Counting Edges and Triangles in Online Social Networks via Random Walk

Counting Edges and Triangles in Online Social Networks via Random Walk

Yang Wu[1], Cheng Long[2], Ada Wai-Chee Fu[1], and Zitong Chen[1]

[1] The Chinese University of Hong Kong
{yangwu,adafu, ztchen}@cse.cuhk.edu.hk
[2] Queen's University Belfast
cheng.long@qub.ac.uk

Abstract. Online social network (OSN) analysis has attracted much attention in recent years. Edge and triangle counts are both fundamental properties in OSNs. However, for many OSNs, one can only access parts of the network using the application programming interfaces (APIs). In such cases, conventional algorithms become infeasible. In this paper, we introduce efficient algorithms for estimating the number of edges and triangles in OSNs based on random walk sampling on OSNs. We also derive a theoretical bound on the sample size and number of APIs calls needed in our algorithms for a probabilistic accuracy guarantee. We ran experiments on several publicly available real-world networks and the results demonstrate the effectiveness of our algorithms.

Keywords: Graph sampling · Random walk · Triangle Counting

1 Introduction

Triangle counting is a fundamental problem in network analysis and triangle structure is widely used in many applications such as spam detection [10], hidden thematic layers uncovery, [11] and community detection [25]. There is a rich related work on enumerating all triangles [12,13,14,15]. However, exhaustive counting is not scalable, since it has to explore every triangle. Even with state-of-art algorithm, enumeration of all triangles has prohibitive cost for real-world large networks.

One alternative way is to use sampling algorithm to speed up the counting process with acceptable error [8,16], but both of these methods need to know the complete data of the network. For example, in [8], the authors proposed to count triangles by wedge sampling. In their method, we have to know the topology of the network in advance, but most OSN's service providers are unwilling to share the complete data for public use.

As noted in [2], for a typical online social network (OSN), the underlying social network may be available only through a public interface, in the form of an application programming interface (API) which may provide a list of a user's neighbors. As in [2] we assume that we can only have external access to the social network via its public interface, and only a fraction of the users/nodes can be sampled. Graph sampling through random walk is widely used in this scenario to estimate graph properties such as degree distribution[4,5,9], clustering coefficient[2], graph size[1,2] and graphlet statistics[6,17]. However, to our knowledge, no one has considered how to get the counting of triangle or even larger graphlets in OSNs without knowing the complete data.

In this paper, we propose a random walk-based method for triangle counting in OSN (online social network), where we have no assumption on the graph, the complete data of the OSNs is not available, and the number of vertices and edges in the graph is not known. Specifically, our method runs a random walk based on a *subgraph relationship graph* (SR graph) of the original graph by using API calls during which some nodes of the SR graph are sampled and some states of an *expanded Markov chain* that could be built on the random walk process are also sampled. Then, based on the sampled nodes, we estimate the number of edges/links in the SR graph (which could be instantiated as the number of edges in the original graph in some case), based on which and also the sampled states, we estimate the number of triangles in the original graph.

Our contributions are summarized as follows:

- We propose a novel random walk based algorithm to estimate the number of edges and triangles in networks with restricted accesses. To the best of our knowledge, we are the first to estimate the number of edges and triangles in such a scenario.
- We derive a theoretical bound on the sample size for achieving an (ϵ, δ)-approximation.
- Experiments are conducted on publicly available real-world networks which verified the effectiveness of our algorithms.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 introduces our problem, background and sampling algorithm. Section 4 introduces our estimators, and Section 5 gives some implementation details and cost analysis. Section 6 reports our experiment results, and Section 7 concludes the paper.

2 Related Work

In graphs with full access, edge counting is a trivial task, but triangle counting has been a hot topic in graph analysis for a long time and a number of algorithms have been proposed to solve this problem. Most of the algorithms can be classified into two categories: exact counting and estimation by sampling.

Exact counting. Enumeration of all triangle counting has a rich history [12,13,14,15]. However, as the graphs become much larger, exhaustive counting is not scalable, since it has to explore every triangle. Recently, many MapReduce based algorithms have been proposed to solving exact triangle counting problem [18,19,20]. In addition, using external memory is also considered as a solution to this problem [21,22,23].

Estimation by sampling. Sampling methods are more related to our work. *Doulion* [16] samples a subgraph from the original graph by keeping each edge with probability p and then estimating triangle counting based on the sampled graph. Another important algorithm is wedge sampling [24]. It samples several wedges from the network as the sample set, and then estimating the number of triangles based on how many triangles exists in the sampled wedge set. However, these sampling methods are based on the facts that the topology of the network is known and we can access the whole network, which are not the assumptions in our scenario.

Most previous works on networks with restricted access are based on random walk methods, such as estimating degree distribution [4,5,9], clustering coefficient [2] and graph size [1,2]. Recently, Chen et al. [17] proposed the state-of-art algorithms for graphlet statistics. To the best of our knowledge, edge or triangle counting has not been studied before in this setting.

3 Problem, Background, and Sampling Algorithm

An OSN can be captured by a simple graph $G = (V, E)$ which is accessible via APIs: given a query vertex u, the API returns all the neighbors of u. We are interested in the problem of estimating two properties of the social network, namely (1) the number of edges in G, $|E|$, and (2) the number of triangles in G, denoted by N. To the best of our knowledge, this is the first attempt for these problems in an online context.

Our solution is based on a random walk on a *subgraph relationship graph* [7,6], from which we build an *expanded Markov chain* [17], and then construct the count estimator based on some sampled states of the expanded Markov chain and the sampled nodes of the random walk.

Our experiments show that our algorithm works best on the subgraph relationship graph when it reduces to the original graph, but since presenting the algorithm based on the original graph requires the same complexity as the general SR graph, we present our algorithm based on a general subgraph relationship graphs in this paper

In the following, we first give some background knowledge about a subgraph relationship graph and an expanded Markov chain process, and then introduce our sampling algorithm.

