Counting Edges with Target Labels in Online Social Networks via Random Walk

Yang Wu
Chinese University of Hong Kong
Shatin, Hong Kong, China
yangwu@cse.cuhk.edu.hk

Cheng Long
Queen's University Belfast
Belfast, United Kingdom
cheng.long@qub.ac.uk

Ada Wai-Chee Fu
Chinese University of Hong Kong
Shatin, Hong Kong, China
adafu@cse.cuhk.edu.hk

Zitong Chen
Chinese University of Hong Kong
Shatin, Hong Kong, China
ztchen@cse.cuhk.edu.hk

ABSTRACT

Online social networks (OSNs) embed a rich set of information that could be used in a few fields. Extensive research has been done on estimating graph properties such as counts of wedges and triangles in OSNs. While these graph properties which are defined based on the structural information only are useful at a coarse level, they are not sufficient in applications where fine-grained information is desired. In this paper, we study a problem of estimating a type of graph property, namely the count of edges, refined by the labels of the users, which are usually available in users' profiles, and serves as finer-grained information. Existing solutions for estimating graph properties pay no attention on users' labels and thus they are not suitable for many real world applications. We develop two algorithms for the problem, each of which samples a set of edges or nodes via a random walk process and construct estimators based on the sampled edges or nodes. Theoretical analysis on the accuracy guarantees of our algorithms and extensive experiments based on real datasets verify that our algorithms are superior over baseline algorithms.

1 INTRODUCTION

Online social networks (OSN) is very commonly used in real life and it embeds a rich set of information that would be useful in applications from different fields such as social community, marketing business, political campaigns, etc. People are interested in knowing some information of graph properties such as counts of wedges, triangles, cliques, and k-node structures etc. embedded in OSNs. In the literature, researchers have studied problems of estimating degree distribution[7, 14, 16], clustering coefficient[11], graph size[11, 13] and graphlet statistics[5, 21]. These graph properties are usually defined based on the structural information, e.g., a triangle is a triplet of three nodes which are connected with one another via links.

We notice that graph properties based on the structure information correspond to information at a coarse level only, which may not be sufficient in some applications. For example, if an education institution considers to introduce a new Spanish course in Hong Kong, the most important step is to determine whether there are enough potential users who are likely to take this course in Hong Kong. One simple but efficient way is to estimate the number of links/friendships between a user living in Hong Kong and another user living in Spain in OSNs. The reason is that if a user has Spanish friends, then it is likely that he/she will be interested in learning Spanish. Another example is that estimating the number of links/friendships between a user living in China and another user living in Austria in an OSN is an indicator of how many people from Austria and those from China interact with each other. Such information is very useful for airlines in web marketing and advertising, e.g., it could be used for decision making about whether or not to launch a new flight route between China and Austria. Thus, graph properties refined by some feature/label information of users (which are available in user's profiles in many cases) correspond to finer-grained information and could be highly valuable in real world application.

Motivated by this, we propose to estimate graph properties refined by users' labels. In this paper, we focus on one type of graph properties, namely the number of edges with some target labels. Specifically, given two target labels, we say that an edge is a *target edge* if one node of the edge has one target label and the other node has the other target label. For example, in the example of estimating the number of links between users living in Spain and those living in Hong Kong, "Spain" and "Hong Kong" could be used as two target labels and an edge between a user living in Spain and another user living in Hong Kong corresponds to a target edge. Then, the problem studied in this paper is to estimate the number of target edges for two given target labels.

Existing solutions of estimating graph properties (based on the structural information only) do not pay any attention to users' labels and thus, they could not be used for our problem of estimating graph properties (based on both the structural information and the information of users' labels). Another branch of studies that is related to ours is labeled graph mining, such as graph classification [20], subgraph mining [3, 4, 9] and label prediction [24]. However, these solutions all assume full access to the graph, which is not true when dealing with OSNs as we do in this paper since OSNs are only accessible via provided APIs[11, 13].

To solve the problem of estimating the number of target edges, we develop two algorithms, namely, NeighborSample and NeighborExploration, both of which are based on a random walk on the graph and sample a set of edges or nodes. NeighborSample samples a set of k edges with k iterations, at each iteration it samples one edge by sampling a user via a random walk process and then sampling a neighbor of this user. NeighborExploration samples a set of nodes with k iterations via a random walk process and also explores all neighbors of each sampled node and records the number of target edges incident to the sampled node, if the sampled node involves a target label (with the purpose of sampling target edges with higher probabilities). Then, based on the

sample set, we construct unbiased estimators using some statistical techniques. For each estimator, we conduct some theoretical analysis on the relationship between the number of samples and the corresponding accuracy guarantees.

In summary, our main contributions are as follows. First, we propose to estimate graph properties refined by users' labels in OSNs, which, to the best of our knowledge, is the first attempt to do so. Second, we develop two algorithms for estimating the number of edges with target labels and provide theoretical analysis on the accuracy guarantees. Third, we conducted extensive experiments which verified that the algorithms developed in this paper are superior over baseline algorithms.

This paper is organized as follows. Section 2 reviews the related work. Section 3 provides some preliminaries and the problem definition. Section 4 introduces our proposed algorithms for estimating the number of edges with target labels. Section 5 presents the experiments and Section 6 concludes the paper and gives a few directions for future study.

2 RELATED WORK

In the literature, OSNs with restricted access and labeled graphs with full access are both popular topics, but to our knowledge, there has been no study about labeled graphs with restricted access, which we study in this paper.

A lot of work has been done on online social networks with restricted access and most existing work is based on random walk methods, which have been used for estimating degree distribution [7, 14, 16], clustering coefficients [11] and graph size [11, 13]. In [14], the authors introduced a non-backtracking random walk method, which is more efficient than traditional random walk for estimating degree distribution. In [11], the authors proposed simple but efficient sampling algorithms for estimating the clustering coefficient and the graph size via simple random walk. In addition, node pairs sampling in OSNs has also been studied in [22]. Recently, Chen et al. [5] proposed the state-of-the-art random walk based algorithms for graphlets statistics using the concepts of subgraph relationship graphs and expanded Markov Chain.

Labels in graph are widely used in many applications, which has attracted much attention from researchers and a considerable amount of work has been done on labeled graphs. Particularly, it has been studies in the area of subgraph mining a lot. In [9], the authors studied the problem of mining frequent neighbourhood pattern in labeled graphs and in [4], a method for mining significant connected subgraph in labeled graphs is proposed. Anchuri et al. consider the difference between labels as a cost and introduce an algorithm for mining approximate subgraph patterns with label cost [3]. Labels are also used in graph classification. In [20], the authors proposed an algorithm for classifying labeled nodes based on structural neighbourhood. Recently, Ye et al. [24] considered a new scenario where only very few vertices have labels compared to large amounts of unlabeled vertices, and proposed an algorithm which leverages the limited user information and friendship network wisely to infer the labels of unlabeled users in OSNs.

3 PRELIMINARIES AND PROBLEM DEFINITION

In this section, we first introduce some basic notations, and then give a formal definition of the problem.

An OSN is represented as an undirected graph $G(V, E)$, where a user corresponds to a node in V and a friendship between two users corresponds to an edge in E. For each user u in G, we denote u's degree by $d(u)$, i.e., $d(u)$ corresponds to the number of u's friends. Each user/node in V has a set of labels such as this user's gender, profession, living country etc. which could be found out in most cases by checking users' profile. For an edge (u, v), we define its label as a pair of two labels, one is a label of u and the other is a label of v.

Let t_1 and t_2 be two target labels. We say that an edge (u, v) is a *target edge* if the edge has the pair of t_1 and t_2 as one of its labels, i.e., either u has t_1 and v has t_2, or v has t_1 and u has t_2. We say that (t_1, t_2) is the target edge label. Let F be the number of target edges. In this paper, we study the problem of estimating F with the following assumptions: (1) we have no full access to the graph $G(V, E)$ but only some limited access via APIs each of which can be used to retrieve the list of friends/neighbors of a given user; (2) the information of $|V|$ and $|E|$ is available as prior knowledge (this is reasonable since this information can be often obtained from the OSN owner's reports or Internet, and in case that such information is not publicly available, some existing methods such as [11] and [23] could be used to estimate $|V|$ and $|E|$, respectively).

We also derive the bound on the sample size which can achieve an (ϵ, δ)-approximation estimation of F. The $\widehat{F}$ which satisfies the following equation is an (ϵ, δ)-approximation estimation of F.

$$P[(1-\epsilon)F < \widehat{F} < (1+\epsilon)F] \geq 1-\delta \tag{1}$$

4 ESTIMATORS OF NUMBER OF EDGES WITH TARGET LABEL

We propose to estimate the number of edges with target labels by first sampling a set of edges or nodes and then constructing an (un-biased) estimator based on the set of sampled edges or nodes. In the following, we introduce two algorithms, one for sampling edges and the other for sampling nodes, both of which are based on a random walk on the graph. The first one, as presented in Section 4.1, is called *NeighborSample* and samples a set of k edges with k iterations at each of which it samples one edge by sampling a user via a random walk process and then sampling a neighbor of this user. The second one, as presented in Section 4.2, is called *NeighborExploration* and samples a set of nodes with k iterations via a random walk process and also explores all neighbors of each sampled node and records the number of target edges incident to the sampled node, if the sampled node involves a target label (with the purpose of sampling target edges with higher probabilities).

For each sampling algorithm, we use Hansen-Hurwitz estimator [10], Horvitz-Thompson estimator [12] and Re-weighted estimator [17] to estimate the number of target edges. Hansen-Hurwitz estimator and Horvitz-Thompson estimator are two simple and widely used estimator when samples are sampled with unequal probabilities, and Re-weighted estimator is an estimator based on the Hansen-Hurwitz estimator.

4.1 Estimation Based on NeighborSample

4.1.1 Sampling Process. NeighborSample samples a set S of k edges with k iterations. At each iteration, it samples an edge by sampling a user u via simple random walk first and then randomly picking one of u's neighbors, says v (i.e., (u, v) corresponds to the edge sampled at this iteration). The pseudo-code of NeighborSample is presented in Algorithm 1.

