the OSN.
- 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 (Gross, Acquisti & Heinz 2005).

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 estimators with real OSNs. Our contributions are summarized as follows.

- Three estimators are introduced to estimate the size of OSNs by using widely available OSN services. (Section 3, 4, and 5)
- An $O(\log n)$ algorithm is developed to solve the MLE problem quickly (70 times faster in our experiments). (Section 3.2)
- MLE and MR are employed to estimate the number of users behind Twitter's public timeline service. (Section 3.4 and 4)
- In-depth evaluations are conducted over real OSNs to examine their bias and variance. (Section 3.3, 3.4 and 5.2)
- 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 6)
- We also discuss the challenges and pitfalls when developing these estimators. (Section 7)

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.

2 Preliminaries

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

2.1 Social graph

By treating a user as a node (or vertex) and the *friend* relationship between two users as a link (or 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.

Different OSNs have different definitions of *friends*. Some allow friendship to be asymmetrical, i.e., Alice can choose Bob as her friend while Bod does not need to set Alice as his friend, while others require friendship to be symmetrical, i.e., both Alice and Bob need to specify each other as a friend. Symmetrical (or mutual) friendship corresponds to undirected graphs and asymmetrical friendship leads to directed graphs. In this paper 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 which are linked by u, i.e., $\{v|\forall v, \exists e_{uv} \in E\}$. In this paper, we assume that when a node is *crawled* (or *visited*), its neighbors are known, but the neighbors of its neighbors are not. This definition corresponds to the graph crawling process in real world. Crawling an OSN user may also include crawling his/her profiles, messages, etc., but in this paper we only need the lists of friends.

2.2 Bias and variance

Formally, the estimation problem can be expressed as follows: Given G, can we estimate n without crawling the entire G? This paper answers this question with three estimators. To evaluate how good these estimators are, the main performance metrics being used here are *bias* and *variance*.

Bias quantifies the expected difference between the estimated size ($\hat{n}$) and the real size (n) of the graph. In other words, if we use an estimator to estimate the size of the graph k times and each time we compute how far away our estimated $\hat{n}$ is from the true n, the mean of these k differences is the bias of this estimator when $k \to \infty$. As 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. Given two estimators with the same bias, the one giving more consistent results is preferred in most cases.This paper uses its squared root, i.e., standard deviation.

3 Estimating with uniform sampling

Assuming we are able to sample uniformly over an OSN with replacement, as more and more nodes are sampled, the probability of sampling a previous sampled node again increases. Given the number of samples, the smaller the OSN is, the more duplicate samples we may get. Based on this intuition, we develop a maximum-likelihood estimator for the size of the OSN. Many OSNs provide random users upon request, therefore this approach is widely applicable.

We first formulate this MLE problem in Section 3.1 and then develop a fast solution in Section 3.2. Section 3.3 gives the simulation results for its bias and variance and Section 3.4 presents our experiment on `Twitter.com`.

3.1 Maximum likelihood estimator

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

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

Driml and Ullrich (Driml & Ullrich 1967) proved that there exists exactly one $\hat{n}$ which maximizes $\mathrm{P}(k|n, s)$. Therefore a brute-force solution is to compute $\mathrm{P}(k|n, s)$, starting with $n = k$ and ending with the first n which satisfies

$$\mathrm{P}(k|n, s) > \mathrm{P}(k|n + 1, s) \tag{3}$$

To compute $\mathrm{P}(k|n, s)$, we first choose k unique nodes from n, which is $\binom{n}{k}$, then assign k labels (node ids) to s samples, i.e.,

$$\mathrm{P}(k|n, s) = \frac{\binom{n}{k} F(k, s)}{n^s} \tag{4}$$

where $F(k, s)$ can be thought as the number of different placements of s different balls in k different urns. Considering the sth ball, if there is one empty urn left, the sth ball has to be put into this urn, thus the problem becomes putting $s - 1$ different balls into $k - 1$ different urns, i.e., $F(k - 1, s - 1)$. Considering that the empty urn can be any of the k urns, the number of different placements is $kF(k - 1, s - 1)$. If there is no empty urns, the sth ball can be put into any urn, thus there are $kF(k, s - 1)$ placements following the same reasoning. Combining these two cases, we have

$$F(k, s) = kF(k - 1, s - 1) + kF(k, s - 1) \tag{5}$$

Therefore a recurrence can be established to compute $F(k, s)$ as proposed by Dhakar and Mattheiss (Dhakar & Mattheiss 1989). This linear search solution, however, does not scale to large n and s.