3.1 Subgraph Relationship Graphs

Given a graph $G = (V, E)$, and an integer $d \geq 1$, the d-node subgraph relationship graph of G (SR graph in short) [7,6], denoted by $G^{(d)} = (H^{(d)}, R^{(d)})$, is defined as follows. First, each node in $H^{(d)}$ corresponds to a connected and induced d-node subgraph in G. For example, each node in $H^{(1)}$ corresponds to a vertex in graph G and each node in $H^{(2)}$ corresponds to an edge in graph G. Second, two nodes in $H^{(d)}$ are connected by an edge in $G^{(d)}$ if and only if they share $d-1$ common vertices of G (exceptionally for $d = 1$, two nodes are connected by an edge if there exists an edge between their corresponding vertices in G), and all these edges constitute $R^{(d)}$. Note that $G^{(1)} = G$. Given a node y in $G^{(d)}$, $N(y)$

denotes the set of nodes adjacent to y (i.e., the neighbors of y) and d_y denotes the degree of y, i.e., $d_y = |N(y)|$. Given two nodes y and y', $\theta(y, y')$ denotes the number of common neighbors of y and y', i.e., $\theta(y, y') = |N(y) \cap N(y')|$.

Example 1. For illustration, consider Figure 1, where a graph G and its corresponding 2-node SR graph $G^{(2)}$ and 3-node SR graph $G^{(3)}$ are shown. Here, each dashed circle corresponds to a node in a SR graph. In $G^{(2)}$, consider the node y containing vertices 1 and 5 and another node y' containing vertices 2 and 5, d_y is 3, $d_{y'}$ is 3, and $\theta(y, y')$ is 1.

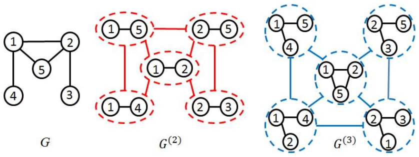

Fig. 1. $G^{(1)}, G^{(2)}$ and $G^{(3)}$

In this paper, we use the subgraph relationship graph $G^{(d)}$ with three settings of d, namely 1, 2, and 3 ($G^{(d)}$ with larger d's is unnecessarily complicated for estimating the counts of edges and triangles since each of them has at most 3 vertices).

3.2 Expanded Markov Chain

A simple random walk process on a graph is to start at a random node, and then randomly pick a neighbor of the current node, go to that node and then repeat the step. It can be regarded as a Markov chain where each node is a state. An *expanded Markov chain* on $G^{(d)}$ [17] based on a random walk remembers *the last l nodes* as the current state, i.e., each possible l consecutive steps/nodes in the random walk process, denoted by $x^{(l)} = (y_1, y_2, ..., y_l)$, is considered as a state of the expanded Markov chain. We use $M^{(l)}$ to denote the state space of the expanded Markov chain. In this paper, depending on the parameter d of $G^{(d)}$, we use different l's. Specifically, for $G^{(1)}$, we use $l = 3$, for $G^{(2)}$, we use $l = 2$, and for $G^{(3)}$, we use $l = 1$. The reason is that each triangle in G is covered in $G^{(1)}$ by 3 nodes, or in $G^{(2)}$ by 2 nodes, or in $G^{(3)}$ by 1 node.

Existing studies show that the expanded Markov chain has a unique *stationary distribution*, denoted by π_M.

Theorem 1. *[17] The stationary distribution π_M exists and is unique. For any $x^{(l)} = (y_1, y_2, ..., y_l) \in M^{(l)}$, we have*

$$\pi_M(x^{(l)}) = \begin{cases} \frac{d_{y_1}}{2|R^{(d)}|} & l{=}1 \\ \frac{1}{2|R^{(d)}|} & l{=}2 \\ \frac{1}{2|R^{(d)}|}\frac{1}{d_{y_2}} \cdots \frac{1}{d_{y_{l-1}}} & l > 2 \end{cases} \qquad (1)$$

where $d_{y_i} (1 \leq i \leq l)$ *is the number of neighbors of* y_i.

3.3 Sampling Algorithm

The idea of our sampling algorithm is very simple, which is to run a random walk on the SR graph $G^{(d)}$ for a sufficient number of steps (such that the stationary distribution is attained) and then collect m nodes, denoted by $y_1, ..., y_m$.

Next, in Section 4, we introduce two estimators for $|E|$ and N based on the sampled nodes, and in Section 5, we give some implementation details of this algorithm and also the time and space cost analysis.

4 Estimators of $|E|$ and N

We introduce an estimator of the number of links/edges in $G^{(d)}$, i.e., $|R^{(d)}|$, in Section 4.1 (note that this estimator corresponds to one for the number of edges in G, i.e., $|E|$, when $d = 1$), and then based on this estimator, we design another estimator for the number of triangles N, in Section 4.2. We provide some accuracy results of these estimators in Section 4.3.

4.1 Estimator of $|R^{(d)}|$

Let Y and Y' be two *independent* nodes sampled from the stationary distribution. First, we define a random variable Φ as the degree of Y (or Y'), and a variable Ψ. That is,

$$\Phi = d_Y; \qquad \Psi = \tfrac{\theta(Y,Y')}{d_Y \cdot d_{Y'}} \tag{2}$$

Then, we have

$$E[\Phi] = E[d_Y] = \textstyle\sum_{y \in H^{(d)}} d_y \cdot \frac{d_y}{2|R^{(d)}|} = \sum_{y \in H^{(d)}} \frac{d_y^2}{2|R^{(d)}|} \tag{3}$$

$$E[\Psi] = E\left[\tfrac{\theta(Y,Y')}{d_Y \cdot d_{Y'}}\right] = \textstyle\sum_{y \in H^{(d)}} \sum_{y' \in H^{(d)}} \frac{d_y}{2|R^{(d)}|} \cdot \frac{d_{y'}}{2|R^{(d)}|} \cdot \frac{\theta(y,y')}{d_y \cdot d_{y'}} \tag{4}$$

$$= \textstyle\sum_{y \in H^{(d)}} \frac{d_y^2}{4|R^{(d)}|^2} \tag{5}$$

The deduction within Equation (4) is based on the assumption that Y and Y' are independent and the deduction from Equation (4) to Equation (5) is based on the fact that each node y contributes d_y^2 times as a common neighbor for two nodes.

