Walk, Not Wait: Faster Sampling Over Online Social Networks

Azade Nazi[‡], Zhuojie Zhou[††], Saravanan Thirumuruganathan[‡], Nan Zhang[††], Gautam Das[‡]
[‡]University of Texas at Arlington; [††]George Washington University
[‡]{azade.nazi@mavs, saravanan.thirumuruganathan@mavs, gdas@cse}.uta.edu, [††]{rexzhou,nzhang10}@gwu.edu

ABSTRACT

In this paper, we introduce a novel, general purpose, technique for faster sampling of nodes over an online social network. Specifically, unlike traditional random walk which wait for the convergence of sampling distribution to a predetermined target distribution - a waiting process that incurs a high query cost - we develop WALK-ESTIMATE, which starts with a much shorter random walk, and then proactively estimate the sampling probability for the node taken before using acceptance-rejection sampling to adjust the sampling probability to the predetermined target distribution. We present a novel backward random walk technique which provides provably unbiased estimations for the sampling probability, and demonstrate the superiority of WALK-ESTIMATE over traditional random walks through theoretical analysis and extensive experiments over real world online social networks.

1. INTRODUCTION

Online social networks often feature a web interface that only allows local-neighborhood queries - i.e., given a user of the online social network as input, the system returns the immediate neighbors of the user. In this paper, we address the problem of enabling third-party analytics over an online social network through its restrictive web interface. As demonstrated by a wide range of existing services (e.g., Twitris, Toretter, AIDR), such third-party analytics applications benefit not only social network users, social scientists, but the entire society at large (e.g., through epidemic control).

1.1 Problem of Existing Work

The restrictive local-neighborhood-only access interface makes it extremely difficult for a third party to crawl all data from an online social network, as a complete crawl requires as many queries as the number of users in the social network. To address this challenge and enable analytics tasks such as aggregate estimation through the restrictive access interface, many existing studies resort to the *sampling* of users from the online social network. If we consider a social network as a graph, the idea here is to first draw a sample of nodes from the graph, and then generate (statistically accurate) aggregate estimations based on the sample nodes.

The nature of the interface limitation - i.e., allowing only local-neighborhood queries - makes *random walk* based Monte Carlo Markov Chain (MCMC) methods an ideal fit for the sampling of users from an online social network. Intuitively, a random walk starts from an arbitrary user, and then randomly moves to one of its neighbors selected randomly according to a pre-determined probability distribution (namely the *transition probability*). The movement continues for a number of steps, namely the "*burn-in period*" before the node being selected is taken as a sample.

A critical problem with the existing random walk techniques, however, is the long burn-in period it requires and, therefore, the significant query cost it incurs for drawing a sample. Since many online social networks limit the number of queries one can issue (e.g., from an IP address) within a period of time (e.g., Twitter allows , every 15 minutes, only 15 API requests to retrieve ids of a user's followers) the high query cost limits the sample size one can draw from the social network and, consequently, the accuracy of analytics tasks.

To understand why the burn-in period is required, an important observation is that, before a sample can be used for analytical purposes, we must know the *sampling distribution* - i.e., the probability for each node in the graph to be sampled - because without such knowledge, one might "over-consider" certain parts of the graph in the analytics tasks, leading to errors such as biased aggregate estimations. However, since a third party has no knowledge of even the global graph topology, it seems infeasible to compute the sampling distribution for a random walk. Fortunately, the property of MCMC methods ensures that, as a random walk grows longer, the sampling distribution becomes asymptotically close to a *stationary distribution* that can be computed from the design of transition probabilities alone. For example, with a simple random walk (featuring a uniform transition distribution - see Section 2.2 for details), the stationary probability for a node to be sampled is always proportional to its degree, no matter how the global graph topology looks like.

It is also the availability of such a stationary distribution that leads to the mandate of a burn-in period. Note that, while the MCMC property ensures asymptotic convergence to the stationary distribution, the actual convergence process can be slow - and the length of burn-in required is essentially uncomputable without knowledge of the entire graph topology [19, 24]. Facing this problem, what the existing techniques can do is to either set a conservatively large burn-in period [23, 24], or use one of the heuristic convergence monitors and "wait" for the sampling distribution to converge to its stationary value. In either case, the sampling process may require a large number of queries during the burn-in period.

1.2 Our Idea: WALK-ESTIMATE

In this paper, our objective is to significantly reduce the query cost of node sampling over an online social network by (nearly) eliminating the costly waiting process. Of course, as one can see from the above discussions, if we do not wait for the convergence to stationary distribution, we must somehow estimate the probability for our short walk to take a node as a sample (i.e., the node's *sampling probability*) before we can use the node as a sample. This is exactly what we do - i.e., we introduce a novel idea of having a (much) shorter, say t-step, walk, before taking a node v as a sample candidate, but follow it up with a *proactive* process which estimates v's sampling probability $p_t(v)$ - i.e., the probability for our walk to reach node v at Step t, so that we can then use acceptance-rejection sampling to "correct" the sampling probability to the desired distribution. As we shall prove in the paper, even though the acceptance-rejection step introduces additional query cost, the savings from having a shorter walk in the first place far outweighs the additional cost, leading to a significantly more efficient sampling process.