A simpler solution is developed by Finkelstein *et al.* (Finkelstein, Tucker & Veeh 1998), shown as Theorem 1.

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

3.2 Finding $\hat{n}$ quickly

Finkelstein *et al.* (Finkelstein et al. 1998) 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

$$f(j) = \log(\frac{j+1}{j+1-k}) + s\log(\frac{j}{j+1}) < 0 \tag{6}$$

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

To find $\hat{n}$ quickly, we have the following theorem to reduce the search space.

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: Notice that

$$\begin{aligned} f(j) &= (1-s)\log(j+1) - \log(j+1-k) + s\log(j) &(7)\\ \frac{df(j)}{dj} &= (1-s)\frac{1}{j+1} - \frac{1}{j+1-k} + s\frac{1}{j} &(8)\\ &= \frac{(s-k)j - s(k-1)}{j(j+1)(j+1-k)} &(9) \end{aligned}$$

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 $f(\lceil j_c \rceil) < 0$. □

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. Therefore the runtime complexity can be reduced to $O(\log(j_c - k))$. 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.

3.3 Simulation results

MLE neither needs any graph information nor assumes any graph properties. It just requires the capability to do uniform sampling over the graph. To set up the simulation, we 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 Figure 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 Figure2, the standard deviation of the estimated $\hat{n}$ exhibits the same trend. Both Figure 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 size when the sample size is 1%. It is a good indicator for the worst case performance of MLE.

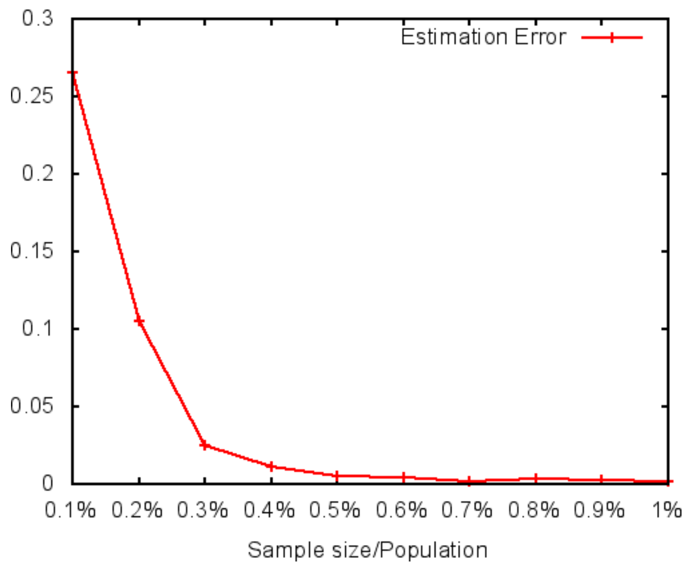


Figure 1 Estimation error versus sample size

3.4 Experiment on 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 messages are chosen from "non-protected users who have set a custom user icon" according to Twitter API (`http://apiwiki.twitter.com`). From Feb. 2009 to Nov. 2010, we collected $17,281,077$ user IDs (s) through the public timeline feed, among which there are $6,929,343$ unique ones (k). With MLE, we have $\hat{n} = 7,772,803$.