Algorithm 1 NeighborSample

Require: an online social network $G(V, E)$ accessible via APIs
1: $S \leftarrow \emptyset$
2: **for** i: $1 \leftarrow k$ **do**
3: Sample a user u_i via simple random walk
4: Sample a neighbor v_i of u_i randomly
5: $S \leftarrow S \cup (u_i, v_i)$
6: **end for**
7: **Return** S

Based on the sampling process, we construct two different estimators, one is based on the *Hansen-Hurwitz Estimator* and the other is based on the *Horvitz-Thompson Estimator*.

4.1.2 Hansen-Hurwitz Estimator. We define X_i $(1 \le i \le k)$ to be the edge sampled at the i^{th} iteration of the sampling process. Consider the distribution of X_i. First, X_i could be any edge $(u, v) \in E$. Second, an edge (u, v) is sampled if and only if the following two events happen: (E1) u is sampled by the random walk (line 3 in Algorithm 1) and then v, as one of u's neighbors, is picked (line 4 in Algorithm 1) and (E2) v is sampled by the random walk (line 3 in Algorithm 1) and then u, as one of u's neighbors, is picked (line 4 in Algorithm 1). Third, the probability of event E1 is equal to $\frac{d(u)}{2|E|} \cdot \frac{1}{d(u)} = \frac{1}{2|E|}$ (the probability that u is sampled by the random walk is equal to $\frac{d(u)}{2|E|}$ according to the stationary distribution of a random walk [8, 18] and the probability that one specific neighbor of u is sampled is equal to $\frac{1}{d(u)}$) and so is that of event E2. Therefore, the probability that X_i corresponds to any edge (u, v), denoted by $\pi(X_i = (u, v))$, is equal to $\frac{1}{2|E|} + \frac{1}{2|E|} = \frac{2}{2|E|} = \frac{1}{|E|}$, i.e., X_i corresponds to a uniform sample from the set of edges.

Now, consider $\frac{I(X_i)}{\pi(X_i)}$ which is also a random variable, where $I(X_i)$ is an indicator function and $I(X_i) = 1$ if X_i is one target edge, and 0 otherwise. We deduce that $E[\frac{I(X_i)}{\pi(X_i)}] = \sum_{X_i \in E} I(X_i) = F$. Based upon on this, we construct an estimator of F as $\frac{I((u_i, v_i))}{\pi(X_i = (u_i, v_i))} = |E| \cdot I((u_i, v_i))$, which could be verified to be unbiased. Since in each iteration, we can construct such an estimator, we use the average of these estimators as the final estimator, which is presented as follows.

$$\hat{F} = \frac{1}{k} \sum_{1 \le i \le k} |E| \cdot I(X_i) = \frac{1}{k} \sum_{1 \le i \le k} |E| \cdot I((u_i, v_i)) \quad (2)$$

This estimator corresponds to a Hansen-Hurwitz estimator [10].

Implementation. A straightforward implementation of the NeighborSample sampling process as shown in Algorithm 1 would perform k random walk processes. We note that performing a random walk process is costly since it needs to walk for some enough steps (which corresponds to the *mixing time* [6] of the random walk) in order to achieve the stationary distribution, where each walk from one user to one of her neighbors requires to issue an API call. Fortunately, we observe that the above estimator does not require to sample edges (u_i, v_i) $(1 \le i \le k)$ independently, and thus we propose to sample all these edges using a *single* random walk process. Specifically, it performs a enough number of simple random walk steps first (i.e., the mixing time is achieved) and then continues to walk for k steps further, each via an edge. At the end, it picks those edges it walks through at the last k steps as the sampled edges. In this way, k edges are sampled via only a single random process and as could be verified, the probability that one sampled edge corresponds to a specific edge in the graph is still equal to $\frac{1}{|E|}$ and thus the estimator constructed above is still valid.

Analysis. In this part, we derive theoretical results on the number of edges to sample in order to achieve some pre-set accuracy guarantee.

Theorem 4.1. *Let $1 > \delta > 0$, $\epsilon \le 1$ and $k \ge 1$. Sampling k edges in NeighborSample will return an (ϵ, δ)-approximation estimation of F by the Hansen-Hurwitz estimator, if*

$$k \ge \frac{\sum_{X \in E} |E| \cdot I(X) - F^2}{\epsilon^2 \cdot F^2 \cdot \delta}$$

Proof. Since $F = E[\frac{I(X_i)}{\pi(X_i)}]$, if $\frac{1}{k} \sum_{i=1}^{k} \frac{I(X_i)}{\pi(X_i)}$ is an (ϵ, δ)-approximation estimation of $E[\frac{I(X_i)}{\pi(X_i)}]$, then $\widehat{F}$ is also an (ϵ, δ)-approximation estimation of F.

Let $Y = \frac{1}{k} \sum_{i=1}^{k} \frac{I(X_i)}{\pi(X_i)}$, then $E[Y] = F$ and $Var[Y] = \frac{1}{k}(E[\frac{I(X_i)^2}{\pi(X_i)^2}] - E[\frac{I(X_i)}{\pi(X_i)}]^2) = \frac{1}{k} \cdot \sum_{X \in E} \frac{I(X)}{\pi(X)} - F^2$ and $E[Y] = F$. By Chebyshev's inequality (see Appendix A), we obtain the bound of k for achieving (ϵ, δ)-approximation.

$$k \ge \frac{\sum_{X \in E} |E| \cdot I(X) - F^2}{\epsilon^2 \cdot F^2 \cdot \delta} \qquad \square$$

4.1.3 Horvitz-Thompson Estimator. For each edge $e = (u, v) \in E$, we define a new indicator function $H(e \in S)$ such that $H(e \in S)$ is equal to 1 if e is sampled by the NeighborSample sampling process once or multiple times, and 0 otherwise. We define $Pr(e)$ as the probability that an edge e is sampled in at least one iteration of the NeighborSample sampling process, i.e., $e \in S$. Consider $Pr(e)$ for a specific edge e, we observe that the event that e is not sampled happens if and only if all following k events happen: e is not sampled in the i^{th} iteration for $1 \le i \le k$. Considering that the probability that each of these events happens is equal to $(1 - \frac{1}{|E|})$ and these events are independent, we know that the probability that e is not sampled in any of iterations is equal to $(1 - \frac{1}{|E|})^k$, which further implies that the probability that e is sampled in at least one of the iterations, i.e., $Pr(e)$, is equal to $(1 - (1 - \frac{1}{|E|})^k)$.

Then, we construct an estimator of F as follows.

$$\hat{F} = \sum_{e \in E} \frac{I(e)}{Pr(e)} H(e \in S) = \sum_{e \in E} \frac{I(e)}{1 - (1 - \frac{1}{|E|})^k} H(e \in S) \quad (3)$$

Next we show how this estimator is derived and that it is unbiased. Define $\{S_1, S_2, ..., S_m\}$ as the collection of all possible sample sets each of which contains k edges from the edge set E. Let $P(S_i)$ be the probability that we get set S_i as the sample set in NeighborSample process. Let S be a random set from $\{S_1, S_2, ..., S_m\}$, we have

$$\begin{aligned} E[\textstyle\sum_{e \in E} \frac{I(e)}{Pr(e)} H(e \in S)] &= \textstyle\sum_{j=1}^{m} P(S_j) \sum_{e \in E} \frac{I(e)}{Pr(e)} H(e \in S_j) \\ &= \textstyle\sum_{e \in E} \frac{I(e)}{Pr(e)} \sum_{j=1}^{m} P(S_j) H(e \in S_j) \\ &= \textstyle\sum_{e \in E} \frac{I(e)}{Pr(e)} Pr(e) = \sum_{e \in E} I(e) = F \end{aligned} \quad (4)$$

So the estimator $\widehat{F} = \sum_{e \in E} \frac{I(e)}{Pr(e)} H(e \in S)$ is an unbiased estimator.

Implementation. Different from the case of the Hansen-Hurwitz estimator in Section 4.1.2, the Horvits-Thompson estimator requires that the edges sampled in different iterations are independent. With the implementation in Section 4.1.2, the edges sampled are not independent since the edge sampled at the current iteration is adjacent to the one sampled at the last iteration. To meet the independence requirement, we adopt an existing strategy [11], which is to use those vertices (and edges) which are

sampled far away from each other by a certain number r of steps in the random walk process, (in this way, every two sampled edges could be regarded as approximately independent sampled edges, and following [11], we set r as 2.5%k) in our experiments.

Analysis. In this part, we derive some theoretical results on the number of edges to sample in order to achieve some pre-set accuracy guarantee.

Theorem 4.2. *Let $1 > \delta > 0$, $\epsilon \leq 1$ and $k \geq 1$. Sampling k edges in NeighborSample will return an (ϵ, δ)-approximation estimation of F by the Horvitz-Thompson estimator, if*

$$k \geq \max_{e \in E} \log \frac{I(e)^2+B}{B} / \log \frac{1}{A(e)}$$

where $A(e) = 1 - \frac{1}{|E|}$ and $B = \delta\epsilon^2 \cdot F^2/|E|$.