Based on this idea, we develop Algorithm WALK-ESTIMATE. The algorithm takes as input a random walk based MCMC sampler, and produces samples according to the exact same distribution as the input sampler - i.e., the stationary distribution of the MCMC process. One can see that this design makes WALK-ESTIMATE transparent to the desired target distribution, making it a swap-in replacement for any random walk sampler being used (e.g., simple random walk [19, 21], Metropolis-Hastings random walk [19, 21]. As we shall demonstrate through theoretical analysis and experimental evaluation over real-world social networks, while the proactive probability-estimation process may consume a small number of queries, the significant savings from the shorter walk more than offset the additional consumption, and lead to a much more efficient sampling process over online social networks.

1.3 Outline of Technical Results

We now provide an overview of the main technical results in the design of WALK-ESTIMATE. The algorithm is enabled by two main components: WALK and ESTIMATE. The WALK component determines how many, say t, steps to (randomly) transit before taking a node v as a candidate (for sampling), and then calls upon the ESTIMATE component for an estimation of the probability for the walk to reach v after t steps. Based on the estimated probability, WALK then performs acceptance-rejection sampling on v to determine if it should be included in the final (output) sample.

For the WALK component, we start by developing IDEAL-WALK, an impractical sampler which makes two ideal-case assumptions: One is access to an oracle that precisely compute the $p_t(v)$, i.e., the probability for the walk to reach a node v at Step t. The other is access to certain *global* topological information about the underlying graph - e.g., $|E|$, the total number of edges in the graph, $\mathcal{D}(G)$, the graph diameter, λ, the spectral gap of the transition matrix, and $d_{\max}$, the maximum degree of a node in the graph - so that IDEAL-WALK can determine the optimal number of steps t to walk. We rigidly prove that no matter what the target distribution is (barring certain extreme cases, e.g., when the distribution is 1 on the starting node and 0 on all others), IDEAL-WALK *always* outperforms its corresponding traditional random walk algorithm. Further, it also produce samples with absolutely zero bias (while random walks often cannot, depending on the graph topology). We also demonstrate through analysis of multiple theoretical graph models the significance of such efficiency enhancements for the sampling process.

Of course, IDEAL-WALK makes two unrealistic assumptions, which we remove through the design of Algorithms WALK and ESTIMATE, respectively. Algorithm WALK removes the assumption of access to global parameters by requiring access to only one parameters (besides the local neighborhood of the current node): $\bar{\mathcal{D}}(G)$, i.e., an upper bound on the diameter of the graph - which is often easy to obtain (e.g., it is commonly believed that 8 to 10 is a safe bet for real-world online social networks [3, 23]).

Algorithm ESTIMATE, on the other hand, estimates $p_t(v)$, i.e., the probability for Algorithm WALK to sample node v at Step t. To illustrate our main idea here, we start by developing UNBIASED-ESTIMATE, a simple algorithm which takes a *backward* random walk from Node v for estimating $p_t(v)$. We rigidly prove the unbiasedness of the estimation produced by UNBIASED-ESTIMATE. Nonetheless, we also note its problem: a high estimation variance which grows rapidly with the number of backward steps one has to take for producing the estimation. Since the error of estimation is determined by both bias and variance, the high variance produced by UNBIASED-ESTIMATE introduces significant error in the estimation of $p_t(v)$.

To address the problem of UNBIASED-ESTIMATE, we introduce two main ideas for variance reduction in developing Algorithm ESTIMATE, our final algorithm for estimating $p_t(u)$: One is *initial crawling*, i.e., the crawl of the h-hop neighborhood (where h is a small number like 2 or 3) of the starting node to reduce the number of backward steps required by Algorithm ESTIMATE, and the second is *weighted sampling*, i.e., to carefully design the transition matrix of the backward random walk process to reduce the variance of estimation. We shall demonstrate through experimental evaluation that Algorithm ESTIMATE significantly reduces the estimation variance for $p_t(u)$. Finally, we combine Algorithms WALK and ESTIMATE to produce Algorithm WALK-ESTIMATE.

1.4 Summary of Contributions

We make the following main contributions in this paper:

- We propose a novel idea of WALK-ESTIMATE, a swap-in replacement for any random walk sampler which forgoes the long burn-in period and instead uses a proactive sampling probability estimation step to produce samples of a desired target distribution.
- To demonstrate the superiority of our WALK step - i.e., performing a short random walk followed by acceptance-rejection sampling to reach the target distribution - we rigidly prove that, given a reasonable sample-bias requirement, no matter what the graph topology or target distribution is (barring certain extreme cases, e.g., when the distribution is 1 on the starting node and 0 on all others), a short random walk followed by acceptance-rejection sampling always outperforms its corresponding traditional random walk process.
- For the ESTIMATE step, we introduce a novel UNBIASED-ESTIMATE algorithm which uses a small number of queries to produce a provably unbiased estimation of the sampling probability of a given node. In addition, we also propose two heuristics, initial crawling and weighted sampling, to reduce the variance (and consequently error) of an estimation.
- Our contributions also include extensive experiments over real-world online social networks such as Google Plus, which confirm the significant improvement offer by our WALK-ESTIMATE algorithm over traditional random walks such as simple random walk and Metropolis-Hastings random walk.