Equation (3) and Equation (5) imply the following.

$$|R^{(d)}| = \tfrac{E[\Phi]}{2 \cdot E[\Psi]} \tag{6}$$

Thus, in order to get an estimator of $|R^{(d)}|$, we need estimators of $E[\Phi]$ and $E[\Psi]$. Since Φ and Ψ are based on one and two random nodes from $H^{(d)}$, respectively (see Equation (2)), we design the estimators of $E[\Phi]$ and $E[\Psi]$, denoted by $\widehat{E[\Phi]}$ and $\widehat{E[\Psi]}$, as the average based on the m sampled nodes y_i $(1 \leq i \leq m)$ and the average based on a set Q of n pairs of sampled nodes (y_r, y_t) $(1 \leq r < t \leq m)$, respectively, where Q consists of those pairs of two sampled nodes y_r and y_t from the m sampled nodes such that y_r and y_t correspond to two nodes that are sampled at two steps far away from each other by a certain number f of steps in the random walk process, so we have $Q=\{(y_r, y_t)|\ t-r \geq f \wedge 1 \leq r < t \leq m\}$ and $|Q| = n$ (in this way, the two sampled nodes could be regarded as approximately independent sampled nodes according to the existing study [2], and in our experiments, following [2], we set f as $2.5\%m$).

$$\widehat{E[\Phi]} = \frac{1}{m} \cdot \sum_{i=1}^{m} d_{y_i}; \qquad \widehat{E[\Psi]} = \frac{1}{n} \cdot \sum_{(y_r,y_t)\in Q} \frac{\theta(y_r,y_t)}{d_{y_r} \cdot d_{y_t}} \tag{7}$$

Then, we design the estimator of $|R^{(d)}|$, denoted by $\widehat{|R^{(d)}|}$, based on Equation (6) as follows.

$$\widehat{|R^{(d)}|} = \widehat{E[\Phi]}/(2 \cdot \widehat{E[\Psi]}) \tag{8}$$

Note that $\widehat{|R^{(1)}|}$ corresponds to an estimator of the number of edges in G.

4.2 Estimator of N

Let $x^{(l)}$ be a state in the expanded Markov Chain that involve three vertices in G. We define an indicator function $h(x^{(l)})$ such that $h(x^{(l)}) = 1$ if the three vertices form a triangle and $h(x^{(l)}) = 0$ otherwise.

We note that a certain number of states in $M^{(l)}$ based on $G^{(d)}$ correspond to the same triangle in G. For example, for the case of $G^{(1)}$ and $M^{(3)}$, for a triangle consisting of vertices v_1, v_2 and v_3, 6 states, namely $x_1^{(3)} = (v_1, v_2, v_3)$, $x_2^{(3)} = (v_1, v_3, v_2)$, $x_3^{(3)} = (v_2, v_1, v_3)$, $x_4^{(3)} = (v_2, v_3, v_1)$, $x_5^{(3)} = (v_3, v_1, v_2)$, and $x_6^{(3)} = (v_3, v_2, v_1)$, correspond to it. Similarly, it could be verified that 6 states for the case of $G^{(2)}$ and $M^{(2)}$ and 1 state for the case of $G^{(3)}$ and $M^{(1)}$ correspond to a triangle in G. Let α denote the number of states that correspond to the same triangle. We then know that the number of triangles is equal to the total number of states in which the vertices involved form a triangle divided by α.

$$N = \frac{1}{\alpha} \sum_{x^{(l)}\in M^{(l)}} h(x^{(l)}) \tag{9}$$

Let $X^{(l)}$ be random state with the stationary distribution π_M. Based on Equation (9), we have the following.

$$N = \frac{1}{\alpha} \sum_{x^{(l)}\in M^{(l)}} h(x^{(l)}) = \frac{1}{\alpha} \sum_{x^{(l)}\in M^{(l)}} \frac{h(x^{(l)})}{\pi_M(x^{(l)})} \pi_M(x^{(l)}) \tag{10}$$

$$= \frac{1}{\alpha} E\left[\frac{h(X^{(l)})}{\pi_M(X^{(l)})}\right] = \frac{|R^{(d)}|}{\alpha} E\left[\frac{h(X^{(l)})}{\pi_M(X^{(l)}) \cdot |R^{(d)}|}\right] \tag{11}$$

Here, the deduction from Equation (10) to Equation (11) is based on the fact that $X^{(l)}$ (and also $h(X^{(l)})$) follows the π_M distribution. We define a random variable T as follows.

$$T = \frac{h(X^{(l)})}{\pi_M(X^{(l)}) \cdot |R^{(d)}|} \tag{12}$$

Thus, we have

$$N = \frac{|R^{(d)}|}{\alpha} E\left[\frac{h(X^{(l)})}{\pi_M(X^{(l)}) \cdot |R^{(d)}|}\right] = \frac{|R^{(d)}|}{\alpha} E[T] \tag{13}$$

We note that based on a sampled state for $X^{(l)}$, T could be easily computed since $|R^{(d)}|$ would be canceled out in $\pi_M(X^{(l)}) \cdot |R^{(d)}|$ according to Equation (1).

Based on Equation 13, we know that in order to estimate N, we need an estimator of $E[T]$. Since T is based on one random state from $M^{(l)}$ with distribution π_M(See Equation (12)), we design the estimator of $E[T]$, denoted by $\widehat{E[T]}$, as the average based on k states $x_i^{(l)}$ $(1 \leq i \leq k)$ of the expanded Markov chain built on the random walk process, where $k = m - l + 1$ and $x_i^{(l)} = (y_i, y_{i+1}, ..., y_{i+l-1})$.

$$\widehat{E[T]} = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{h(x_i^{(l)})}{\pi_M(x_i^{(l)}) \cdot |R^{(d)}|} \tag{14}$$

Then, we design the estimator of N, denoted by $\widehat{N}$, based on Equation (13) and (8)as follows.

$$\widehat{N} = \frac{\widehat{E[\Phi]}\widehat{E[T]}}{2\alpha\widehat{E[\Psi]}} \tag{15}$$

4.3 Accuracy Analysis

We did some analysis of the accuracies of $\widehat{|R^{(d)}|}$ and $\widehat{N}$ based on the settings of k, m and n, and the results are shown in the following theorem.