Proof. Let $\pi_e = \frac{1}{|E|}$ be the probability of sampling edge e in one NeighborSample process. In [12], it has been proved that the variance of the Horvitz-Thompson estimator is

$$\begin{aligned} &Var[\widehat{F}] = E[\widehat{F}^2] - E[\widehat{F}]^2 \\ &= \sum_{e_1 \in E} \sum_{e_2 \in E} \frac{I(e_1)I(e_2)}{Pr(e_1)Pr(e_2)} E[H(e_1 \in S)H(e_2 \in S)] \\ &- \sum_{e_1 inE} \sum_{e_2 inE} \frac{I(e_1)I(e_2)}{Pr(e_1)Pr(e_2)} E[H(e_1 \in S)]E[H(e_2 \in S)] \\ &= \sum_{e_1 inE} \sum_{e_2 inE} \frac{I(e_1)I(e_2)}{Pr(e_1)Pr(e_2)} Cov(H(e_1 \in S), H(e_2 \in S)) \end{aligned} \tag{5}$$

Since $Cov(H(e_1 \in S), H(e_2 \in S)) = Pr(e_1, e_2) - Pr(e_1)Pr(e_2)$, where $Pr(e_1, e_2) = Pr(e_1) + Pr(e_2) - (1 - (1 - \pi_{e_1} - \pi_{e_2})^k)$ for $e_1 \neq e_2$ and $Cov(H(e_1 \in S), H(e_2 \in S)) = Pr(e_1)(1 - Pr(e_1))$ for $e_1 = e_2$, so we have

$$\begin{aligned} &Var[\widehat{F}] = \sum_{e_1 \in E} (\frac{1-Pr(e_1)}{Pr(e_1)}) I(e_1)^2 + \\ &\sum_{e_1 \in E} \sum_{e_2 \in E, e_2 \neq e_1} (\frac{Pr(e_1)+Pr(e_2)}{Pr(e_1)Pr(e_2)} + \\ &\frac{-\{1-(1-\pi_{e_1}-\pi_{e_2})^k\} - Pr(e_1)Pr(e_2)}{Pr(e_1)Pr(e_2)}) I(e_1)I(e_2) \end{aligned} \tag{6}$$

The second term in the right hand side of Equation (6) can be simplified as

$$\begin{aligned} &\sum_{e_1 \in E} \sum_{e_2 \in E, e_2 \neq e_1} (\frac{(1-\pi_{e_1}-\pi_{e_2})^k}{Pr(e_1)Pr(e_2)} \\ &- \frac{(1-\pi_{e_1}-\pi_{e_2}+\pi_{e_1}\pi_{e_2})^k}{Pr(e_1)Pr(e_2)}) I(e_1)I(e_2) \end{aligned} \tag{7}$$

since $\pi_{e_1} = \pi_{e_2} = 1/|E| \geq 0$, it is obvious that $1 - \pi_{e_1} - \pi_{e_2} \leq 1 - \pi_{e_1} - \pi_{e_2} + \pi_{e_1}\pi_{e_2}$. As a result, the term 7 is also negative, so we can ignore this term. By Chebyshev's inequality (see Appendix A), we have

$$\sum_{e \in E} (\frac{(1-\pi_e)^k}{1-(1-\pi_e)^k}) I(e)^2 \leq \delta\epsilon^2 \cdot F^2 \tag{8}$$

so if

$$(\frac{(1-\pi_e)^k}{1-(1-\pi_e)^k}) I(e)^2 \leq \delta\epsilon^2 \cdot F^2/|E| \tag{9}$$

holds for each $e \in E$, then Equation (8) also holds.

Define $A(e) = 1 - \pi_e$ and $B = \delta\epsilon^2 \cdot F^2/|E|$, then we obtain the bound of k for achieving (ϵ, δ)-approximation.

$$k \geq \max_{e \in E} \log \frac{I(e)^2+B}{B} / \log \frac{1}{A(e)} \tag{10}$$

□

4.2 Estimation Based on NeighborExploration

4.2.1 Sampling Process. NeighborExploration samples a set S of nodes with k iterations for a given integer k. At each iteration, it first samples a node by a random walk process and then explores *all* edges incident to u if u involves a target label. The rationale of exploring all neighbors of the user u is that once we know that u has a target label, the probability that we can find a target edge incident to u would be relatively high since one of the two target labels has been covered already. The pseudo-code of NeighborExploration is presented in Algorithm 2.

Algorithm 2 NeighborExploration

Require: an online social network $G(V, E)$ accessible via APIs
Require: a pair of two target labels t_1 and t_2
1: $S \leftarrow \emptyset$
2: **for** i: $1 \leftarrow k$ **do**
3: Sample a user u_i via simple random walk
4: **if** u_i has label t_1 or label t_2 **then**
5: Explore all the neighbors of u_i and compute the number of target edges incident to u_i and we use function T to record the mapping from u_i to the number of target edges incident to u_i.
6: **end if**
7: $S \leftarrow S \cup u_i$
8: **end for**
9: **Return** S and function T

4.2.2 Hansen-Hurwitz Estimator. We define a random variable Y_i to be the node sampled at i^{th} iteration of NeighborExploration sampling process. Y_i could be any user $u \in V$ and the probability that Y_i corresponds to a specific user u, denoted by $\pi(Y_i = u)$, is equal to $\frac{d(u)}{2|E|}$ which is based on the stationary distribution of a simple random walk. We define $T(Y_i)$ as the number of target edges incident to Y_i, which also corresponds to a random variable and could be computed when all neighbors of Y_i are explored after we sample Y_i in the random walk process.

Now, consider $\frac{T(Y_i)}{\pi(Y_i)}$ which is also a random variable. We deduce that $E[\frac{T(Y_i)}{\pi(Y_i)}] = \sum_{u \in V} T(u) = 2 \cdot F$. Based upon on this, we construct an estimator of F as $\frac{T(Y_i)}{2 \cdot \pi(Y_i)}$ which could be easily verified to be unbiased. Since in each iteration, we can construct such an estimator, we use the average of these estimators as the final estimator, which is presented as follows.

$$\hat{F} = \frac{1}{k} \sum_{1 \leq i \leq k} \frac{T(Y_i)}{2 \cdot \pi(Y_i)} = \frac{1}{k} \sum_{1 \leq i \leq k} \frac{|E| \cdot T(u_i)}{d(u_i)} \tag{11}$$

This estimator corresponds to a Hansen-Hurwitz estimator [10].

Implementation. Same as the case in Section 4.1.2, the estimator here does not require that the sampled nodes are independent. As a result, it can sample all nodes via a single random process by performing an enough number of simple random walk steps first (i.e., the mixing time is achieved) and then continuing to walk for k steps further. At each of the last k steps, it checks whether the current user u involves a target label. If so, it explores all edges incident to this user, and records $T(u)$.

Analysis. In this part, we derive some theoretical results on the number of nodes to sample in order to achieve some pre-set accuracy guarantee.

Theorem 4.3. *Let $1 > \delta > 0$, $\epsilon \leq 1$ and $k \geq 1$. Sampling k nodes in NeighborExploration will return an (ϵ, δ)-approximation estimation of F by the Hansen-Hurwitz estimator, if*

$$k \geq \frac{\sum_{u \in V} \frac{2|E| \cdot T(u)^2}{d_u} - 4F^2}{4\epsilon^2 \cdot F^2 \cdot \delta}$$

Proof. Since $F = \frac{1}{2}E[\frac{T(Y_i)}{\pi(Y_i)}]$, if $\frac{1}{k}\sum_{i=1}^{k}\frac{T(Y_i)}{\pi(Y_i)}$ is an (ϵ,δ) - approximation estimation of $E[\frac{T(Y_i)}{\pi(Y_i)}]$, then $\widehat{F}$ is also an (ϵ,δ)-approximation estimation of F.

Let $X = \frac{1}{k}\sum_{i=1}^{k}\frac{T(Y_i)}{\pi(Y_i)}$, then $E[X] = 2F$ and $Var[X] = \frac{1}{k}(E[\frac{T(Y)^2}{\pi_Y^2}]$ - $E[\frac{T(Y)}{\pi_Y}]^2) = \frac{1}{k}\cdot\sum_{u\in N}\frac{T(u)^2}{\pi_u} - 4F^2$. By Chebyshev's inequality (see Appendix A), we obtain the bound of k for achieving (ϵ,δ)-approximation.

$$k \geq \frac{\sum_{u\in V}\frac{2|E|\cdot T(u)^2}{d_u} - 4F^2}{4\epsilon^2\cdot F^2\cdot\delta} \tag{12}$$

□

4.2.3 Horvitz-Thompson Estimator. We define an indicator function $H(u \in S)$ such that $H(u \in S)$ is equal to 1 if u is sampled by the NeighborExploration sampling process once or multiple times, and 0 otherwise. S is the sample set obtained from the NeighborExploration sampling process. Then, we define $Pr(u)$ as the probability that a node u is sampled in at least one iteration of the NeighborExploration sampling process, i.e., $u \in S$. Consider $Pr(u)$ for a specific node u. We observe that the event that u is not sampled happens if and only if all following k events happen: u is not sampled in the i^{th} iteration for $1 \leq i \leq k$. Considering that the probability that one of these events happens is equal to $(1 - \frac{d(u)}{2|E|})$, and these events are independent, we know that the probability that u is not sampled in any of iterations is equal to $(1 - \frac{d(u)}{2|E|})^k$, which further implies that the probability that u is sampled in at least one of the iteration, i.e., $Pr(u)$ is equal to $(1 - (1 - \frac{d(u)}{2|E|})^k)$.

Then, we construct a Horvits-Thompson estimator of the number of target edges as follows.

$$\hat{F} = \frac{1}{2}\sum_{u\in V}\frac{T(u)}{Pr(u)}H(u \in S) = \frac{1}{2}\sum_{u\in V}\frac{T(u)}{1-(1-\frac{d(u)}{2|E|})^k}H(u \in S) \tag{13}$$

This estimator could be verified to be unbiased similarly as it is done for the Horvits-Thompson estimator based on NeighborSample in Section 4.1.3.

Implementation. Also, the Horvits-Thompson estimator requires that the nodes sampled in different iterations are independent. With the implementation in Section 4.2.2, the nodes sampled are not independent since the node sampled at the current iteration is adjacent to the one sampled at the last iteration. To meet the independence requirement, we use the same strategy introduced in 4.1.3, which is to use those nodes which are sampled far away from each other by a certain number r of steps in the random walk process, and we set r as $2.5\%k$.

Analysis. In this part, we derive theoretical results on the number of nodes to sample in order to achieve some pre-set accuracy guarantee.

Theorem 4.4. *Let $1 > \delta > 0$, $\epsilon \leq 1$, and $k \geq 1$. Sampling k nodes in NeighborExploration will return an (ϵ,δ)-approximation estimation by the Horvitz-Thompson estimator, if*

$$k \geq \max_{y\in V}\log\frac{T(y)^2+B}{B}/\log\frac{1}{A(y)}$$

where $A(y) = 1 - \pi_y$ and $B = 4\delta\epsilon^2\cdot F^2/|V|$.