1.5 Paper Organization

The rest of the paper is organized as follows. We discuss preliminaries in Section 2. Then, in Section 3, we present an overview of our WALK-ESTIMATE algorithm, and outline the key technical challenges for this design. In Sections 4 and 5, we develop the two main steps, WALK and ESTIMATE, respectively. We discuss in Section 6 two related issues: one is the application of our idea to another way of performing random walks, i.e., the "one-long-run" scheme. The other is the limitation of our techniques. We present the experimental results in Section 7, followed by a discussion of related work in Section 8 and the final remarks in Section 9.

2. PRELIMINARIES

2.1 Graph Model

In this paper, we consider online social networks with the underlying topology of an undirected graphs $G\langle V, E\rangle$, where V and E

are the sets of vertices and edges, respectively. Note that for online social networks which feature directed connections (e.g., Twitter), a common practice in the literature (e.g., [16]) is to reduce it to an undirected graph by defining two vertices $v_1, v_2 \in V$ to be connected in the undirected graph if and only if both $v_1 \rightarrow v_2$ and $v_2 \rightarrow v_1$ exist as directed connections. We use $|E|$ to denote the number of edges in the graph. For a given node $v \in V$, let $N(v)$ be the set of neighbors of v, and $d(v) = |N(v)|$ be its degree.

The web interface of the online social network exposes a restricted access interface which only allows *local neighborhood queries*. That is, the interface takes as input a node $v \in V$, and outputs $N(v)$, the set of v's neighbors. The objective of sampling, as mentioned in the introduction, is to generate a sample of V (according to a pre-determined sampling distribution) by issuing as few queries through the restrictive access interface as possible.

2.2 Traditional Random Walks

2.2.1 Overview

A random walk is a Markov Chain Monte Carlo (MCMC) method on the above-described graph G. Intuitively, all random walks share a common scheme: it starts from a *starting node* $v_0 \in V$. At each step, given the current node it resides on, say v_i for the i-th step, the random walk randomly chooses its next step from v_i's immediate neighbors and v_i itself (i.e., self-loop might be allowed) according to a pre-determined distribution, and then transits to the chosen node (one can see that $v_{i+1} \in \{v_i\} \cup N(v_i)$). We refer to this distribution over $N(v_i)$ as the *transit design*. As discussed below in examples of random walks, existing random walk designs often choose either a fixed distribution (e.g., uniform distribution), or a distribution determined by certain measurable attributes (e.g., degree) for nodes in $N(v_i)$. One can see that the transition design can be captured by a $|V| \times |V|$ transition matrix T, in which T_{ij} is the probability for the random walk to transit to node v_j if its current state is v_i.

Let $p_t(u)$ be the *sampling probability* for a node $u \in V$ to be taken at Step t of the random walk (i.e., $p_t(u) = \Pr\{u = v_t\}$). A special property of random walk, which makes it suitable for our purpose of node sampling, is that as long as the graph G is irreducible [10], $p_t(u)$ always converges to a fixed distribution when $t \rightarrow \infty$, no matter what the starting node v_0 is for the random walk.

2.2.2 Examples

There are many different types of random walks according to different designs of transition matrix T. We use *Simple Random Walk (SRW)* and *Metropolis-Hastings Random Walk (MHRW)* because of their popularity in the study of sampling online social networks.

DEFINITION 1. **(Simple Random Walk (SRW))**. *Given graph $G\langle V, E\rangle$, and a current node $u \in V$, a random walk is called Simple Random Walk if it uniformly at random chooses a neighboring node v from u's neighbors as the next step. The transition matrix T is*

$$T(u,v) = \begin{cases} 1/|N(u)| & \textit{if } v \in N(u), \\ 0 & \textit{otherwise.} \end{cases} \tag{1}$$

DEFINITION 2. **(Metropolis-Hastings Random Walk (MHRW))**. *Given graph $G\langle V, E\rangle$, and a current node $u \in V$, a random walk is called Metropolis-Hastings Random Walk if it chooses a neighboring node v according to the following transition matrix T:*

$$T(u,v) = \begin{cases} \frac{1}{|N(u)|} \cdot \min\{1, \frac{|N(u)|}{|N(v)|}\} & \textit{if } v \in N(u) \\ 1 - \sum_{w \in N(u)} T(u,w) & \textit{if } u = v \\ 0 & \textit{otherwise} \end{cases} \tag{2}$$

We note that we set the target stationary distribution as uniform distribution for MHRW.

2.2.3 Burn-In Period

With the traditional design of a random walk, its performance is determined by how fast the random walk converges to its stationary distribution, because only after so can the random walk algorithm takes a node as a sample. To capture this performance measure, *burn-in period* is defined as the number of steps it takes for a random walk to converge to its stationary distribution, as shown in the following definition.