Theorem 2. *For any $\epsilon \leq 1/2$ and $\delta \leq 1$ we have*

$$Pr[N(1-\epsilon) \leq \widehat{N} \leq N(1+\epsilon)] \geq 1 - \delta \tag{16}$$

$$\begin{aligned} \textit{if } k &\geq \frac{48 \sum_{x^{(l)} \in M^{(l)}} \frac{h(x^{(l)})}{\pi_M(x^{(l)})} - \alpha^2 N^2}{\delta \epsilon^2 \alpha^2 N^2}, \\ m &\geq \frac{48[\sum_{y \in H^{(d)}} \frac{d_y^3}{2|R^{(d)}|} - (\sum_{y \in H^{(d)}} \frac{d_y^2}{2|R^{(d)}|})^2]}{\delta \epsilon^2 (\sum_{y \in H^{(d)}} \frac{d_y^2}{2|R^{(d)}|})^2}, \textit{ and} \\ n &\geq \max\left\{ \frac{384|R^{(d)}|^2}{\epsilon^2 \delta \sum_{y \in H^{(d)}} d_y^2}, \left[\frac{384\sqrt{2}|R^{(d)}| \cdot |H^{(d)}|}{\epsilon^2 \delta \sum_{y \in H^{(d)}} d_y^2}\right]^2 \right\} \end{aligned} \tag{17}$$

Proof. Please see Appendix A. □

The analysis of $\widehat{|R^{(d)}|}$ can be found in our technical report [26].

5 Implementation Details and Cost Analysis

With the algorithms introduce in the previous section, we now explain some details in our implementation, along with the time and space analysis.

First, we explain how to implement a random walk process based on $G^{(d)}$ during which m random nodes are collected. The core of the random walk process is the transition procedure which is to pick one neighbor of the current node $y = (v_1, ..., v_d)$ randomly. In the following, we explain how to compute the set of neighbors of y, based on which the transition procedure could be done easily. We consider three cases. Case 1, $d = 1$. In this case, the problem is easy and could be solved by invoking an API call based on y which corresponds to a vertex. Case 2, $d = 2$. In this case, $y = (v_1, v_2)$. The set of neighbors of y in $G^{(2)}$ corresponds to $\{(u,v) \mid u = v_1 \wedge v \in N(v_1)\backslash v_2\} \bigcup \{(u,v) \mid u = v_2 \wedge v \in N(v_2)\backslash v_1\}$, and thus it could be computed based on $N(v_1)$ and $N(v_2)$ which could be obtained by invoking two API calls, one with v_1 as input and the other with v_2 as input. Case 3, $d = 3$. In this case, $y = (v_1, v_2, v_3)$. We define $M(v_i,v_j) = N(v_i) \bigcup N(v_j)\backslash V(y)$, if $(v_i,v_j) \in E$ and $M(v_i,v_j) = N(v_i) \bigcap N(v_j)\backslash V(y)$, otherwise, where $V(y)$ denotes the set of vertices $\{v_1, v_2, ..., v_d\}$. Then the set of neighbors of y in $G^{(3)}$ corresponds to $\{(u, v, w) \mid u = v_1 \wedge v = v_2 \wedge w \in M(v_1,v_2)\} \bigcup \{(u, v, w) \mid u = v_2 \wedge v = v_3 \wedge w \in M(v_2,v_3)\} \bigcup \{(u, v, w) \mid u = v_1 \wedge v = v_3 \wedge w \in M(v_1,v_3)\}$, and thus it could be computed based on $N(v_1)$, $N(v_2)$ and $N(v_3)$ which could obtained by invoking three API calls.

Note that with the above implementation, each transition requires d API calls. Fortunately, since two nodes that are visited *consecutively* during the process are neighbors to each other (i.e., they share $d-1$ vertices), the results of the previous API calls could be saved for the current node and thus exactly one new API call is needed. Therefore, each transition requires one API call on average.

We regard one API call as a unit of time and we exclude the time cost before the mixing time in the random walk process. Then the complexities of our algorithms are as follows, the proofs can be found in [26]

Theorem 3. *Let d_{max} be the maximum degree of $G^{(d)}$. The time complexity of the random walk process is $O(k)$ for $G^{(1)}$ and $O(k \cdot d_{max})$ for $G^{(2)}$ and $G^{(3)}$. The time complexities of computing $\widehat{R^{(d)}}$ and $\widehat{N}$ are $O(k \cdot d_{max})$. The space complexities for both processes are $O(k \cdot d_{max})$.*

6 Experiments

In this section, we report on our experimental findings. First we describe our experimental setup. We used 8 real datasets as shown in Table 1, which are used in the literature for estimating graphlet statistics and network size of OSNs [2,17]. Following these studies, we simulate the scenario where we only have accesses to the datasets via APIs. In each network, we remove the directions of edges, self-loops and multi-edges. We used the largest connected component for each network (since the method could be similarly run on other connected components). The statistics of the largest connected components of networks are shown in Table 1 where N is the number of triangles and W is number of wedges.

We adopt the following two metrics in our experiments.