Proof. Let $\pi_y = \frac{d_y}{2|E|}$ be the probability of sampling node y in the random walk process. In [12], it has been proved that the variance of the Horvitz-Thompson estimator is

$$\begin{aligned} Var[\widehat{F}] = \frac{1}{4}\{&\sum_{y\in V}(\frac{1-Pr(y)}{Pr(y)})T(y)^2 + \\ &\sum_{y\in V}\sum_{z\in V, z\neq y}(\frac{Pr(y)+Pr(z)}{Pr(y)Pr(z)} + \\ &\frac{-\{1-(1-\pi_y-\pi_z)^k\}-Pr(y)Pr(z)}{Pr(y)Pr(z)})T(y)T(z)\} \end{aligned} \tag{14}$$

The second term in the right hand side of Equation (14) can be simplified as

$$\begin{aligned} &\sum_{y\in V}\sum_{z\in V, z\neq y}(\frac{(1-\pi_y-\pi_z)^k}{Pr(y)Pr(z)} \\ &-\frac{(1-\pi_y-\pi_z+\pi_y\pi_z)^k}{Pr(y)Pr(z)})T(y)T(z) \end{aligned} \tag{15}$$

since $\pi_y = d_y/(2|E|) \geq 0$, it is obvious that $1 - \pi_y - \pi_z \leq 1 - \pi_y - \pi_z + \pi_y\pi_z$. As a result, term (15) is negative, so we can ignore this term. Then By Chebyshev's inequality (see Appendix A), we have

$$\sum_{y\in V}(\frac{(1-\pi_y)^k}{1-(1-\pi_y)^k})T(y)^2 \leq 4\delta\epsilon^2\cdot F^2 \tag{16}$$

so if

$$(\frac{(1-\pi_y)^k}{1-(1-\pi_y)^k})T(y)^2 \leq 4\delta\epsilon^2\cdot F^2/|V| \tag{17}$$

holds for each $y \in V$, then Equation (16) also holds.

Define $A(y) = 1 - \pi_y$ and $B = 4\delta\epsilon^2\cdot F^2/|V|$, we can get the bound of k,

$$k \geq \max_{y\in V}\log\frac{T(y)^2+B}{B}/\log\frac{1}{A(y)} \tag{18}$$

□

4.2.4 Re-Weighted Estimator. Based on the NeighborExploration sampling process, the nodes are sampled with non-uniform probabilities, which is different from the case based on the NeighborExploration sampling process. This makes it possible to construct a *Re-weighted* estimator [17] as follows.

$$\hat{F} = \frac{\sum_{i=1}^{k}T(u_i)/d(u_i)}{2\sum_{i=1}^{k}1/d(u_i)}\cdot|V| \tag{19}$$

Here, u_i corresponds to the user sampled via the random walk process in i^{th} iteration. It was known that the Re-weighted estimator can be interpreted using the *importance sampling* (IS) framework [17]. Specifically, instead of sampling nodes from the target distribution (i.e., the uniform distribution), the IS framework samples edges from a different and easily implemented trial distribution (i.e., the stationary distribution of a random walk process). According to the IS framework, the importance weight of a user u is given by $\frac{1/|V|}{d(u)/2|E|} \propto 1/d(u)$, which meets the definition in Equation (19), where $1/|V|$ corresponds to the probability based on the target distribution and $d(u)/2|E|$ corresponds the probability based on trial distribution.

Implementation. Same as the Hansen-Hurwitz estimator in Section 4.2.2, the Re-weighted estimator constructed here does not require that the nodes sampled are independent, and thus the implementation described in Section 4.2.2, which samples all nodes with one single random walk process, could be applied.

Analysis. In this part, we derive theoretical results on the number of nodes to sample in order to achieve some pre-set accuracy guarantee.

Theorem 4.5. *Let $1 > \delta > 0$, $\epsilon \leq 1$ and $k \geq 1$. Sampling k nodes in NeighborExploration will return an (ϵ,δ)-approximation estimation of F the by the Re-Weighted estimator, if*

$$k \geq \max\{18\frac{\sum_{y\in V}\frac{T(y)^2}{\pi_y} - 4F^2}{\epsilon^2\cdot 4F^2\cdot\delta}, 18\frac{\sum_{y\in V}\frac{1}{\pi_y} - |V|^2}{\epsilon^2\cdot|V|^2\cdot\delta}\}$$

Proof. Let Y be an random node sampled from one random walk step and $\pi_Y = \frac{d(Y)}{2|E|}$ be the probability of sampling node Y in the random walk process. Since $F = E[\frac{T(Y)}{\pi_Y}]/E[\frac{1}{\pi_Y}] \cdot \frac{|V|}{2}$ and $\widehat{F} = \widehat{E[\frac{T(Y)}{\pi_Y}]}/\widehat{E[\frac{1}{\pi_Y}]} \cdot \frac{|V|}{2}$, if $\widehat{E[\frac{T(Y)}{\pi_Y}]}/\widehat{E[\frac{1}{\pi_Y}]}$ is an (ϵ, δ)-approximation estimation of $E[\frac{T(Y)}{\pi_Y}]/E[\frac{1}{\pi_Y}]$, then $\widehat{F}$ is also an (ϵ, δ)-approximation estimation of F.

Assume that $\widehat{E[\frac{T(Y)}{\pi_Y}]}$ and $\widehat{E[\frac{1}{\pi_Y}]}$ are $(\epsilon/3, \delta/2)$-approximation estimations of $E[\frac{T(Y)}{\pi_Y}]$ and $E[\frac{1}{\pi_Y}]$ respectively. Let $A = \widehat{E[\frac{T(Y)}{\pi_Y}]}/\widehat{E[\frac{1}{\pi_Y}]}$ and $B = E[\frac{T(Y)}{\pi_Y}]/E[\frac{1}{\pi_Y}]$, then we have $P[A/B > \frac{1-\epsilon/3}{1+\epsilon/3} > 1 - \epsilon, A/B < \frac{1+\epsilon/3}{1-\epsilon/3} > 1+\epsilon] \geq (1-\delta/2)^2 > 1-\delta$. So $\widehat{E[\frac{T(Y)}{\pi_Y}]}/\widehat{E[\frac{1}{\pi_Y}]}$ is an (ϵ, δ)-approximation estimation of $E[\frac{T(Y)}{\pi_Y}]/E[\frac{1}{\pi_Y}]$.

Let $X = \frac{1}{k}\sum_{i=1}^{k} \frac{T(y_i)}{\pi_{y_i}}$ and $Z = \frac{1}{k}\sum_{i=1}^{k} \frac{1}{\pi_{y_i}}$. By Chebyshev's inequality in Appendix A, we have

$$Pr[|X - E[X]| > \epsilon/3 \cdot E[X]] \leq \frac{Var[X]}{(\epsilon/3 \cdot E[X])^2} \leq \delta/2 \quad (20)$$

$$Pr[|Z - E[Z]| > \epsilon/3 \cdot E[Z]] \leq \frac{Var[Z]}{(\epsilon/3 \cdot E[Z])^2} \leq \delta/2 \quad (21)$$

Since $E[X] = 2F, Var[X] = \frac{1}{k}(E[\frac{T(Y)^2}{\pi_Y^2}] - E^2[\frac{T(Y)}{\pi_Y}]) = \frac{1}{k}(\sum_{y\in V} \frac{T(y)^2}{\pi_y} - 4F^2), E[Z] = |V|$ and $Var[Z] = \frac{1}{k}(E[\frac{1}{\pi_Y^2}] - E^2[\frac{1}{\pi_Y}]) = \frac{1}{k}(\sum_{y\in V} \frac{1}{\pi_y} - |V|^2)$, we have

$$k \geq \max\{18\frac{\sum_{y\in V} \frac{T(y)^2}{\pi_y} - 4F^2}{\epsilon^2 \cdot 4F^2 \cdot \delta}, 18\frac{\sum_{y\in V} \frac{1}{\pi_y} - |V|^2}{\epsilon^2 \cdot |V|^2 \cdot \delta}\} \quad (22)$$

□

5 EXPERIMENTAL RESULTS

5.1 Experimental Set-up

Datasets. We used 5 real datasets which are publicly available and widely used in previous work [5, 11, 13] as shown in Table 1. In the experiment, we simulate the scenario where we only have accesses to the graphs via APIs. In each network, we remove the directions of edges, self-loops and multi-edges. We use the largest connected component for each network (since the method could be similarly run on other connected components) and the statistics of the largest connected components of networks are shown in Table 1.

In order to show the efficiency and effectiveness of our algorithms comprehensively, we use several types of labels to evaluate our algorithms. In Facebook and Google+, we use users' genders as node labels. In Pokec, we use users' locations as node labels. Such information can be obtained in the users' profiles in these networks. While in Orkut and Livejournal, the node degree is considered as the node label since we do not have the users' profiles in these two networks. Node degree contains structural information about the graph and in OSNs, it shows the number of friends that the user has. In order to simplify the discussion, all the labels are denoted by integers in the experiments.

Mixing Time. The mixing time of the Markov Chain is defined as the minimal length of the random walk in order to reach the stationary distribution. Following [2, 19], we define the mixing time of a Markov chain on G parameterized by a variation distance parameter ϵ as follows,

Definition 5.1. The mixing time parameterized by ϵ of a Markov Chain is defined as

$$\begin{aligned} T(\epsilon) &= \max_i \min\{t : |\pi - \pi^{(i)}|_1 < \epsilon\} \\ &= \max_i \min\{t : \tfrac{1}{2}\sum_{u\in V} |\pi(u) - [\pi^{(i)}P^t](u)| < \epsilon\} \end{aligned} \quad (23)$$

where vector π is the stationary distribution and P is the transition matrix. Vector $\pi^{(i)}$ is the initial distribution concentrated at node i, i.e. the i-th element is 1 and all the other elements are 0. $[\pi^{(i)}P^t](u)$ is the u-th element in $\pi^{(i)}P^t$. $|\pi - \pi_{(i)}|_1$ is the total variation distance which is a distance measure of two probability distributions.