Moore (Moore 2010) 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

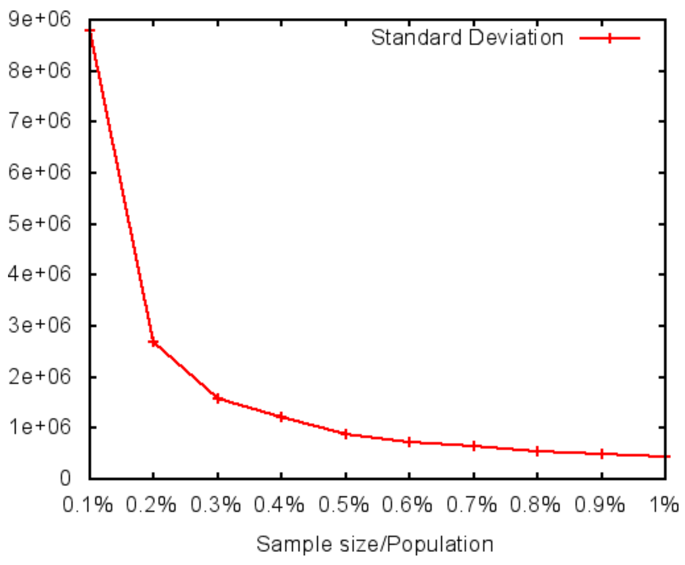


Figure 2 Standard deviation versus sample size

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 the huge gap.

- These messages are sampled from a smaller population as described by Twitter API.
- The sample may not be uniform across all users. The public timeline service provides random samples of messages instead of users. As some users post more messages than others, this sample is likely skewed towards users with more messages, which leads to underestimation of the size.
- The network grows during the sampling period. In the past 12 months, Twitter has almost doubled its size (Moore 2010). It is hard to estimate a moving target.
- 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.

4 Mark and recapture

Twitter's public timeline data are a good candidate for *mark and recapture*, which is commonly used in ecology to estimate the population size of a certain species (Seber 1982). The basic idea is that if we perform two independent samplings on the same population, the smaller the population is, the larger the overlap between these two samples is likely to be. More specifically, 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 in the second sample are marked (a.k.a. they are in the prior sampled list), the population size n can be estimated as

$$\hat{n} = \mathrm{mC/R} \tag{10}$$

As shown in Section 3.4, it is hard to make a good estimation if the underlying population is growing. To use this data, we make a more realistic assumption: the size of this user pool does not change a lot within a short period of time. More specifically, we split the data by days, then use the list of ids in Day i as the marked sample and the list of ids in Day $i+1$ as the recaptured sample. Therefore we can make a sequence of estimations, shown as Figure 3.

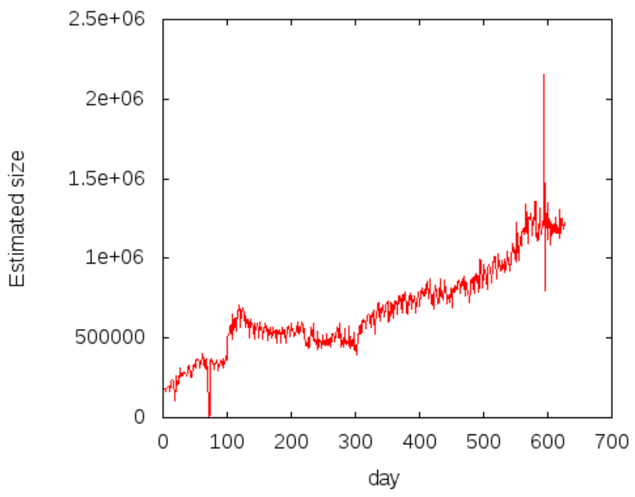


Figure 3 Estimating with mark and recapture

This gives us some idea of the number of Twitter users covered by its public timeline service. Although the estimated size of this user pool, 1.2 million, is small compared to the size of the entire `Twitter.com`, which is reported as 175 million as of Oct. 2010 (Miller 2010), this corresponds to the number of users who actively post messages every day. Excluding a couple of outliers, Figure 3 shows a clear increasing trend over the past 600 days. In fact, it is 10 times larger in Nov. 2010 than what it was in Feb. 2009. It was reported that in 2009 Twitter almost doubled its total number of users. If its public timeline service uniformly samples over daily message posters, our results show that the actual user generated content on `Twitter.com` is growing even faster. More importantly, if we make a sliding window with several days as its width, the estimations are consistent within the window, which gives us some confidence of how reliable this estimator is.