Table 1. Statistics of Datasets

Network	$\|V\|$	$\|E\|$	N	W
Facebook [28]	6.34×10^4	8.17×10^5	3.5×10^6	7.1×10^7
Epinion [27]	7.6×10^4	4.06×10^5	1.6×10^6	7.4×10^7
Slashdot [27]	7.7×10^4	4.69×10^5	5.5×10^5	6.8×10^7
Gowalla [28]	1.97×10^5	9.5×10^5	2.3×10^6	2.9×10^8
Pokec [28]	1.6×10^6	2.23×10^7	3.26×10^7	2.08×10^9
Flickr [28]	2.2×10^6	2.28×10^7	8.38×10^8	2.33×10^{10}
Orkut [28]	3.08×10^6	1.17×10^8	6.28×10^8	4.56×10^{10}
Live Journal [28]	4.8×10^6	4.28×10^7	2.86×10^8	7.27×10^9

- *Normalized root mean square error* (NRMSE): NRMSE is defined as

$$\text{NRMSE}(\widehat{N}) = \frac{\sqrt{\mathbb{E}[(\widehat{N}-N)^2]}}{N} = \frac{\sqrt{Var[\widehat{N}]+(N-\mathbb{E}[\widehat{N}])^2}}{N}, \quad (18)$$

  Note that NRMSE captures both the variance and the bias of the estimator.
- *Confidence interval*: A $[p_1, p_2]$-confidence interval in our case corresponds to an interval $[L, U]$ such that $Pr[\widehat{N}/N \leq L] = p_1$ and $Pr[\widehat{N}/N \leq U] = p_2$. In our experiments, for each result pair, we run 200 simulations independently and get an estimate of the [5%, 95%]-confidence interval, $[L, U]$, such that L and U are the 5^{th} and 95^{th} percentile values, respectively.

We study the performance of three algorithms each corresponding to our sampling algorithm based on one of the following settings: (1) $d = 1, l = 3$, (2) $d = 2, l = 2$, and (3) $d = 3, l = 1$, the algorithms are denoted by SRWd, depending on the d value for the algorithm. Note that to the best of our knowledge, the problem of estimating the number of edges and triangles in an OSN has not been studied before, and thus we have no state-of-the-arts to compare with in our experiments.

All algorithms are implemented in C++, and we run experiments on a Linux machine with Intel 3.40GHz CPU.

6.1 Performance Studies for Estimating N

Fig. 2 shows the NRMSE results, where the x-axis is the number of random walk steps as a percentage of the set of nodes in the network, and the y-axis is the NRMSE. Each NRMSE value is calculated by averaging over 200 independent simulations. We summarize our results as follows.

- On all networks, SRW1 always achieves the lowest NRMSE among SRW1, SRW2 and SRW3, which means that SRW1 has the highest accuracy in estimating the number of triangles.
- SRW1, the best method in our framework, gives relatively accurate estimation. For example, when 2% of the nodes are sampled, the NRMSE of SRW1 is in the range [0.033, 0.23]. Besides, for most of the networks (6 out of 8 networks), the NRMSE of SRW1 is just around or less than 0.1.

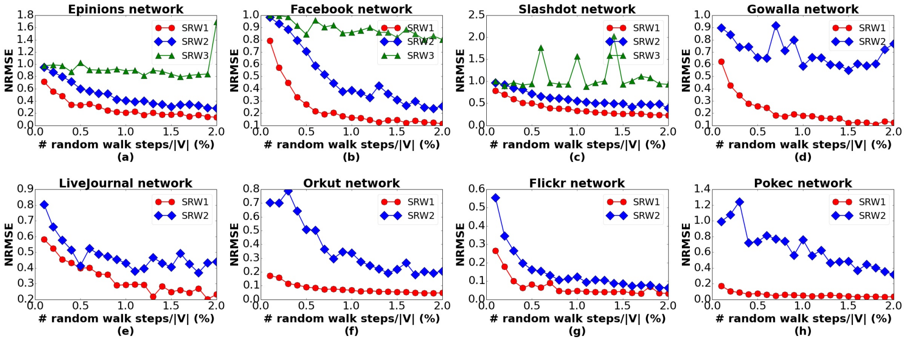

Fig. 2. NRMSE of triangle count estimation

- SRW3 is the worst method in terms of accuracy. It is slow and cannot finish running within a reasonable time on large networks, and as a result, its measurements on some datasets are not available.

Table 2 shows the [5%, 95%]-confidence interval of SRW1 and SRW2 when only 2% of the nodes are sampled. The confidence interval of SRW1 is much tighter than SRW2, which again demonstrates that SRW1 outperforms SRW2 significantly. This result is also consistent with the theoretical analysis in Section 4.3.

Table 2. [5%, 95%]-confidence interval of triangle count estimation

Network	SRW1	SRW2	Network	SRW1	SRW2
Facebook	[0.818, 1.180]	[0.612, 1.521]	Epinion	[0.764, 1.244]	[0.518, 1.438]
Slashdot	[0.629, 1.400]	[0.324, 1.788]	Gowalla	[0.823, 1.218]	[0.200, 2.187]
Pokec	[0.941, 1.066]	[0.669, 1.600]	Flickr	[0.944, 1.053]	[0.878, 1.119]
Orkut	[0.923, 1.083]	[0.754, 1.357]	Live Journal	[0.629, 1.353]	[0.496, 1.741]

Table 3 shows the running time of SRW1 and SRW2 when only 2% of the nodes are sampled. The running time of SRW1 is much smaller than SRW2, which demonstrates that SRW1 is much more efficient than SRW2. In addition, our algorithm SRW1 only takes quite short time to process these networks in experiments, which means our algorithm is very piratical. Fig. 3 shows the running time of SRW1 and SRW2 with different sample size. We find that For both algorithms, the running time is nearly linear with the sample size.

6.2 Performance Studies for Estimating $|E|$

Fig. 4 shows the NRMSE results for edge counts. Based on the experiments results, our estimator of the number of edges is quite accurate. For example,

Table 3. Running time when 2% of the total nodes are sampled

Network	SRW1	SRW2	Network	SRW1	SRW2
Facebook	0.0632s	18.90s	Epinion	0.1134s	19.00s
Slashdot	0.1152s	17.77s	Gowalla	0.44s	66.90s
Pokec	2.43s	177.70s	Flickr	39.81s	226.22s
Orkut	13.15s	637.54s	Live Journal	13.05s	867.28s