After testing, we find that when $\epsilon = 10^{-3}$ which is small enough, the mixing time of Facebook, Google+, Pokec, Orkut and Livejournal is 3200, 200, 100, 800 and 900 respectively which are not very large. So it is easy to achieve the stationary distribution quickly in our experiments. Note that the nodes or edges encountered in the random walk before the mixing time are not included in the sample set.

Table 1: Statistics of Datasets

Network	$\|V\|$	$\|E\|$
Facebook [15]	4.0×10^3	8.82×10^4
Google+ [15]	1.08×10^5	1.22×10^7
Pokec [15]	1.6×10^6	2.23×10^7
Orkut[1]	3.08×10^6	1.17×10^8
Livejournal[1]	4.8×10^6	4.28×10^7

Table 2: Abbreviations of Algorithms

Algorithm Name	Abbreviation
NeighborSample with the Hansen-Hurwitz estimator	NeighborSample-HH
NeighborSample with the Horvitz-Thompson estimator	NeighborSample-HT
NeighborExploration with the Hansen-Hurwitz estimator	NeighborExploration-HH
NeighborExploration with the Horvitz-Thompson estimator	NeighborExploration-HT
NeighborExploration with the with Re-weighted method	NeighborExploration-RW
Existing algorithm using re-weighted method	EX-RW
Existing algorithm using Metropolis-Hastings random walk	EX-MHRW
Existing algorithm using maximum degree random walk	EX-MD
Existing algorithm using Rejection-controlled Metropolis-Hastings Random Walk Algorithm on Edges	EX-RCMH
Existing algorithm using General Maximum Degree Random Walk Algorithm on Edges	EX-GMD

Adaptations of Existing Algorithms. In addition to the two algorithms and their five corresponding estimators developed in this paper, we consider a few baseline methods adapted from an existing study [16]. In [16], the authors have summarized several common used algorithms which perform random walk on nodes to get unbiased estimation of the relative count of target nodes which has a particular degree. If we multiply this estimation by the total number of nodes, then we can obtain the estimation of the count of target nodes. Those existing methods cannot be applied directly to our problem, since our problem is to estimate the number of target edges instead of target nodes. However, we find that if we transform the original graph G into a new graph G', then we can apply those existing algorithms in [16] on graph G' to get the estimation of the count of target edges in G. We first describe how to construct G' base on G.

Let $G = (V, E)$ be the given graph, we construct a new graph $G' = (H, R)$ based on G with the following properties,

- Each edge in G corresponds to a node in G' and all these nodes constitute the node set H in G'. Thus, we have $|H| = |E|$.
- Two nodes in H are connected by an edge in G' if and only if they share one common vertex of G and all these edges constitute R.

where V and E are node set and edge set in G, and H and R are node set and edge set in G'.

It is obvious that if we apply the existing algorithms in [16] on graph G', then we can get the estimation of the count of target nodes in G'. Since each node in G' corresponds to an edge in G, counting the number of target edges in G is the same as counting the number of target nodes in G'.

In [16], three existing algorithms, Re-weighted method, Metropolis-Hastings Random Walk algorithm (MHRW), and Maximum Degree Random Walk algorithm (MDRW), are reviewed by the authors. Also, two new algorithms, Rejection-controlled Metropolis-Hastings Random Walk algorithm (RCMH) and General Maximum Degree Random Walk algorithm (GMD), are proposed by the authors. Two parameters, α and δ, are used to control the performance of RCMH and GMD, respectively. The authors suggested to set $\alpha \in$ [0,0.3] and $\delta \in$ [0.3,0.7], and in this paper, we adopt settings which give the best results.

The abbreviation of each tested algorithm is shown in Table 2, and all algorithms are implemented in C++, and we conducted experiments on a Linux machine with Intel 3.40GHz CPU.

Measurements. We adopt the *normalized root mean square error* (NRMSE) measure as our error measurement, which is defined as follows.

$$\text{NRMSE}(\widehat{F}) = \frac{\sqrt{\mathbb{E}[(\widehat{F}-F)^2]}}{F} = \frac{\sqrt{Var[\widehat{F}]+(F-\mathbb{E}[\widehat{F}])^2}}{F}, \qquad (24)$$

Note that NRMSE captures both the variance and the bias of the estimator.

Objectives. The objectives of the experiments can be summarized as follows:

(1) The diversity of the network types and corresponding label types serve to show that our methods sustain satisfactory performance across different domains.
(2) The different types of labels in different networks have very different frequencies. This helps us to investigate the effect of target edge frequency on the accuracy of the estimation.
(3) Another factor which can affect the accuracy is the sample size, we expect the accuracy to improve with more samples taken. Hence in our experiments, we vary the sample sizes and examine the impact.
(4) A major objective is to compare our proposed methods with the baseline methods, which are outlined in Section 5.1. We aim to show that our proposed algorithms outperform these baseline methods.
(5) Since we propose two algorithms, NeighborSample and NeighborExploration, it is of interest to compare the two and find out how they differ and how to choose between these algorithms depending on the given problem characteristics.

5.2 Comparison among algorithms with varying sample size

Firstly, we compare the estimation accuracies of different algorithms. We examine the NRMSE results of different algorithms while varying the sample size from $0.5\%|V|$ to $5\%|V|$. Each target edge label is represented in the form of (A, B) where A and B are two integers representing two node labels.

Table 3: The labels and their corresponding locations in Pokec

Label	Location
2	zilinsky kraj, kysucke nove mesto
13	zahranicie, zahranicie - australia
20	kosicky kraj, michalovce
24	trnavsky kraj, trnava
51	trnavsky kraj, skalica
86	bratislavsky kraj, bratislava - nove mesto
122	kosicky kraj, kosice - ostatne
135	banskobystricky kraj, dudince

In Facebook and Google+, we use one target edge label $(1, 2)$ (1 and 2 represent female and male respectively), while in Pokec, Orkut and Livejournal, we pick 4 different target edge labels to evaluate all algorithms. The results on Facebook are shown in Table 4. The results on Google+ are shown in Table 5. The results on Pokec are shown in Tables 6 - 9 (We use 4 target edge labels, (86,135), (2,51), (13,20), and (24,122). All these numbers represent locations using Slovak language, which are shown in Table 3). The results on Orkut are shown in Tables 10 - 13. The results on Livejournal are shown in Tables 14 - 17.

In these tables, each row shows the NRMSE of an algorithm with increasing sample size and each column shows the NRMSE of each algorithm for a fixed sample size. The target edge label, the count and the percentage count of the target edges are shown in the caption of each table. Each NRMSE value is calculated by averaging over 200 independent simulations.

In Pokec, Orkut and Livejournal, there are thousands of edge labels we can choose. We first order those edge labels in ascending order of the count of target edges and divide them into 4 parts with equal size, then we pick one target edge label from each part randomly. With this method, we can test our algorithms on both high frequency edge labels and low frequency edge labels.

Tables 18 - 22 show the bounds of number of samples needed to achieve an $(0.1, 0.1)$-approximation based Theorem 4.1 - 4.5. However, from the experimental results in Tables 4 - 17, we find that the number of samples needed to achieve a good estimation is much less than the bound.

The best NRMSE results for each sample size are underlined and marked with bold font. The best NRMSE results and the corresponding algorithms are also summarized in Tables 23 - 26 when $5\%|V|$ API calls are used.

We summarize our findings as follows.

(1) The best algorithm in each table is always one of our newly proposed algorithms (NeighborSample and NeighborExploration), demonstrating that our new algorithms outperform adaptations of existing algorithms.
(2) Our algorithms give good estimation with low API cost. Tables 23 - 26 summarize the best algorithms and the corresponding NRMSE values of each tested label when only $5\%|V|$ API calls are used. The largest NRMSE is 0.209 and most of the NRMSE values are smaller than 0.1. Note that for some tested target labels, the number of target edges is relatively small compared with the total number of edges, while our algorithms can still obtain accurate estimations. This shows that our proposed algorithms are highly effective.
(3) The NRMSE results of all algorithms decrease as the number of API calls increases, which means that our estimation

Table 4: Facebook, target label=(1,2), number of target edges=37400, precentage=42.4%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	0.341	0.227	0.187	0.182	0.171	0.164	0.153	0.142	0.129	0.127
NeighborSample-HT	**0.222**	**0.162**	**0.159**	**0.153**	**0.134**	**0.118**	**0.125**	**0.105**	**0.102**	**0.104**
NeighborExploration-HH	0.284	0.334	0.247	0.29	0.272	0.164	0.21	0.234	0.178	0.186
NeighborExploration-HT	0.465	0.509	0.52	0.371	0.332	0.296	0.338	0.234	0.324	0.271
NeighborExploration-RW	3.881	2.919	3.857	2.781	2.482	2.891	1.584	2.279	2.363	2.339
EX-MDRW	0.875	0.741	0.676	0.692	0.575	0.554	0.559	0.531	0.485	0.456
EX-MHRW	0.377	0.299	0.246	0.245	0.241	0.182	0.183	0.19	0.164	0.157
EX-RW	0.338	0.244	0.219	0.215	0.177	0.17	0.193	0.148	0.157	0.172
EX-RCMH	0.645	0.513	0.437	0.387	0.421	0.386	0.298	0.30	0.321	0.318
EX-GMD	0.277	0.240	0.181	0.188	0.162	0.179	0.171	0.156	0.156	0.145

Table 5: Google+, target label=(1,2), number of target edges=3280000, precentage=26.89%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	**0.089**	**0.061**	**0.053**	**0.046**	0.043	0.037	**0.032**	**0.031**	0.031	**0.029**
NeighborSample-HT	0.092	0.073	0.059	0.048	**0.04**	**0.036**	0.033	0.034	**0.029**	0.03
NeighborExploration-HH	0.7	0.689	0.642	0.627	0.647	0.58	0.558	0.582	0.49	0.491
NeighborExploration-HT	0.611	0.676	0.607	0.713	0.536	0.578	0.547	0.477	0.436	0.499
NeighborExploration-RW	13.506	11.856	16.765	21.985	19.323	16.279	15.079	11.97	6.65	16.06
EX-MDRW	0.478	0.451	0.443	0.379	0.24	0.269	0.259	0.225	0.261	0.207
EX-MHRW	0.169	0.118	0.089	0.078	0.075	0.06	0.066	0.053	0.057	0.055
EX-RW	0.162	0.117	0.113	0.08	0.078	0.07	0.066	0.067	0.058	0.051
EX-RCMH	0.161	0.108	0.09	0.074	0.066	0.051	0.062	0.063	0.052	0.043
EX-GMD	0.388	0.302	0.228	0.252	0.211	0.187	0.163	0.178	0.169	0.161