DEFINITION 3. **(Relative Point-wise Distance)**. *Given graph $G\langle V, E\rangle$, and positive number of steps t, Relative Point-wise Distance is defined as the following distance between the stationary distribution and the probability distribution for nodes to be taken at Step t:*

$$\triangle(t) = \max_{u,v \in V, v \in N(u)} \left\{ \frac{|T^t_{uv} - \pi(v)|}{\pi(v)} \right\} \tag{3}$$

where T^t_{uv} is the element of T^t (transition matrix T to the power of t) with indices u and v, and π is the stationary distribution of the random walk [12]. The **burn-in period** *of a random walk is the minimum value of t such that $\triangle(t) \leq \epsilon$ where ϵ is a pre-determined threshold on relative point-wise distance.*

In practice, a popular technique for checking (on-the-fly) whether a random walk has reached its stationary distribution is called the *convergence monitors* (i.e. MCMC convergence diagnostics) [19]. For example, the Geweke method (summarized in [7]) considers two "windows" of a random walk with length l: Window A is formed by the first 10% steps, and Window B is formed by the last 50%. According to [30], if the random walk indeed converges to the stationary distribution after burn-in, then the two windows should be statistically indistinguishable. Let

$$Z = \left| \frac{\bar{\theta}_A - \bar{\theta}_B}{\sqrt{\hat{S}^A_\theta + \hat{S}^B_\theta}} \right|, \tag{4}$$

where θ is the attribute that can be retrieved from nodes (a typical one is the degree of a node), and $\bar{\theta}_A, \bar{\theta}_B$ are means of θ for all nodes in Windows A and B, respectively, and S^A_θ and S^B_θ are their corresponding variances. One can see that $Z \rightarrow 0$ when the random walk converges to the stationary distribution. We use Geweke method as the convergence monitor in the experiments, and we set the threshold to be $Z \leq 0.1$ by default, while also performing tests with the threshold $Z \leq 0.01$.

A property of the graph which has been proven to be strongly correlated with the burn-in period length is the *spectral gap* of the transition matrix T. We denote the spectral gap as $\lambda = 1 - s_2$ where s_2 is the second largest eigenvalue of T.

2.3 Acceptance-Rejection Sampling

Acceptance-rejection sampling (hereafter referred to as rejection sampling) is a technique we use to "correct" a sampling distribution to the desired target distribution. To understand how, consider the case where our algorithm samples a node with probability $p(u)$, while the desired distribution assigns probability $q(u)$ to node u. In order to make the correction, we take as input a node u sampled by our algorithm, and "accept" it as a real sample with probability

$$\beta(u) = \frac{q(u)}{p(u)} \cdot \min_{v \in V} \frac{p(v)}{q(v)}, \tag{5}$$

because after such correction, the probability distribution of node u in the final sample is

$$\frac{p(u) \cdot \beta(u)}{\sum_{u \in V}(p(u) \cdot \beta(u))} = q(u), \tag{6}$$

which conforms to the desired target distribution.

A practical challenge one often faces when applying rejection sampling is the difficulty of computing $\min_v p(v)/q(v)$, especially when the graph topology is not known beforehand. Even when a theoretical lower bound on $\min_v p(v)/q(v)$ can be computed, its value is often too small to support the practical usage of rejection sampling. A common practice to address this challenge is to replace $\min_v p(v)/q(v)$ with a manual threshold (e.g., [8, 9]). Note that a large threshold might introduce bias to the sample - e.g., a threshold greater than $\min_v p(v)/q(v)$ would make the computed $\beta(u) > 1$ for certain nodes, essentially under-sampling them in the final sample. Nonetheless, such a large threshold also improves the efficiency of sampling, as the rejection probability will be lower. Of course, such an approximation can be made more conservatively (i.e., lower) to reduce bias, or more aggressively (i.e., higher) to make the sampling process more efficient.

2.4 Performance Measures

There are two important performance measures for a sampling algorithm over an online social network. One is its query cost - i.e., the number of nodes it has to access in order to obtain a predetermined number of samples. Note that query cost is the key efficiency measures here because many website enforce a query rate limit on the number of nodes one can access from an IP address or API account for a given time period (e.g., a day). As such, a random walk based sampling algorithm has to minimize the number of steps it takes to generate samples.

The other key performance measures here is the sample bias - i.e., the distance between the actual sampling distribution (i.e., the probability distribution according to which each node is drawn as a sample) and a predetermined target distribution. Note that while the uniform distribution is often used as the target distribution to ensure equal chance for all nodes, the target distribution can also have other values - e.g., proportional to the node degree (when simple random walk is used).

Another important issue with the definition of sample bias is the distance measure being used. Traditionally (e.g., in the studies of burn-in period and convergence monitoring for bounding the sampling bias), a popular measure is the vector norm for the difference between the two probability distribution vectors. For example, the *variation distance* measure the ℓ_∞-norm of the difference vector - i.e., the maximum absolute difference for the sampling probability of a node. While this is a reasonable measure for theoretical analysis, it can be difficult to use for experimental evaluations, as obtain the actual sampling probability for every node requires running the sampling algorithm repeatedly for extremely large number of times, especially when the underlying graph is large. To address the problem, in this paper, we use the vector norm measure for theoretical analysis, while using a different measure for experiments for the large graphs: specifically, we measure the error while using the obtained sample to estimate AVG aggregates such as the average degree of all nodes in the graph. We shall further elaborate the design of this experimental measure and the various AVG aggregates we use in the experimental evaluation section.

3. OVERVIEW OF WALK-ESTIMATE

In this section, we provide an overview of WALK-ESTIMATE, our main contribution of the paper. Specifically, we first describe the input and output of the algorithm, followed by a brief description of our key ideas and an outline of the main technical challenges for WALK and ESTIMATE, respectively.