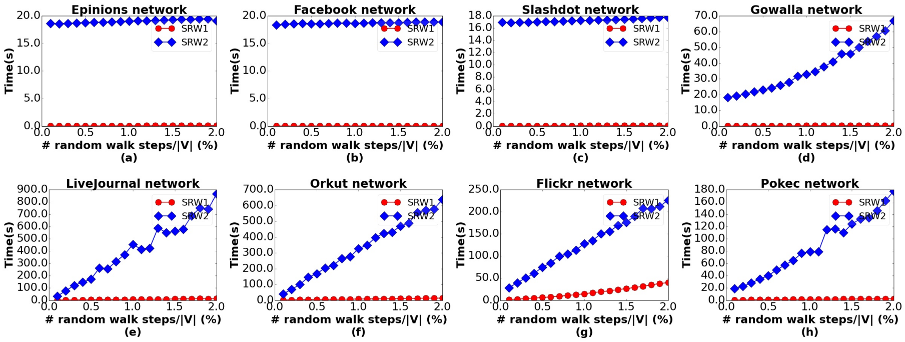

Fig. 3. Running time of triangle count estimation

when 2% of the nodes are sampled, the NRMSE of SRW1 is in the range [0.015, 0.14] and the NRMSE is below 0.1 in 7 out of 8 networks. More details could be found in the full version of the paper.

Remark: Our algorithm is designed for graphs with restricted access. To our knowledge, there is no published work on edge and triangle counting that we can compare with directly. A most relevant work is PSRW [6], which estimates certain graphlet statistics, but does not estimate the counts of edges or triangles. For triangles, they consider only random walks on $G^{(2)}$. With our proposed method involving the estimation of the graph size of $|R^{(d)}|$, we can convert PSRW to compute the counts, in which case it becomes SRW2. However, we have shown that our new method SRW1 outperforms SRW2 significantly in Section 6.

7 Conclusion

In this paper, we present efficient random walk-based algorithms to estimate the number of triangles and the number of edges in OSNs with restricted access. We derive a theoretical bound on the number of samples needed for an accuracy guarantee. Our experiments on real-world OSNs showed that our algorithms provide accurate estimations.

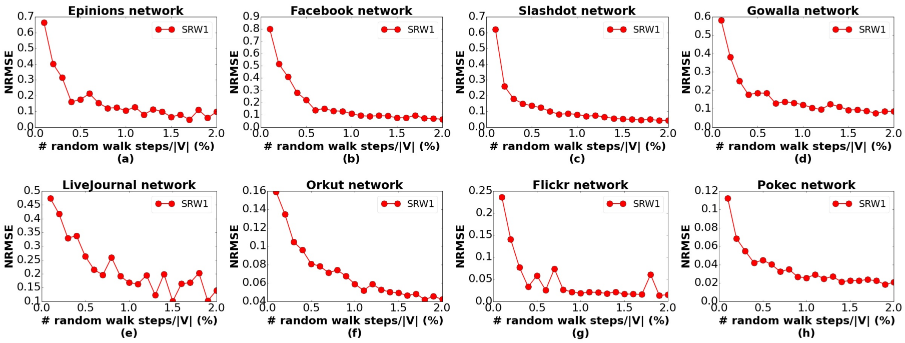

Fig. 4. NRMSE of edge count estimation

References

1. Katzir, L., and Liberty, E., and Somekh, O.: Estimating sizes of social networks via biased sampling. In WWW, pp. 597-606 (2011)
2. Hardiman, S.J., and Katzir, L.: Estimating clustering coefficients and size of social networks via random walk. In WWW, pp. 539-550 (2013)
3. Jha, M., and Seshadhri, C., and Pinar, A.: Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In WWW, pp. 495-505 (2015)
4. Lee, C.-H., and Xu, X., and Eun, D.Y.: Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. In SIGMETRICS, pp. 319-330 (2012)
5. Li, R.-H., and Yu, J., and Qin, L., and Mao, R., and Jin, T.: On random walk based graph sampling. In ICDE, pp. 927-938 (2015)
6. Wang, P., and Lui, J.C.S., and Ribeiro, B., and Towsley D., and Zhao, J., and Guan, X.: Efficiently estimating motif statistics of large networks. In ICDE, pp. 8:1-8:27 (2014)
7. Bhuiyan, M., and Rahman, M., and Al Hasan, M.: Guise: Uniform sampling of graphlets for large graph analysis. In ICDM, pp. 91-100 (2012)
8. Seshadhri, C., and Pinar, A., and KoldaWedge, T.G.: Sampling for computing clustering coefficients and triangle counts on large graphs. Statistical Analysis and Data Mining, vol. 7, no. 4, pp. 233-235 (2014)
9. Gjoka, M., and Kurant, M., and Butts, C.T.: Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In INFOCOM, pp. 1-9 (2010)
10. Becchetti, L., and Boldi, P., and Castillo, C. and Gionis, A.: Efficient semi-streaming algorithms for local triangle counting in massive graphs. In KDD, pp. 16-24 (2008)
11. Eckmann, J.-P., and Moses, E.: Curvature of co-links uncovers hidden thematic layers in the World Wide Web. In PNAS, pp. 5825-5829 (2002)
12. Chiba, N. and Nishizeki, T.: Arboricity and subgraph listing algorithms. In SIAM J. Computing, pp. 210-223 (1985)
13. Berry, J.W., and Fosvedt, L., and Nordman, D., and Phillips, C.A., and Wilson, A.G.: Listing triangles in expected linear time on power law graphs with exponent at least 7/3. Tech. Report SAND 2010-4474C (2011)
14. Chu, S., and Cheng, J.: Triangle listing in massive networks and its applications. In KDD, pp. 672-680 (2011)