This estimator, also known as the Lincoin-Petersen method, has been extended with various assumptions and settings, such as using multiple samples instead of two and deciding m and C adaptively according to the previous R. Its bias and variance have also been extensively studied thus we omit these analysis here. We believe that there exist more sophisticated methods to get better results than what we present in this section but the naive estimator already gives us some insights. As far as we know, this is the first estimation to the size of this population. As this feed is becoming widely used in many OSN studies, we encourage researchers to apply this estimation to their data.

5 Estimating with random walkers

When uniform sampling is unavailable or we can not get a lot of uniform samples, random walkers can be employed to estimate the size of the graph. This section introduces an unbiased estimator (named as RW in this paper) proposed by Marchetti-Spaccamela (Marchetti-Spaccamela 1989) and evaluates this estimator with five real OSNs.

5.1 Estimating the size of a connected graph

Given a node v_0, Marchetti-Spaccamela (Marchetti-Spaccamela 1989) 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. More specifically, the random walker works as follows.

   (a) Set $\mathcal{P} = \phi$ and the current node to be v_0.

   (b) Assuming the current node is v_{i-1}, uniformly select a neighbor v_i.

   (c) Terminate if v_i does not have any outlinks (dead ends) or is a node the random walker has previously visited (acyclic path), i.e., $v_i \in \mathcal{P}$.

   (d) Add v_i to $\mathcal{P}$, i.e., $\mathcal{P} = \{v_0, v_1, \ldots, v_i\}$.

   (e) Return to Step (b).

   When the random walker terminates, $\mathcal{P}$ 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. In the reverse graph, a random walker generates acyclic paths starting from v as what we do in the forward walking phase. The random worker terminates when it reaches v_0. If the random walker reaches a dead end or a previously seen node, it restarts from v. The only information we need to keep during this phase is the number of restarts it takes to reach v_0, i.e., the number of acyclic paths it generates.

To generalize this estimator in Section 6, we present its proof in detail, which is initially provided by Marchetti-Spaccamela (Marchetti-Spaccamela 1989).

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 $\sum\limits_{v\in\mathcal{P}} b_v c_v$ *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.,

$$E(\sum_{v\in\mathcal{P}} b_v c_v) = \sum_{\mathcal{P}}[(\sum_{v\in\mathcal{P}} b_v c_v)\mathrm{P}(\mathcal{P})] \tag{11}$$

$$= \sum_{v}\sum_{q\in Q_v} b_q(v)c_q(v)\mathrm{P}(q) \tag{12}$$

$$= \sum_{v} E(b_q(v)c_q(v)) \tag{13}$$

where $\mathrm{P}(\mathcal{P})$ 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 $\mathrm{P}(q)$ 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. $c_q(v)$ is determined by the number of random walks needed to backtrace v from v_0. They are independent of each other because the random walk is memoryless. Therefore we have

$$E(b_q(v)c_q(v)) = E(b_q(v))E(c_q(v)) \tag{14}$$

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

$$E(b_q(v)) = \sum_{\forall q\in Q_v} b_q(v)\mathrm{P}(q) \tag{15}$$

$$= \sum_{\forall q\in Q_v} \frac{\Pi_{\text{out}}(q)}{\Pi_{\text{in}}(q)}\mathrm{P}(q) \tag{16}$$

$$= \sum_{\forall q\in Q_v} \frac{1}{\Pi_{\text{in}}(q)} \tag{17}$$

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

$$\sum_{\forall q\in Q'_v} \frac{1}{\Pi_{\text{out}}(q)} \tag{18}$$

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

$$\sum_{\forall q \in Q_v} \frac{1}{\Pi_{\mathrm{in}}(q)} = \sum_{\forall q \in Q'_v} \frac{1}{\Pi_{\mathrm{out}}(q)} \tag{19}$$

Combining (17) and (19), $E(b_q(v))$ 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 $\mathrm{P}(v \to v_0)$.