Input & Output: The design objective of WALK-ESTIMATE is to achieve universal speed-up for MCMC sampling (random walks) over online social networks regardless of their target sampling distribution (and correspondingly, transition design). To achieve this goal, WALK-ESTIMATE takes as input (1) the transition design of an MCMC sampling algorithm, and (2) the desired sample size h. The output consists of h samples taken according to the exact same target distribution as the input MCMC algorithm (subject to minimal sampling bias, as we shall further elaborate in latter sections). As discussed in Section 2, during this sampling process, WALK-ESTIMATE aims to minimize the query cost.

Key Ideas: Recall from the introduction that our main novelty here is to forgo the long "wait" (i.e., burn-in period) required by traditional random walks, and instead WALK an optimal (much smaller) number of steps (often only a few steps longer than the graph diameter - see below for details). Of course, having a drastically shorter walk also makes our sampling distribution different from the target one we have to achieve at the end. To address this problem, our WALK calls upon the ESTIMATE component to estimate the probability for a node to be sampled by a (now much shorter) walk. Not that such estimated probability allows us to perform acceptance-rejection sampling [21] over the nodes sampled in WALK, which eventually leads to samples taken according to the target distribution.

Technical Challenges for WALK: One can see from the above description that the design of the two components face different challenges: For "WALK", the main challenge is how to properly determine the number of steps to walk. Clearly, the walk length must be at least the diameter of the graph in order to ensure a positive sampling probability for each node. On the other hand, an overly long walk not only diminishes the saving of queries, but might indeed cost even more queries than traditional random walks when the cost of ESTIMATE is taken into account. We shall address this challenge in Section 4 - and as we shall further elaborate there, fortunately, for real-world social networks, there is usually a wide range of walk lengths with which the WALK step can have a significant saving of query cost even after rejection sampling.

Technical Challenges for ESTIMATE: For ESTIMATE, the key challenge is how to enable an accurate estimation for the sampling probability of a node without incurring a large query cost. Note that, after we repeatedly run WALK to generate (say 100) samples, there may be nodes sampled multiple times by WALK for which we can directly estimate their sampling probability (as their relative frequency within the collected sample). Nonetheless, for the vast majority of nodes which are sampled only once (almost always the case when the graph being sampled is large), it is unclear how one can estimate their sampling probabilities. We shall address this challenge in Section 5 and show that (1) there is a surprisingly simple algorithm which enables a completely unbiased estimation of the sampling probability and consumes only a few extra queries, and (2) there are two effective heuristics which reduce the estimation variance even further, leading to more accurate estimations.

In the next two sections, we shall develop or techniques for the two components, WALK and ESTIMATE, respectively. The combination of them forms Algorithm WALK-ESTIMATE which, as we demonstrate in the extensive experimental results in Section 7, produces higher quality (i.e., lower bias) samples than the traditional random walk algorithms while consuming fewer queries.

4. WALK

We start with developing Algorithm WALK which significantly improves the efficiency of sampling by having a much shorter random walk followed by a rejection sampling process. Note that, for the ease of discussions, we separate out the discussion of sampling-probability estimation to the ESTIMATE component discussed in the next section - i.e., Algorithm WALK calls upon Algorithm ESTIMATE as a subroutine.

In this section, we first illustrate the key rationale behind our design with an ideal-case algorithm, IDEAL-WALK, and then present theoretical analysis which shows that a shorter walk followed by an acceptance-rejection procedure can almost always outperform the traditional random walk, no matter what the starting point is or the graph topology looks like. To study how much improvement a short walk can offer, we describe case studies with the underlying graph generated from various theoretical graph models. Finally, we conclude this section with the practical design of Algorithm WALK.

4.1 IDEAL-WALK: Main Idea and Analysis

The key rationale behind our idea of performing a short walk followed by acceptance-rejection sampling can be stated as follows. Recall from Section 2 that the long walk is required by traditional random walks to reduce the "distance" between its sampling distribution and the target, stationary, distribution - a distance often measured by the difference (e.g., ℓ_∞-norm) between the two probability vectors.

Consider how such a difference changes as the walk becomes longer. When the walk first starts, the sampling distribution is extremely skewed - i.e., $p_1(v) = 1$ on one node (the starting one) and 0 on all others - leading to an extremely large distance. As the walk proceeds, the distance decreases quickly - for example, as long as the walk length exceeds the graph diameter, all values in the sampling probability vector become positive[1], while the maximum value in the vector tends to decrease exponentially with the (initial few) steps taken - leading to a sharp decrease of the distance.

Nonetheless, it is important to note that the speed of reduction on the distance becomes *much slower* as the random walk grows longer. A simple evidence is the asymptotic nature of burn-in as discussed in Section 2 - which shows that, for some graphs, the ultimate reduction to zero distance never completes with a finite number of steps. Figure 1 demonstrates a concrete example for a random scale free network with 31 nodes generated by the Barabasi-Albert model [4], where number of edges to attach from a new node to existing nodes is 3. One can see from the figure that the speed of reduction declines sharply once the random walk grows longer than the graph diameter. In summary, one can observe the following "behavior pattern" of traditional random walks: to achieve a preset goal of shrinking the distance measure below a threshold, the random walk makes significant progress in the first few steps. Nonetheless, the "benefit-cost ratio" diminishes quickly as the random walk continues. As a result, a random walk might require a very long burn-in period to achieve the preset distance threshold.