15. Latapy, M.: Main-memory triangle computations for very large (sparse (power-law)) graphs. Theoretical Computer Science, pp. 458-473 (2008)
16. Tsourakakis, C.E., and Kang, U., and Miller, G.L., and Faloutsos, C.: Doulion: counting triangles in massive graphs with a coin. In KDD, pp. 837-846 (2009)
17. Chen, X., and Li, Y., and Wang, G.P., and Lui, J.C.S.: A general framework for estimating graphlet statistics via random walk. CoRR abs/1603.07504 (2016)
18. Park, H.-M., and Silvestri, F., and Kang, U., and Pagh, R.: MapReduce triangle enumeration with guarantees. ACM Conf. Inf. Knowl. Manage, pp. 1739-1748 (2014)
19. Suri, S., and Vassilvitskii, S.: Counting triangles and the curse of the last reducer. In WWW, pp. 607-614 (2011)
20. Yoon, J.-H., and Kim, S.-R.: Improved sampling for triangle counting with MapReduce. Convergence Hybrid Inform. Technol., vol.6935, pp. 685-689 (2011)
21. Hu, X., and Tao, Y., and Chung, C.-W.: Massive graph triangulation. In SIGMOD, pp. 325-336 (2013)
22. Chu, S., and Cheng, J.: Triangle listing in massive networks. ACM Trans. Knowl. Discovery Data, vol. 6, no. 4, pp. 17 (2012)
23. Pagh, R., and Silvestri, F.: The input/output complexity of triangle enumeration. ACM Symp. Principles Database Syst., pp. 224-233 (2014)
24. Seshadhri, C., and Pinar, A., and Kold, T.G.: Triadic measures on graphs:The power of wedge sampling. In SDM (2013)
25. Berry, J.W., and Hendrickson, B., and LaViolette, R.A., and Phillips, C.A.: Tolerating the community detection resolutionlimit with edge weighting. Phys.Rev.E, vol. 83:056119 (2011)
26. Counting Edges and Triangles in Online Social Networks via Random Walk, https://www.dropbox.com/sh/228dpiup7qgp6mw/AAA4ijK6jsVUosKNz2OHSlUba?dl=0
27. SNAP Datasets: Standford large network dataset collection, http://snap.standford.edu/data
28. KONECT Datasets: The koblenz network collection, http://konect.uni-koblenz.de

Appendix

8 Proof of Theorem 2

Proof. Define three new random variables, $T' = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{h(X_i^{(l)})}{\pi_M(X_i^{(l)}) \cdot |R^{(d)}|}$,$\Psi' = \frac{1}{m} \cdot \sum_{i=1}^{m} d_{Y_i}$, and $\Phi' = \frac{1}{n} \cdot \sum_{(i,j) \in Q} \frac{\theta(Y_i, Y_j)}{d_{Y_i} \cdot d_{Y_j}}$. Here, we don't have a specific sample set from random walk and the sample set itself is also a random variable. This is different from $\widehat{E[T]}$, $\widehat{E[\Phi]}$ and $\widehat{E[\Psi]}$ which are values based on a specific sample set from random walk. It is obvious that $E[T'] = E[T]$, $E[\Psi'] = E[\Psi]$ and $E[\Phi'] = E[\Phi]$. We require that each of the variable T', Φ' and Ψ' is an $\epsilon/4$ approximation of their respective expected values with probability at least 1-$\delta/3$. From Chebyshev's inequality for variable Z, If Z is an $\epsilon/4$ approximation of their respective expected values with probability at least 1-$\delta/3$, then we have,

$$\begin{aligned} Pr(|Z - E[Z]| \geq \epsilon/4 \cdot E[Z]) &\leq \frac{Var[Z]}{(\epsilon/4 \cdot E[Z])^2} \leq \delta/3 \\ Var[Z] &\leq \delta \epsilon^2 (E[Z])^2/48 \end{aligned} \tag{19}$$

Sample size of k. The Variance of T' is:

$$\begin{aligned} Var[T'] = \tfrac{1}{k} \cdot Var\left[\tfrac{h(X^{(l)})}{\pi_M(X^{(l)})\cdot|R^{(d)}|}\right] &= \tfrac{1}{k} \cdot \left\{E\left[\left(\tfrac{h(X^{(l)})}{\pi_M(X^{(l)})\cdot|R^{(d)}|}\right)^2\right] - E^2\left[\tfrac{h(X^{(l)})}{\pi_M(X^{(l)})\cdot|R^{(d)}|}\right]\right\} \\ &= \tfrac{1}{k} \cdot \textstyle\sum_{x^{(l)}\in M^{(l)}} \tfrac{h(x^{(l)})}{\pi_M(x^{(l)})\cdot|R^{(d)}|^2} - \tfrac{\alpha^2 N^2}{|R^{(d)}|^2} \end{aligned} \tag{20}$$

With Equations (19) and (20), we can get the bound of sample size k:

$$k \geq \frac{48 \sum_{x^{(l)}\in M^{(l)}} \frac{h(x^{(l)})}{\pi_M(x^{(l)})} - \alpha^2 N^2}{\delta\epsilon^2\alpha^2N^2} \tag{21}$$

Sample size of m.

$$\begin{aligned} Var[\Phi'] &= \tfrac{1}{m} \cdot Var[d_Y] = \tfrac{1}{m} \cdot (E[d_Y^2] - E^2[d_Y]) \\ &= \tfrac{1}{m} \cdot [\textstyle\sum_{y\in H^{(d)}} \tfrac{d_y^3}{2|R^{(d)}|} - (\sum_{y\in H^{(d)}} \tfrac{d_y^2}{2|R^{(d)}|})^2] \end{aligned} \tag{22}$$

so we have, $m \geq \frac{48[\sum_{y\in H^{(d)}} \frac{d_y^3}{2|R^{(d)}|} - (\sum_{y\in H^{(d)}} \frac{d_y^2}{2|R^{(d)}|})^2]}{\delta\epsilon^2(\sum_{y\in H^{(d)}} \frac{d_y^2}{2|R^{(d)}|})^2}$

Sample size of n

$$Var[\Psi'] = E[\Psi'^2] - E^2[\Psi'] \tag{23}$$

$E^2[\Psi']$ can be obtained from Equation 5. Then, we discuss how to get $E[\Psi'^2]$.

$$\begin{aligned} E[\Psi'^2] &= \tfrac{1}{n^2} E[(\textstyle\sum_{(i,j)\in Q} \theta(Y_i, Y_j) \cdot \tfrac{1}{d_{Y_i}\cdot d_{Y_j}})^2] \\ &= \tfrac{1}{n^2} \textstyle\sum_{(i,j)\in Q} \sum_{(i',j')\in Q} E[\theta(Y_i, Y_j) \cdot \tfrac{1}{d_{Y_i}\cdot d_{Y_j}} \theta(Y_i', Y_j') \cdot \tfrac{1}{d_{Y_i'}\cdot d_{Y_j'}}] \end{aligned} \tag{24}$$

To calculate this summation easily, we divide it into three cases.