Table 6: Pokec, target label=(86,135), number of target edges=295, precentage=0.001%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	2.526	1.935	1.413	1.608	1.273	1.38	1.381	1.074	1.007	1.016
NeighborSample-HT	2.802	1.862	1.478	1.378	0.965	1.124	1.174	0.826	1.323	0.853
NeighborExploration-HH	**0.761**	**0.606**	**0.445**	**0.339**	0.386	0.426	**0.302**	0.36	**0.238**	**0.209**
NeighborExploration-HT	2.023	0.778	0.541	0.659	0.512	**0.307**	0.364	**0.233**	0.3	0.241
NeighborExploration-RW	1.861	0.685	0.542	0.466	**0.325**	0.362	0.457	0.317	0.355	0.307
EX-MDRW	1.0	1.0	1.0	1.0	104.73	1.0	16.607	2.222	13.005	1.0
EX-MHRW	3.492	2.597	1.935	1.783	1.184	1.521	1.527	2.105	1.136	1.47
EX-RW	3.52	2.22	2.656	2.555	1.472	1.533	1.292	1.532	1.237	1.921
EX-RCMH	0.949	0.607	0.450	0.477	0.430	0.405	0.303	0.352	0.314	0.226
EX-GMD	1.0	1.0	1.23	1.35	0.98	2.45	1.23	0.88	0.93	1.06

Table 7: Pokec, target label=(2,51),number of target edges=1163, precentage=0.005%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	1.262	1.036	0.748	0.73	0.701	0.649	0.551	0.478	0.503	0.444
NeighborSample-HT	1.62	1.17	0.768	0.742	0.739	0.578	0.759	0.559	0.599	0.461
NeighborExploration-HH	0.448	**0.301**	0.319	**0.203**	**0.177**	0.169	**0.139**	**0.129**	0.16	**0.124**
NeighborExploration-HT	**0.424**	0.401	**0.235**	0.203	0.196	**0.159**	0.188	0.139	**0.149**	0.149
NeighborExploration-RW	0.941	0.407	0.257	0.231	0.22	0.175	0.188	0.156	0.172	0.155
EX-MDRW	1.0	3.104	13.812	1.0	27.873	2.122	1.649	4.274	2.599	2.392
EX-MHRW	1.494	1.248	1.132	0.886	0.987	0.759	0.611	0.628	0.506	0.624
EX-RW	1.905	1.604	1.4	0.996	0.921	0.665	0.719	0.751	0.655	0.528
EX-RCMH	1.65	1.00	0.971	0.759	0.648	0.709	0.628	0.511	0.613	0.497
EX-GMD	1.0	5.79	1.0	1.07	1.57	6.08	1.34	3.36	1.68	1.25

Table 8: Pokec, target label=(13,20), number of target edges=2134, precentage=0.01%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	1.108	0.85	0.635	0.696	0.522	0.531	0.448	0.374	0.404	0.36
NeighborSample-HT	1.555	0.877	0.72	0.552	0.565	0.607	0.458	0.381	0.406	0.382
NeighborExploration-HH	0.396	**0.264**	0.228	**0.192**	0.164	0.176	**0.139**	0.137	0.136	**0.12**
NeighborExploration-HT	0.445	0.28	**0.205**	0.211	0.173	**0.156**	0.15	**0.128**	0.138	0.104
NeighborExploration-RW	**0.344**	0.275	0.214	0.194	**0.149**	0.163	0.144	0.135	**0.127**	0.146
EX-MDRW	7.56	1.0	9.953	11.815	25.159	3.314	8.077	8.987	3.582	2.476
EX-MHRW	1.373	1.291	0.935	0.706	0.695	0.552	0.545	0.546	0.539	0.415
EX-RW	1.803	1.885	0.864	0.679	0.678	0.58	0.616	0.639	0.451	0.548
EX-RCMH	1.21	0.811	0.625	0.877	0.541	0.461	0.496	0.442	0.527	0.419
EX-GMD	1.27	1.0	1.0	1.28	1.00	1.77	3.05	2.30	2.67	1.24

Table 9: Pokec, target label=(24,122), number of target edges=5784, precentage=0.03%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	0.727	0.532	0.41	0.358	0.291	0.314	0.282	0.25	0.229	0.213
NeighborSample-HT	0.839	0.46	0.409	0.292	0.226	0.34	0.25	0.292	0.267	0.189
NeighborExploration-HH	0.349	**0.247**	**0.196**	0.192	**0.154**	**0.124**	0.141	**0.115**	**0.096**	0.101
NeighborExploration-HT	0.382	0.29	0.214	**0.156**	0.178	0.143	**0.117**	0.118	0.107	**0.093**
NeighborExploration-RW	**0.342**	0.251	0.204	0.214	0.165	0.147	0.122	0.115	0.121	0.095
EX-MDRW	1.163	20.821	6.422	2.987	2.546	4.971	2.431	6.339	2.183	5.172
EX-MHRW	1.063	0.592	0.516	0.57	0.452	0.354	0.387	0.324	0.393	0.294
EX-RW	0.996	0.84	0.643	0.455	0.482	0.467	0.501	0.39	0.328	0.334
EX-RCMH	0.949	0.607	0.450	0.477	0.430	0.405	0.303	0.352	0.314	0.226
EX-GMD	1.67	1.12	3.03	2.58	2.30	1.81	1.41	1.09	1.25	1.90

Table 10: Orkut, target label=(48,45), number of target edges=5627, precentage=0.001%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	1.08	0.884	0.688	0.705	0.578	0.473	0.402	0.379	0.436	0.332
NeighborSample-HT	0.917	0.812	0.487	0.687	0.689	0.343	0.485	0.533	0.341	0.332
NeighborExploration-HH	**0.315**	0.265	0.195	**0.172**	0.146	0.141	**0.099**	0.116	**0.096**	**0.089**
NeighborExploration-HT	0.395	**0.237**	**0.154**	0.185	**0.14**	**0.117**	0.123	**0.105**	0.109	0.114
NeighborExploration-RW	0.479	0.304	0.215	0.185	0.161	0.156	0.124	0.118	0.116	0.099
EX-MDRW	1.407	9.96	12.876	13.999	10.188	4.846	4.834	3.759	2.085	3.039
EX-MHRW	1.51	0.852	0.843	0.673	0.601	0.613	0.505	0.471	0.429	0.41
EX-RW	1.181	0.693	0.599	0.558	0.542	0.512	0.378	0.41	0.366	0.373
EX-RCMH	0.944	0.670	0.649	0.524	0.425	0.362	0.37	0.379	0.35	0.32
EX-GMD	1.0	1.34	3.41	1.48	1.28	1.40	1.50	1.70	1.35	1.44

Table 11: Orkut, target label=(11,0),number of target edges=49879, precentage=0.043%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	0.386	0.357	0.231	0.237	0.168	0.186	0.163	0.148	0.147	0.142
NeighborSample-HT	**0.284**	0.278	0.249	0.268	0.216	0.197	0.149	0.136	0.167	0.152
NeighborExploration-HH	0.491	0.331	0.278	0.228	0.207	0.193	0.168	0.143	0.147	0.147
NeighborExploration-HT	0.425	0.286	0.274	0.21	0.198	0.185	0.156	0.143	0.152	0.122
NeighborExploration-RW	0.31	**0.268**	**0.202**	**0.188**	**0.151**	**0.166**	**0.149**	**0.122**	**0.111**	**0.124**
EX-MDRW	8.977	6.288	2.317	6.809	8.318	7.408	8.366	3.708	2.174	2.973
EX-MHRW	0.662	0.49	0.381	0.368	0.345	0.261	0.291	0.243	0.238	0.212
EX-RW	1.005	0.788	0.678	0.544	0.503	0.424	0.37	0.379	0.355	0.373
EX-RCMH	0.997	0.651	0.491	0.453	0.384	0.312	0.268	0.281	0.271	0.261
EX-GMD	0.995	3.51	1.78	3.11	1.66	1.76	1.39	2.35	1.71	1.47

Table 12: Orkut, target label=(1,0),number of target edges=128501, precentage=0.11%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	0.198	0.162	0.124	0.112	0.113	0.1	0.086	0.081	0.075	0.068
NeighborSample-HT	**0.182**	**0.125**	**0.106**	**0.084**	**0.088**	**0.085**	**0.067**	**0.064**	0.065	**0.063**
NeighborExploration-HH	0.491	0.331	0.278	0.228	0.207	0.193	0.168	0.143	0.147	0.147
NeighborExploration-HT	0.212	0.156	0.136	0.113	0.11	0.087	0.089	0.077	**0.063**	0.071
NeighborExploration-RW	0.523	0.35	0.253	0.215	0.189	0.187	0.17	0.136	0.154	0.15
EX-MDRW	8.977	6.288	2.317	6.809	8.318	7.408	8.366	3.708	2.174	2.973
EX-MHRW	11.117	27.794	11.387	8.223	3.273	4.57	1.0	1.619	4.748	5.788
EX-RW	0.662	0.49	0.381	0.368	0.345	0.261	0.291	0.243	0.238	0.212
EX-RCMH	1.18	0.985	0.75	0.688	0.511	0.553	0.473	0.436	0.427	0.468
EX-GMD	1.0	5.72	5.74	1.22	3.26	1.97	1.45	2.05	1.69	2.92