Standing in sharp contrast to the above described behavior pattern is the performance of using acceptance-rejection sampling to achieve the pre-determined target distribution (instead of waiting for convergence). Interestingly, applying rejection sampling at the beginning of a random walk is often extremely costly - or even outright infeasible. For example, no rejection sampling can correct to a uniform target distribution before the walk is at least as long as the graph diameter. On the other hand, as the walk becomes longer, the cost of applying rejection sampling to reach the target distribution becomes much smaller - as we shall demonstrate as follows.

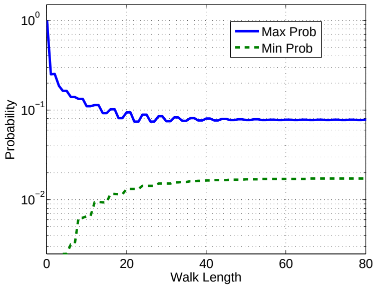


Figure 1: Minimum and maximum probabilities vs walk length

Consider an example where the target distribution is uniform. Note from acceptance-rejection sampling in Section 2.3 that, in this case, the cost of rejection sampling is simply determined by (to be exact, inversely proportional to) the minimum value in the input sampling distribution - as the probability of accepting a sample is exactly the minimum probability multiplied by the number of nodes in the graph. As discussed above, this minimum probability grows from 0 at the very beginning to a positive value when the walk reaches the diameter, and often increases rapidly at the initial stage of random walk (see again Figure 1). Correspondingly, the cost of rejection sampling drops significantly with a longer random walk.

One can observe from the above discussion an interesting distinction between two (competing) methods, (a) wait for the sampling distribution to converge to the stationary one, and (b) taking the current sampling distribution and directly "correct" it through rejection sampling: These two methods are better applied at different stages of a random walk process. Specifically, at the very beginning, method (b) is extremely costly or outright infeasible - so we should follow method (a) and walk longer for the sampling distribution to grow closer to the target vector. Nonetheless, after a certain number of steps, the direct correction (i.e., method (b)) becomes the better option because of the slower and slower convergence speed. Therefore, we should stop waiting for further convergence, and instead use rejection sampling to directly reach the desired distribution.

Of course, the above discussions leaves an important question unanswered: Given a reasonable threshold on the distance (between achieved sampling distribution and the desired stationary one), is there always a tipping point where we switch for waiting to correction? Note that the reason why the threshold value is important here can be observed from the extreme cases: when the threshold is extremely large, there is no need to switch because even the initial (one 1 and all other 0) distribution already satisfies the threshold. On the other hand, when the threshold trends to 0, as we discussed above, there are graphs for which the convergence length tends to infinity - i.e., it is always better to switch to rejection sampling as long as it has a finite cost. One can see from the extreme cases that whether switching to rejection sampling is effective in practice depends on whether the switch is necessary for *reasonable* thresholds that are just small enough to support real-world applications over the samples taken from online social networks. To this end, we have the following theorem.

THEOREM 1. *Given an input random walk which has a transition design with spectral gap λ, to guarantee an ℓ_∞-variation distance of Δ between the sampling and target distributions, the expected query cost per sample of IDEAL-WALK which performs the*

[1] Note that here we assume each node has a nonzero (can be arbitrarily small) probability to transit to itself, to eliminate trivial cases where the graph is not irreducible.

random walk for t_{opt} steps followed by rejection sampling, where

$$t_{\mathrm{opt}} = \frac{-\log(-\frac{1}{\Gamma} \cdot W(-\frac{\Gamma}{ed_{\max}}) \cdot d_{\max})}{\log(1-\lambda)}, \tag{7}$$

where $d_{\max}$ is the maximum degree of all nodes in the graph and W is the Lambert W-function, is always smaller than that of the the input random walk as long as $0 < \Delta < \Gamma$. Specifically, the ratio between the query cost per sample of IDEAL-WALK and the input random walk is at most

$$\begin{aligned} &\frac{\Gamma \cdot t_{\mathrm{opt}} - t_{\mathrm{opt}} \cdot \Delta}{\Gamma - (1-\lambda)^{t_{\mathrm{opt}}} \cdot d_{\max}} \\ \leq &\frac{-\log(-\frac{1}{\Gamma} \cdot W(-\frac{\Gamma}{ed_{\max}}) \cdot d_{\max})}{\log(\Delta/d_{\max})} \cdot \frac{\Gamma - \Delta}{\Gamma + \frac{\Gamma}{W(-\frac{\Gamma}{ed_{\max}})}}. \end{aligned} \tag{8}$$

PROOF. The proof has been provided in technical report [25]. □

This Theorem shows that how the performance of IDEAL-WALK changes when the walk length grows larger. Initially, a larger number of steps leads to a smaller cost, i.e., the expected query cost per sample for IDEAL-WALK, until number of steps reaches the optimal value t_{opt}. Afterwards, a larger number of steps will lead to a larger cost, until cost of the IDEAL-WALK becomes equivalent with the cost of input random walk. To understand the concrete values of optimal number of steps, t_{opt}, and the ratio between the query cost per sample of IDEAL-WALK and the input random walk in theorem, we consider a number of case studies in the following subsection.