$E(c_q(v))$ 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 $\mathrm{P}(v \to v_0)$ thus $E(c_q(v)) = \frac{1}{\mathrm{P}(v \to v_0)}$.

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.

5.2 Experiments on real OSNs

To evaluate the RW estimator, we use four real OSN graphs collected by Mislove *et al.* (Mislove, Marcon, Gummadi, Druschel & Bhattacharjee 2007). Being published in 2007, these four graphs have been widely used in OSN studies. These four OSNs are of different nature. Flickr (`http://www.flickr.com`) specializes in photo sharing, YouTube focuses on video sharing, and LiveJournal (`http://www.livejournal.com`) and Orkut (`http://www.orkut.com`) 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 consider the largest connected components (LCC). Since these graphs are highly symmetrical and well connected, their LCCs cover a large portion of the original graphs, shown as Table 1. Orkut is supposed to be a bi-directed graph but the data we got from Mislove *et al.* (Mislove et al. 2007) do contain some asymmetrical links. We suspect that it is caused by the crawling process. For example, node A is crawled before node B. By the time node A is crawled, A and B are not connected yet. But when node B is crawled, A and B are already connected. As it takes a large amount of time to crawl huge graphs such as Orkut, such asymmetry is likely to be introduced.

Unless explicitly specified, for the rest of the paper we use LCCs instead of the whole graphs. Table 2 summaries the basic properties of these five LCCs.

With LCCs, the number of nodes connected to any node is $n - 1$. From each of the five graphs we randomly select $2,000$ nodes as v_0. The results given by the RW estimator are shown as Figure 4.

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 this social graph.

Table 1 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%

Table 2 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

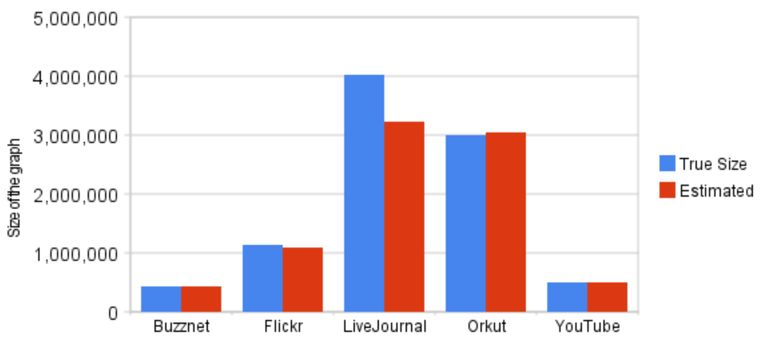
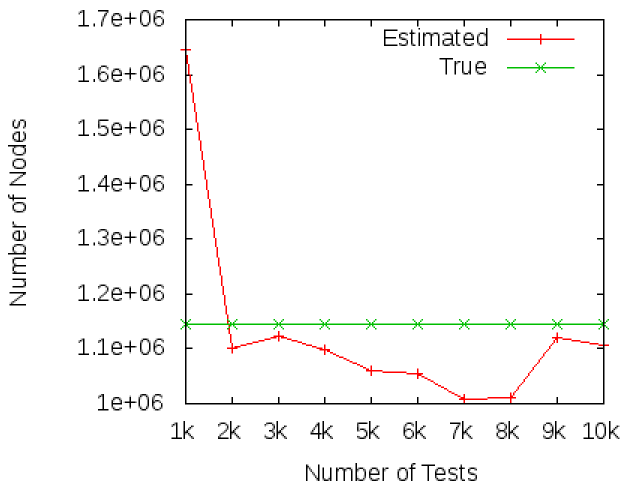
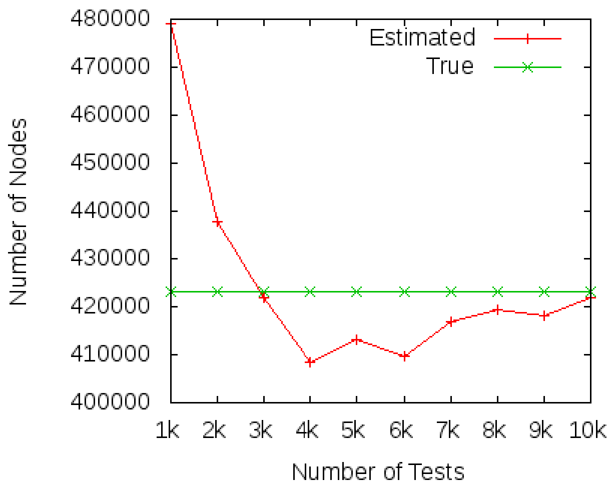