Table 13: Orkut, target label=(6,5),number of target edges=769188, precentage=0.657%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	0.124	0.079	0.067	0.066	0.056	0.054	0.043	0.045	0.042	0.038
NeighborSample-HT	0.107	0.072	0.071	0.057	**0.04**	0.04	0.049	0.043	0.032	0.039
NeighborExploration-HH	0.159	0.105	0.096	0.077	0.075	0.067	0.07	0.053	0.053	0.05
NeighborExploration-HT	0.136	0.105	0.1	0.085	0.07	0.063	0.056	0.06	0.054	0.046
NeighborExploration-RW	**0.084**	**0.063**	**0.057**	**0.043**	0.045	**0.04**	**0.037**	**0.029**	**0.028**	**0.029**
EX-MDRW	3.514	2.366	2.551	2.373	1.451	1.403	1.6	1.29	1.297	1.026
EX-MHRW	0.186	0.128	0.105	0.099	0.091	0.082	0.074	0.062	0.068	0.055
EX-RW	0.352	0.231	0.214	0.156	0.153	0.133	0.115	0.116	0.102	0.096
EX-RCMH	0.238	0.178	0.159	0.116	0.116	0.091	0.101	0.087	0.082	0.078
EX-GMD	1.84	0.914	0.904	0.712	0.77	0.705	0.646	0.683	0.693	0.64

Table 14: Livejournal, target label=(34,12),number of target edges=5168, precentage=0.001%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	0.62	0.445	0.338	0.308	0.25	0.232	0.272	0.254	0.179	0.198
NeighborSample-HT	0.6	0.45	0.259	0.252	0.326	0.243	0.175	0.24	0.209	0.218
NeighborExploration-HH	0.264	**0.164**	**0.158**	0.14	0.138	0.094	**0.085**	0.098	0.089	0.088
NeighborExploration-HT	**0.231**	0.168	0.173	**0.114**	0.2	0.117	0.144	**0.083**	**0.086**	**0.074**
NeighborExploration-RW	1.089	0.205	0.244	0.167	**0.121**	**0.108**	0.112	0.113	0.099	0.091
EX-MDRW	3.482	1.666	3.297	3.952	2.436	4.498	2.553	2.273	1.647	3.465
EX-MHRW	0.669	0.589	0.422	0.373	0.318	0.303	0.278	0.28	0.247	0.245
EX-RW	0.587	0.451	0.324	0.382	0.267	0.253	0.19	0.161	0.213	0.179
EX-RCMH	0.564	0.343	0.303	0.263	0.248	0.216	0.199	0.186	0.171	0.159
EX-GMD	1.86	1.72	1.70	0.991	1.60	1.30	1.00	0.850	1.11	0.987

Table 15: Livejournal, target label=(19,16), number of target edges=15442, precentage=0.04%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	0.442	0.291	0.196	**0.198**	0.193	0.174	0.139	0.144	**0.116**	0.119
NeighborSample-HT	0.557	0.172	0.257	0.182	0.16	0.146	0.157	0.166	0.129	0.117
NeighborExploration-HH	**0.393**	**0.277**	0.265	0.204	**0.15**	**0.136**	**0.125**	0.136	0.118	**0.105**
NeighborExploration-HT	0.466	0.293	**0.236**	0.204	0.156	0.164	0.157	**0.132**	0.133	0.115
NeighborExploration-RW	0.543	0.327	0.278	0.263	0.167	0.159	0.173	0.154	0.13	0.129
EX-MDRW	3.447	4.478	2.861	1.834	2.201	1.527	4.163	1.615	1.89	2.148
EX-MHRW	0.637	0.356	0.303	0.32	0.242	0.233	0.233	0.221	0.187	0.187
EX-RW	0.742	0.359	0.337	0.275	0.333	0.241	0.198	0.194	0.172	0.167
EX-RCMH	0.476	0.282	0.239	0.257	0.224	0.195	0.157	0.151	0.128	0.138
EX-GMD	2.52	1.30	1.26	1.23	1.16	1.33	0.853	0.980	0.735	0.822

Table 16: Livejournal, target label=(8,4), number of target edges=203945 precentage=0.48%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	**0.104**	**0.092**	0.082	**0.06**	**0.051**	**0.048**	**0.038**	**0.04**	**0.039**	0.04
NeighborSample-HT	0.105	0.101	0.08	0.053	0.06	0.042	0.048	0.046	0.04	0.04
NeighborExploration-HH	0.138	0.107	0.09	0.078	0.057	0.055	0.067	0.06	0.043	0.048
NeighborExploration-HT	0.135	0.117	0.101	0.103	0.082	0.095	0.105	0.084	0.094	0.087
NeighborExploration-RW	0.152	0.1	**0.068**	0.061	0.084	0.056	0.055	0.054	0.061	**0.039**
EX-MDRW	1.761	1.535	1.613	1.191	1.191	1.066	1.442	0.942	1.03	0.781
EX-MHRW	0.19	0.127	0.112	0.094	0.071	0.073	0.064	0.064	0.047	0.051
EX-RW	0.201	0.167	0.134	0.096	0.104	0.084	0.08	0.083	0.074	0.082
EX-RCMH	0.172	0.128	0.105	0.093	0.073	0.0745635	0.07	0.08	0.056	0.053
EX-GMD	1.32	0.835	0.717	0.666	0.642	0.613	0.591	0.592	0.546	0.541

Table 17: Livejournal, target label=(1,0), number of target label=1753000, precentage=4.1%

	0.5%$\|V\|$	1.0%$\|V\|$	1.5%$\|V\|$	2.0%$\|V\|$	2.5%$\|V\|$	3.0%$\|V\|$	3.5%$\|V\|$	4.0%$\|V\|$	4.5%$\|V\|$	5.0%$\|V\|$
NeighborSample-HH	0.094	0.054	0.053	0.041	0.04	0.041	0.031	0.029	0.026	0.028
NeighborSample-HT	0.07	0.057	0.048	0.04	0.037	0.031	0.035	0.027	0.027	0.025
NeighborExploration-HH	0.152	0.083	0.072	0.051	0.052	0.049	0.042	0.037	0.035	0.033
NeighborExploration-HT	0.119	0.089	0.076	0.067	0.056	0.05	0.049	0.044	0.042	0.044
NeighborExploration-RW	**0.053**	**0.048**	**0.043**	**0.033**	**0.031**	**0.026**	**0.027**	**0.024**	**0.023**	**0.02**
EX-MDRW	3.332	1.757	1.825	0.978	1.14	1.213	1.101	0.95	0.884	0.935
EX-MHRW	0.196	0.131	0.09	0.096	0.079	0.07	0.079	0.068	0.057	0.064
EX-RW	0.252	0.211	0.186	0.145	0.122	0.107	0.11	0.128	0.103	0.08
EX-RCMH	0.155	0.175	0.133	0.111	0.0786	0.092	0.08	0.09	0.067	0.073
EX-GMD	1.16	0.958	0.867	0.726	0.705	0.654	0.642	0.647	0.686	0.59

Table 18: Bound on the number of samples in Facebook

	NeighborSample-HH	NeighborSample-HT	NeighborExploration-HH	NeighborExploration-HT	NeighborExploration-RW
(1,2)	1359	5398	**921**	3151	53427

Table 19: Bound on the number of samples in Google+

	NeighborSample-HH	NeighborSample-HT	NeighborExploration-HH	NeighborExploration-HT	NeighborExploration-RW
(1,2)	2726	13879	**1714**	25400	445515

converges to the ground truth when more samples or API calls are used. This behavior is as expected.

(4) In most of the cases, NeighborExploration returns the best estimations. However, NeighborSample outperforms NeighborExploration in the cases where the target edges constitute a larger proportion in the whole edge set. This is the case for the results of facebook and google+. This indicates that when the target edges are abundant, neighborhood exploration is not needed to boost the sampling probability for the target edges.

(5) For the datasets of Orkut and Livejournal, we have multiple sets of results for edge labels with different frequencies. It is found that the NRMSE values for more frequent labels are generally smaller than those with less frequent labels. We have therefore conducted a more systematic study about the impact of the label frequency, to be reported in the next subsection.

5.3 Comparison among algorithms with varying relative count of target edges

We notice that in the same graph, for different target labels the best algorithms can be different. It turns out that the relative count of target edges ($F/|E|$) may also affect the performance of different algorithms. In order to study the relationship between the performance of different algorithms and the relative count of target edges, we measure the values of NRMSE for a range of $F/|E|$. The results for Orkut and Livejournal are plotted in Figure 1 and Figure 2. Each node in these figures corresponds a target edge label and the x-coordinate is the relative count of edges with this label and the y-coordinate is the NRMSE of the target edge count estimation when 5%$|V|$ API calls are used. Here, we only run experiments on two networks, Orkut and Livejournal, since the range of the relative count of target edges in Orkut and Livejournal is much larger than the ranges in other networks, so the change of NRMSE is more obvious in these two networks when the relative count of target edges varies. The results of existing algorithms are not shown, since we have demonstrated that those algorithms are much less competitive in the previous experiments. Each NRMSE value is calculated by averaging over 200 independent simulations.

We summarize our results as follows.

(1) In the same network, as the relative count of target edges increases, the NRMSE results of all algorithms decrease, which means that the estimation is more accurate. This is reasonable, since the probability of sampling target edges in random walk will be higher if there are more target edges in the networks, which will result in better estimations.

(2) When the relative count of target edges changes the best algorithm may also be different. When the relative count of target edges is small, NeighborExploration algorithms outperforms NeighborSample algorithms and the difference is quite significant, but when the relative count of target edges is large enough, the results of NeighborExploration algorithms and NeighborSample algorithms are very close and the best algorithm will change from case to case.