(a). The node pairs (Y_i, Y_j) and (Y_i', Y_j') are the same i.e. $i = i'$ and $j = j'$,

$$\begin{aligned} &E[\theta(Y_i, Y_j) \cdot \tfrac{1}{d_{Y_i}\cdot d_{Y_j}} \theta Y_i', Y_j' \cdot \tfrac{1}{d_{Y_i'}\cdot d_{Y_j'}}] = E[(\theta(Y_i, Y_j) \cdot \tfrac{1}{d_{Y_i}\cdot d_{Y_j}})^2] \\ &= \tfrac{1}{4|R^{(d)}|^2} \textstyle\sum_{i\in H^{(d)}} \sum_{j\in H^{(d)}} \tfrac{(\theta(i,j))^2}{d_i\cdot d_j} \end{aligned} \tag{25}$$

Since for general OSN, the degree of every user should be no smaller than 2, we have $d_i \cdot d_j \geq d_i + d_j$. In addition, it is obvious that $d_i + d_j \geq \theta(i,j)$. Based on these results, we have, $\frac{(\theta(i,j))^2}{d_i\cdot d_j} \leq \frac{(\theta(i,j))^2}{d_i+d_j} \leq \theta(i,j)$. So, $E[(\theta(Y_i, Y_j) \cdot \frac{1}{d_{Y_i}\cdot d_{Y_j}})^2] \leq \frac{1}{4|R^{(d)}|^2} \sum_{i\in H^{(d)}} \sum_{j\in H^{(d)}} \theta(i,j) = \frac{1}{4|R^{(d)}|^2} \sum_{i\in H^{(d)}} d_i^2$

Moveover, we have n such cases.

(b). The node pairs (Y_i, Y_j) and (Y_i', Y_j') share no common node.

$$\begin{aligned} &E[\theta(Y_i, Y_j) \cdot \tfrac{1}{d_{Y_i}\cdot d_{Y_j}} \theta(Y_i', Y_j') \cdot \tfrac{1}{d_{Y_i'}\cdot d_{Y_j'}}] \\ &= \tfrac{1}{16|R^{(d)}|^4} \textstyle\sum_{i\in H^{(d)}} \sum_{j\in H^{(d)}} \theta(i,j) \cdot \sum_{i' H^{(d)}} \sum_{j'\in H^{(d)}} \theta(i',j') = \tfrac{1}{16|R^{(d)}|^4} (\sum_{i\in H^{(d)}} d_i^2)^2 \end{aligned}$$

Hence, the number of such case is smaller than $n(n-1)$.

(c). The node pairs (Y_i, Y_j) and (Y_i', Y_j') share one common node.

$$E[\theta(Y_i, Y_j) \cdot \tfrac{1}{d_{Y_i} \cdot d_{Y_j}} \theta(Y_i', Y_j') \cdot \tfrac{1}{d_{Y_i'} \cdot d_{Y_j'}}] = \tfrac{1}{8|R^{(d)}|^3} \textstyle\sum_{i \in H^{(d)}} \sum_{j \in H^{(d)}} \sum_{i' \in H^{(d)}} \tfrac{\theta(i,j) \cdot \theta(j,i')}{d_j}$$

Since $d_{Y_j} \geq \theta(Y_j, Y_i')$, we have,

$$\begin{aligned} & E[\theta(Y_i, Y_j) \cdot \tfrac{1}{d_{Y_i} \cdot d_{Y_j}} \theta(Y_i', Y_j') \cdot \tfrac{1}{d_{Y_i'} \cdot d_{Y_j'}}] \\ & \leq \tfrac{|H^{(d)}|}{8|R^{(d)}|^3} \textstyle\sum_{i \in H^{(d)}} \sum_{j \in H^{(d)}} \theta(i,j) = \tfrac{|H^{(d)}|}{8|R^{(d)}|^3} \sum_{i \in H^{(d)}} d_i^2 \end{aligned} \tag{26}$$

The number of such case is smaller than $2n\sqrt{2n}$. (there are n ways choosing the first pair, and the other pair shares one node with this pair, so there is two possible node to share with).

Then we can compute $Var[\Psi']$,

$$Var[\Psi'] = E[\Psi'^2] - E^2[\Psi'] \tag{27}$$

$$\leq \tfrac{1}{n} \tfrac{1}{4|R^{(d)}|^2} \textstyle\sum_{i \in H^{(d)}} d_i^2 + \tfrac{1}{16|R^{(d)}|^4} (\sum_{i \in H^{(d)}} d_i^2)^2 \tag{28}$$

$$+ \tfrac{2\sqrt{2n}}{n} \tfrac{|H^{(d)}|}{8|R^{(d)}|^3} \textstyle\sum_{i \in H^{(d)}} d_i^2 - \left[\sum_{i \in H^{(d)}} \tfrac{d_i^2}{4|R^{(d)}|^2}\right]^2 \tag{29}$$

$$\leq \tfrac{1}{n} \tfrac{1}{4|R^{(d)}|^2} \textstyle\sum_{i \in H^{(d)}} d_i^2 + \tfrac{2\sqrt{2n}}{n} \tfrac{|H^{(d)}|}{8|R^{(d)}|^3} \sum_{i \in H^{(d)}} d_i^2 \tag{30}$$

Substituting this into Equation (19), and let both of the terms less than half of the right hand size in Equation (19), we got the condition for n:

$$n \geq \max\{\tfrac{384|R^{(d)}|^2}{\epsilon^2 \delta \sum_{i \in H^{(d)}} d_i^2}, \left[\tfrac{384\sqrt{2}|R^{(d)}| \cdot |H^{(d)}|}{\epsilon^2 \delta \sum_{i \in H^{(d)}} d_i^2}\right]^2\} \tag{31}$$

This finishes the proof. □