Figure 4 Estimating the size of five OSNs with RW. Left bars represent the true sizes and right bars represent the estimated sizes.

It might be tempting to understand why some graphs are overestimated whereas others are underestimated. Further examination of 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 Figure 5, the estimation on Buzznet approaches its true size quickly as the number of tests increases while the estimation on Flickr fluctuates.



(a) Flickr



(b) Buzznet

Figure 5 Estimation versus number of tests

To see whether 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 both 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 Figure 6. To compare the results on graphs with different size, the number of nodes crawled is normalized by the number of nodes crawled when $k(v_0) = 10$. For YouTube, the number of nodes which need to be crawled for $k(v_0) = 70$ is about 50% smaller than that for $k(v_0) = 10$. Identifying the graph structure which causes this effect is an interesting direction for future work.

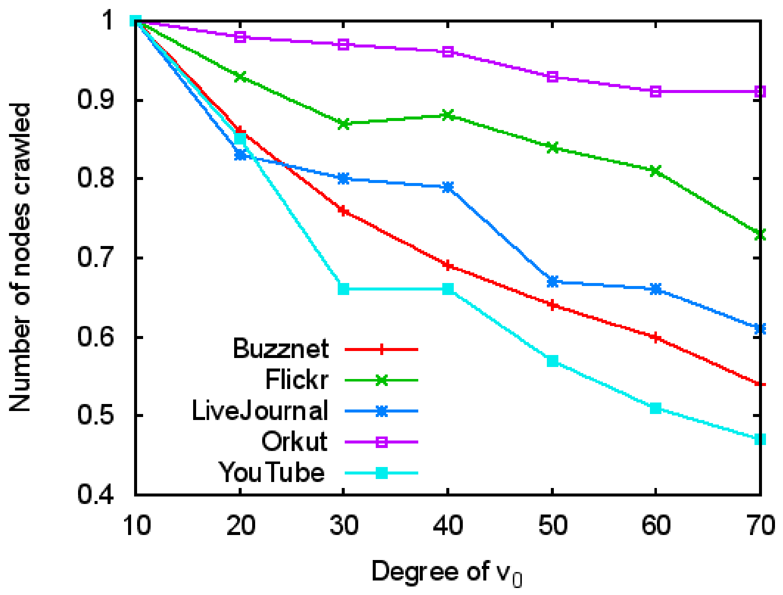


Figure 6 Number of nodes crawled (normalized by the number of nodes crawled when $k(v_0) = 10$) versus the degree of v_0.

6 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.

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

Proof: Following Theorem 3 we have

$$E(\sum_{\forall v\in\mathcal{P}}\omega(v)b_v c_v) = \sum_v E(\omega(v)b_v c_v) \tag{20}$$

$$= \sum_v \omega(v)E(b_v c_v) \tag{21}$$

$$= \sum_v \omega(v) \tag{22}$$

□

For example, if we set $\omega(v)$ to be the degree of v, then $\sum\limits_v \omega(v)$ is the total number of links of the graph. In fact, we got similar results as Section 5.2 when applying RW to estimate the number of links a graph has.

More generally, we have the following corollary:

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

Proof: Following Theorem 3 we have

$$E(\sum_{\forall v \in \mathcal{P}} \omega(v) b_v c_v) = \sum_v E(\omega(v) b_v c_v) \tag{23}$$

$$= \sum_v E(\omega(v)) E(b_v) E(c_v) \tag{24}$$

$$= \sum_v E(\omega(v)) \tag{25}$$

□

In other words, as long as the quantity of interest can be expressed as a random variable which is independent of $b(v)$ and $c(v)$, we can estimate it with this approach.