We explain why our new algorithm NeighborExploration outperforms NeighborSample when the relative count of target edges is small as follows. In NeighborSample algorithms, the probability of sampling a target edge is $\frac{F}{|E|}$, since NeighborSample samples edges uniformly. However, our new algorithm, NeighborExploration, can find a target edge with probability $\frac{\sum_{u\in Q} d_u}{2|E|}$, where node set Q contains all nodes which is included in at least one target edges, since once we sample a node all target edges which contains this node will also be found. As a result, our new algorithm NeighborExploration can obtain target edges with a

Table 20: Bounds on the number of samples in Pokec

	NeighborSample-HH	NeighborSample-HT	NeighborExploration-HH	NeighborExploration-HT	NeighborExploration-RW
(86,135)	7.56×10^7	2.77×10^8	$\underline{\mathbf{4.08} \times 10^6}$	3.77×10^8	7.35×10^7
(2,51)	1.91×10^7	2.16×10^8	$\underline{\mathbf{6.9} \times 10^5}$	2.54×10^8	1.25×10^7
(13,20)	1.04×10^7	1.89×10^8	$\underline{\mathbf{4.6} \times 10^5}$	2.01×10^8	8.27×10^6
(24,122)	3.85×10^6	1.45×10^8	$\underline{\mathbf{2.3} \times 10^5}$	1.15×10^8	4.16×10^6

Table 21: Bounds on the number of samples in Orkut

	NeighborSample-HH	NeighborSample-HT	NeighborExploration-HH	NeighborExploration-HT	NeighborExploration-RW
(48,45)	2.08×10^7	9.62×10^8	$\underline{\mathbf{2.46} \times 10^5}$	4.8×10^6	4.44×10^6
(11,0)	2.34×10^6	4.5×10^8	$\underline{\mathbf{3.3} \times 10^5}$	1.03×10^8	6.03×10^6
(1,0)	9.1×10^5	2.45×10^8	$\underline{\mathbf{1.8} \times 10^5}$	3.97×10^7	3.34×10^6
(6,5)	1.5×10^5	2.11×10^7	$\underline{\mathbf{1.3} \times 10^4}$	1.38×10^6	2.49×10^5

Table 22: Bounds on the number of samples in Livejournal

	NeighborSample-HH	NeighborSample-HT	NeighborExploration-HH	NeighborExploration-HT	NeighborExploration-RW
(34,12)	8.28×10^6	3.16×10^8	$\underline{\mathbf{1.8} \times 10^5}$	5.86×10^6	3.23×10^6
(19,16)	2.77×10^6	2.22×10^8	$\underline{\mathbf{9.6} \times 10^4}$	4.45×10^6	1.74×10^6
(8,4)	2.09×10^5	3.0×10^7	$\underline{\mathbf{1.7} \times 10^4}$	7.10×10^6	3.09×10^5
(6,5)	2.34×10^4	5.93×10^5	$\underline{\mathbf{9.8} \times 10^3}$	6.0×10^5	1.76×10^5

higher probability than NeighborSample, especially when $F/|E|$ is small, so NeighborExploration performs better.

Table 23: Best algorithm for Facebook and Google+ using $5\%|V|$ API calls

Social Network	Label	Best algorithm	NRMSE
Facebook	(1,2)	NeighborSample-HT	0.104
Google+	(1,2)	NeighborSample-HH	0.029

Table 24: Best algorithm for Pokec using $5\%|V|$ API calls

Label	Best algorithm	NRMSE
(135,86)	NeighborExploration-HH	0.209
(2,51)	NeighborExploration-HH	0.124
(13,20)	NeighborExploration-HH	0.12
(24,122)	NeighborExploration-HT	0.093

Table 25: Best algorithm for Orkut using $5\%|V|$ API calls

Label	Best algorithm	NRMSE
(48,45)	NeighborExploration-HH	0.089
(11,0)	NeighborExploration-RW	0.124
(1,0)	NeighborSample-HT	0.063
(6,5)	NeighborSample-RW	0.029

Table 26: Best algorithm for Livejournal using $5\%|V|$ API calls

Label	Best algorithm	NRMSE
(34,12)	NeighborExploration-HT	0.074
(19,16)	NeighborExploration-HH	0.105
(8,4)	NeighborExploration-RW	0.039
(1,0)	NeighborExploration-RW	0.02

6 CONCLUSION

In this paper, we propose to estimate the number of edges with target labels, which to the best of our knowledge corresponds to the first attempt to estimate graph properties refined by users' labels. To solve the problem, we developed two algorithms, namely NeighborSample and NeighborExploration, which samples a set of edges/nodes first and then constructs estimators based on the sampled edges/nodes. These two algorithms are suitable in different cases, e.g., NeighborExploration is better than NeighborSample when the fraction of edges with target labels is low. We also provide some theoretical results on the accuracy guarantees of the algorithms. We conducted extensive experiments which verified that the algorithms developed in this paper are superior over baseline methods.

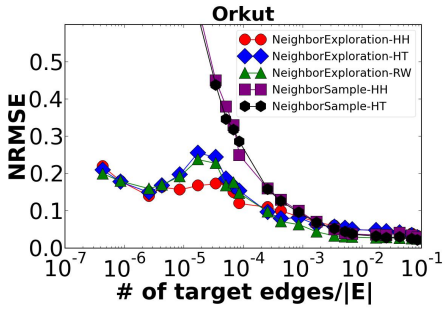

Figure 1: NRMSE vs. number of target edges in Orkut when $5\%|V|$ API calls are used

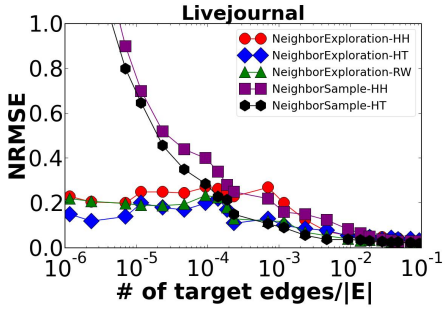

Figure 2: NRMSE vs. number of target edges in Livejournal when $5\%|V|$ API calls are used

There are a few directions for future study. For example, it would be interesting to estimate some other types of graph properties such as numbers of wedges and triangles refined by users' labels in OSNs.

REFERENCES

[1] 2016. KONECT Datasets: The koblenz network collection. http://konect.uni-koblenz.de. (2016).

[2] Louigi Addario-Berry and Tao Lei. 2015. The Mixing Time of the Newman-Watts Small-World Model. *Advances in Applied Probability* 47, 1 (2015), 37–56.

[3] P. Anchuri, M.J. Zaki, O. Barkol, S. Golan, and M. Shamy. 2013. Approximate Graph Mining with Label Costs. *In KDD* (2013), 518–526.

[4] A. Arora, M. Sachan, and A. Bhattacharya. 2014. Mining Statistically Significant Connected Subgraphs in Vertex Labeled Graphs. *In SIGMOD* (2014), 1003–1014.

[5] X. Chen, Y. Li, G. P. Wang, and J. C. S. Lui. 2016. A general framework for estimating graphlet statistics via random walk. *Proc. VLDB Endow.* 10, 3 (2016), 253–264.

[6] Y. Press D. A. Levin and E. L. Wilmer. 2008. *Markov Chains and Mixing Times.* American Mathematical Society.

[7] M. Gjoka, M. Kurant, and C. T. Butts. 2010. Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. *In INFOCOM* (2010), 1–9.
[8] Olle Häggström. 2002. *Finite Markov chains and algorithmic applications*. Vol. 52. Cambridge University Press.
[9] J. Han and J-R. Wen. 2013. Mining Frequent Neighborhood Patterns in a Large Labeled Graph. *In CIKM* (2013), 259–268.
[10] M. Hansen and W. Hurwitz. 1943. On the Theory of Sampling from Finite Populations. *In Annals of Mathematical Statistics* 14, 4 (1943), 333–362.
[11] S. J. Hardiman and L. Katzir. 2013. Estimating clustering coefficients and size of social networks via random walk. *In WWW* (2013), 539–550.
[12] D. G. Horvitz and D. J. Thompson. 1952. A Generalization of Sampling Without Replacement from a Finite Universe. *J. Amer. Statist. Assoc.* 47, 260 (1952), 663–685.
[13] L. Katzir, E. Liberty, and O. Somekh. 2011. Estimating sizes of social networks via biased sampling. *In WWW* (2011), 597–606.
[14] C.-H. Lee, X. Xu, and D. Y. Eun. 2012. Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. *In SIGMETRICS* (2012), 319–330.
[15] J. Leskovec and A. Krevl. 2016. SNAP Datasets: Standford large network dataset collection. http://snap.standford.edu/data. (2016).
[16] R.-H. Li, J. Yu, L. Qin, R. Mao, and T. Jin. 2015. On random walk based graph sampling. *In ICDE* (2015), 927–938.
[17] J. S. Liu. 2001. Monte Carlo Strategies in Scientific Computing. *Springer* (2001).
[18] LÃąszlÃş LovÃąsz. 1993. Random Walks on Graphs: A Survey. *In Combinatorics, Paul Erdos is Eighty* 2 (1993).
[19] Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. 2010. Measuring the mixing time of social graphs. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 383–389.
[20] S. Nandanwar and M.N. Murty. 2016. Structural Neighborhood Based Classification of Nodes in a Network. *In KDD* (2016), 518–526.
[21] P. Wang, J. C. S. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan. 2014. Efficiently estimating motif statistics of large networks. *In ICDE* (2014), 8:1–8:27.
[22] P. Wang, J. Zhao, J. C. Lui, D. Towsley, and X. Guan. 2015. Unbiasedcharacterization of node pairs over large graphs. *In TKDD* 9, 3 (2015).
[23] Y. Wu, C. Long, A. W. Fu, and Z. Chen. 2017. Counting Edges and Triangles in Online Social Networks via Random Walk. *In APWeb-WAIM* (2017), 346–361.
[24] W. Ye, L. Zhou, D. Mautz, C. Plant, and C. Böhm. 2017. Learning from Labeled and Unlabeled Vertices in Networks. *In KDD* (2017), 1265–1274.

A CHEBYSHEV'S INEQUALITY

Let X be a random variable with expectation $E[X]$ and variance $var(X)$. Then the Chebyshev's inequality states that for any $t > 0$,

$$P(|X - E[X]| > t) \leq \frac{var(X)}{t^2} \tag{25}$$

Let $\widehat{X}$ be an estimator of $E[X]$, $t = \epsilon E[X]$ where $0 < \epsilon < 1$ and $0 < \delta < 1$, then we call $\hat{X}$ an (ϵ, δ)-approximation of $E[X]$, if the following Chebyshev's inequality holds.

$$P(|\widehat{X} - E[X]| > \epsilon E[X]) \leq \frac{var(X)}{(\epsilon E[X])^2} \leq \delta \tag{26}$$

which shows that the probability of that $\widehat{X}$ is in the range $[(1 - \epsilon)E[X], (1 + \epsilon)E[X]]$ is larger than $1 - \delta$.