4.2 Case Study

In this subsection, we compute numerically the values of t_{opt} and c/c_{RW} over a number of theoretical graph models, specifically Cycle, Hypercube, Barbell, Tree, and Barbasi-Albert (scale free) models: A cycle graph consists of a single circle of n nodes - i.e., the graph has a diameter of $\lfloor \frac{n}{2} \rfloor$. A k-hypercube consists of 2^k nodes and $2^{k-1}k$ edges. If we represent each node as a (unique) k-bit binary sequence, and two nodes are connected if and only if their representations differ in exactly one bit. One can see that the hypercube has a diameter of k. A barbell graph of n nodes is a graph obtained by connecting two copies of a complete graph of size $\frac{n-1}{2}$ by a central node, i.e, the diameter is 3. A tree of height h is a cycle free graph with at most $2^{h+1} - 1$ nodes and diameter of $2h$. We considered a balanced binary tree, where the leaves are h hops away from the root. Finally, to simulate a scale-free network (with node degrees following a power-law distribution), we use the Barabasi-Albert model [4], where number of edges to attach from a new node to existing nodes is 3.

Figure 2 depicts how the expected query cost per sample changes when the length of walk taken by IDEAL-WALK varies from 1 to 128. We considered graphs with fix number of nodes 31. Since hypercube should have 2^k nodes, we generate the one with 32 nodes. In all cases, the target distribution is the uniform distribution. Note that if the walk length is smaller than the corresponding graph diameter, then we cost c to be infinity. One can see from the figure that, for all graph models, the trend we observe from Theorem 1 holds - i.e., the query cost per sample c drops dramatically at the beginning, reaches a minimal value, and then increases slowly. Another observation from the figure is that, in general, the larger diameter a graph has, the greater the optimal walk length for IDEAL-WALK will be. For example, compared with a Barbell graph with diameter of 3, the cycle graph with diameter of $\lfloor \frac{31}{2} \rfloor = 15$ has a much longer optimal walk length - and consequently requires a larger query cost per sample.

Next, we examine the degree of improvement offered by IDEAL-WALK over the input random walk, again over the various graph models described above. Figure 3 depicts how the ratio of improvement - i.e., $1 - c/c_{\mathrm{RW}}$ - changes when the graph size varies from 4 to 128. There are two interesting observations from the figure: One is that, while IDEAL-WALK offers over 50% savings in almost all cases, the amount of savings does depend on the underlying graph topology - e.g., the improvement ratio is far smaller on cycle graphs than others, mainly caused by its large diameter and small spectral gap of $O(n^{-2})$ [19].

The other observation is on how the improvement ratio changes with graph size: Interesting, when the graph becomes larger, the ratio increases for some models (e.g., Barbell), remains virtually constant for some others (e.g., hypercube, Barabasi-Albert), and declines for the ones left (e.g., cycle). An intuitive explanation here is that how the improvement ratio changes, as predicted in Theorem 1, depends on a joint function of the graph size (e.g., $|E|$) and the spectral gap (i.e., λ). Since the spectral gap is difficult to directly observe, and there is a common understanding that the spectral gap is negatively correlated with the graph diameter [28], we illustrate the issue here by considering how the graph diameter changes with a linearly increasing node count for the various graph models: For the cycle graph, the diameter increases as fast as the node count - leading to a (generally) decreasing improvement ratio. For hypercube, tree and Barabasi-Albert models, the diameter increases at the log scale[2] of node count - correspondingly, the improvement ratio is almost unaffected by the graph size. For Barbell graph, on the other hand, the diameter remains unchanged (i.e., 3) no matter how large the graph is. As a result, we observe a rapidly increasing improvement ratio from Figure 3. Note that this is indeed a promising sign for the performance of IDEAL-WALK over real-world social networks, because it is widely believed that the diameter of such a network remains virtually constant (e.g., [23] [14]) no matter how large the graph size is - in other words, the improvement ratio offered by IDEAL-WALK is likely to increase as the graph becomes larger - a phenomenon we shall verify in the experiments section over the synthetic social networks by changing the size of the graph.

4.3 Algorithm WALK

While the above theoretical analysis demonstrates the significant potential of query-cost savings by our WALK-ESTIMATE scheme, there is one key issue remaining before one can instantiate our idea into a practical WALK algorithm: in practice when the graph topology is not known beforehand, how can we determine the number of steps to walk before calling the ESTIMATE algorithm and performing the rejection sampling process? As one can see from the above discussions, an overly small length would lead to most samples being rejected, while an overly large one would incur unnecessary query cost for the WALK step.

Fortunately, we found through studies over real-world data (more details in the experimental evaluation section) that the setting of walk length is usually easy in practice as long as we set the walk length conservatively rather than aggressively. To understand why, note from the above case study, specifically the change of query cost per sample with walk length, that the query cost drops sharply before reaching the optimal walk length, the increase afterwards is much slower. As such, a reasonable strategy for setting the walk length is to be conservative rather than aggressive - i.e., giving preference to a longer, more conservative walk length. As we shall further elaborate in the experiments section, we use a default walk

[2] To be exact, the diameter for Barabasi-Albert model is proportional to $\log n / \log \log n$ [6] [5]

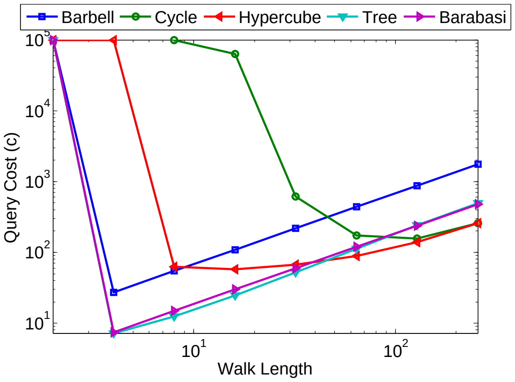


Figure 2: Query cost per sample achieved by IDEAL-AR-SAMPLER.