6.1 Estimating clustering coefficient

As an important metric for small world graphs, the clustering coefficient is computed and analyzed in many social network studies (Watts & Strogatz 1998). 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

$$\frac{\widehat{\sum cc}}{\hat{n}} \tag{26}$$

This is a biased estimator because in general we have

$$E(cc) = E(\frac{\sum cc}{n}) \neq \frac{E(\sum cc)}{E(n)} \tag{27}$$

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

$$E(cc) = E(\frac{\sum cc}{n}) = \frac{E(\sum cc)}{n} \tag{28}$$

The difference is that n in (26) is a random variable thus needs to be estimated whereas the n in (28) is a known constant. The results given by these two estimators (average of $1,000$ tests) are shown as Figure 7. 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 that more tests for the unbiased estimator is needed to get closer to the true value.

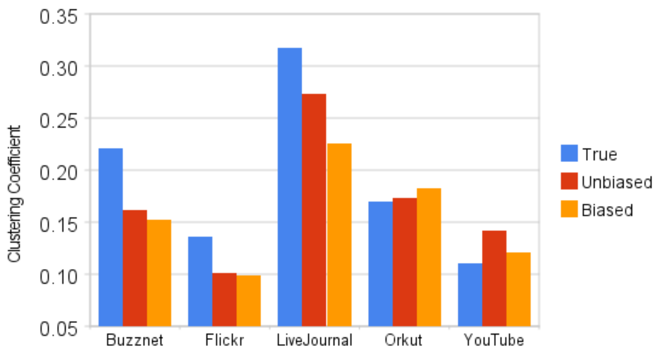


Figure 7 Estimating the clustering coefficient of five OSNs with RW. Left bars represent the true value, middle bars represent the unbiased estimation given by (28) and right bars represent the biased estimation given by (26).

7 Pitfalls

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

7.1 Combining MLE with MHRW

Gjoka *et al.* (Gjoka, Kurant, Butts & Markopoulou 2010) 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.

7.2 Large variance and expensive back tracing

There are two problems when we apply RW to large OSNs. First of all, the variance is large. For example, among $30,000$ tests on Buzznet, only 24% of them 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 some cases, more than 90% of the graph are crawled before v_0 is reached. On average, each test on the Buzznet graph crawls $254,344$ nodes (60% of the entire graph). This number looks scary but it is skewed by a small portion of random walkers which cover a majority of the graph.

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 likely to cover a large portion of the graph as well, as shown by Ye *et al.* (Ye, Lang & Wu 2010).

To finish the back tracing quickly, parallel random walkers can be used while careful coordination needs to be implemented. To avoid duplicated crawlings, we need to cache visited nodes across multiple random walkers. 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).

7.3 Poor lower and upper bounds for RW

Marchetti-Spaccamela (Marchetti-Spaccamela 1989) 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 $k = 1\text{M}$, 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 (Marchetti-Spaccamela 1989) which simply assumes that G is a tree instead of a graph. This estimator, however, greatly overestimates the size when G is large. On our graphs the upper bound estimator overestimates by several magnitudes.

7.4 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 when we deal with LCCs. To make the array as small as possible, we reassign the IDs for the nodes in LCCs such that they are continuous.

- *Accelerating frequently executed routines*: As a lot of similar operations are performed multiple times, making such components faster 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* in the C++ Standard Template Library (STL). 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. Being 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.

  Another example is that we 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.

- *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. Although the random walker makes it impossible to predict the precise access order, our experimental results show that this simple optimization reduces a lot of cache misses.

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

8 Related work

The *coupon collector's problem* (Feller 1957) asks the following question: Given n coupons, from which coupons are randomly sampled with replacement, what is the probability that s sample trials are needed to collect all n coupons? In our MLE problem, n is the quantity of interest while in coupon collector's problem, n is known and the number of unique coupons which have been collected (k) is the quantity of interest.

Knuth (Knuth 1975) proposed an unbiased estimator to estimate the size of a tree with random walkers. Pitt (Pitt 1987) 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 5.1.

Besides estimating the size of graphs, there are studies on estimating other graph properties but few of them focuses on OSNs. Tsonis *et al.* (Tsonis, Swanson & Wang 2008) proposed an estimator for clustering coefficient in scale free networks. Magnien *et al.* (Magnien, Latapy & Habib 2009) developed a fast algorithm to find tight bounds for diameters. Becchetti *et al.* (Becchetti, Boldi, Castillo & Gionis 2008) proposed two approximation algorithms to count the number of triangles incident to every node in a graph. Buchsbaum *et al.* (Buchsbaum, Giancarlo & Racz 2008) considered the problem of finding common neighbors between two nodes. It would be interesting to see how these estimators work on OSNs.

9 Conclusions

In this paper we introduce three existing estimators to estimate the size of OSNs. The first estimator, MLE, relies on uniform sampling and gives accurate estimations 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 employ the second estimator, MR, to give a better estimation of the number of users behind Twitter's public timeline service. We extend the third 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. 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 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.

References and Notes

Becchetti, L., Boldi, P., Castillo, C. & Gionis, A. (2008), 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', pp. 16–24.

Buchsbaum, A. L., Giancarlo, R. & Racz, B. (2008), 'New results for finding common neighborhoods in massive graphs in the data stream model', *Theoretical computer science* **407**(1-3), 302–309.

Dhakar, T. & Mattheiss, T. (1989), 'Determining the size of a finite population of different objects from a finite sample taken at random with replacement', *Communications in statistics - simulation and computation* **18**, 1311–1323.

Driml, M. & Ullrich, M. (1967), 'Maximum likelihood estimate of the number of types', *Acta Technica ČSAV* pp. 300–303.

Feller, W. (1957), *Introduction to probability and its applications*, Vol. 1, second edn, John Wiley.

Finkelstein, M., Tucker, H. G. & Veeh, J. A. (1998), 'Confidence intervals for the number of unseen types', *Statistics and probability letters* **37**(4), 423–430.

Gjoka, M., Kurant, M., Butts, C. T. & Markopoulou, A. (2010), Walking in Facebook: A case study of unbiased sampling of OSNs, *in* 'Proceedings of the 2010 IEEE Infocom conference', pp. 1–9.

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

Knuth, D. E. (1975), 'Estimating the efficiency of backtrack programs', *Mathematics of computation* **29**(129), 121–136.

Magnien, C., Latapy, M. & Habib, M. (2009), 'Fast computation of empirically tight bounds for the diameter of massive graphs', *Journals of experimental algorithmics* **13**, 1.10–1.9.

Marchetti-Spaccamela, A. (1989), 'On the estimate of the size of a directed graph', *Graph-theoretic concepts in computer science* **344**, 317–326.

Miller, C. C. (2010), 'Why twitters c.e.o. demoted himself'. `http://www.nytimes.com/2010/10/31/technology/31ev.html`.

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

Moore, R. J. (2010), 'New data on Twitter's users and engagement'. `http://themetricsystem.rjmetrics.com/2010/01/26/new-data-on-twitters-users-and-engagement/`.

Pitt, L. (1987), 'A note on extending Knuth's tree estimator to directed acyclic graphs', *Information processing letters* **24**(3), 203–206.

Seber, G. (1982), *The Estimation of Animal Abundance*, The Blackburn Press.

Tsonis, A., Swanson, K. & Wang, G. (2008), 'Estimating the clustering coefficient in scale-free networks on lattices with local spatial correlation structure.', *Physica A: Statistical mechanics and its applications* **387**, 5287–5294.

Watts, D. J. & Strogatz, S. H. (1998), 'Collective dynamics of 'small-world' networks', *Nature* **393**, 440–442.

Ye, S., Lang, J. & Wu, F. (2010), Crawling online social graphs, *in* 'APWeb '08: Proceeding of the 2010 Asia Pacific Web conference', pp. 236–242.