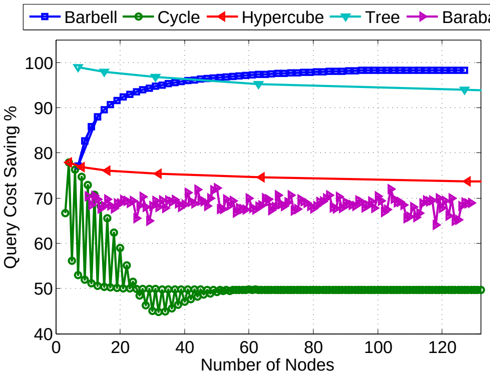


Figure 3: Query cost saving of the IDEAL-AR-SAMPLER.

length of two times the graph diameter, which is conservatively estimated to be 10 for real world online social networks.

It is important to note that, while our experiments demonstrate that the above described heuristic strategy for setting the walk length works well over real world social networks, it is not a theoretically proven technique that works for all graphs. A simple counterexample here is the above-discussed Barbell graph - i.e., two complete graphs connected by one node, with one edge connected to each half. One can see that, while the graph has a very short diameter (i.e., 3), a random walk of length 6 is highly unlikely to cross to the other half of the graph, unless it starts from one of the three nodes that connect the two halves together. As such, the above heuristics for setting the walk length would yield an extremely small sample-acceptance probability and, therefore, a high query cost.

5. ESTIMATE

Algorithm WALK leaves as an open problem of the estimation of sampling probability for a given node, so as rejection sampling can be applied to reach the input target distribution. In this section, we address this problem with Algorithm ESTIMATE. Specifically, we shall first describe a simple algorithm which, somewhat surprisingly, provides a completely unbiased estimation for the sampling probability with just a few queries. Unfortunately, we also point out a problem of this simple method: its high estimation variance which, despite the unbiasedness, still leads to a large error. To address this problem, we develop two heuristics, initial crawling and weighted sampling, to significantly reduce the estimation variance while requiring only a small number of additional queries.

5.1 UNBIASED-ESTIMATE

Unbiased Estimation of Sampling Probability: Recall that we use $p_t(u)$ to denote the probability for a node u to be visited at Step t of a random walk conducted by WALK, and $N(u)$ is the set of neighbors of u. To illustrate the key idea of UNBIASED-ESTIMATE, we start by considering the case where the input random walk is the simple random walk. One can see that

$$p_t(u) = \sum_{u' \in N(u)} \frac{p_{t-1}(u')}{|N(u')|}. \tag{9}$$

Thus, given u, a straightforward method of estimating $p_t(u)$ is to select uniformly at random a neighbor of u (i.e., $u' \in N(u)$), so as to reduce the problem of estimating $p_t(u)$ to estimating $p_{t-1}(u')$, because an unbiased estimation of $p_t(u)$ is simply

$$\tilde{p}_t(u) = \frac{|N(u)|}{|N(u')|} \cdot p_{t-1}(u'). \tag{10}$$

An important property of such an estimation is that as long as we can obtain an unbiased estimation of $p_{t-1}(u')$, say $\tilde{p}_{t-1}(u')$, then the estimation for $p_t(u)$ will also be unbiased. The reason for the unbiasedness can be stated as follows: Note that due to the conditional independence of the estimation of $\tilde{p}_{t-1}(u')$ with the selection of u' from $N(u)$, we have

$$E(\tilde{p}_t(u)) = \sum_{u' \in N(u)} \frac{1}{|N(u)|} \cdot \frac{|N(u)|}{|N(u')|} \cdot E(\tilde{p}_{t-1}(u')) \tag{11}$$

$$= \sum_{u' \in N(u)} \frac{1}{|N(u')|} \cdot E(\tilde{p}_{t-1}(u')) \tag{12}$$

$$= \sum_{u' \in N(u)} \frac{1}{|N(u')|} \cdot p_{t-1}(u') = p_t(u), \tag{13}$$

where $E(\cdot)$ represents the expected value taken over the randomness of the estimation process.

Given the unbiasedness property, we can run a recursive process for estimating $p_t(u)$ (with a decreasing subscript t) until reaching $p_0(w)$. Now we have $p_0(w) = 1$ if w is the starting node of the random walk and 0 otherwise. One can see that this recursive process leads to an unbiased estimation of $p_t(u)$. We refer to this estimation method as UNBIASED-ESTIMATOR. The generic design of UNBIASED-ESTIMATOR (for any input MCMC random walk) is depicted in Algorithm 1, where $p_{uu'}$ is the transition probability from node u' to u and $q_{u'u}$ is the transition probability from node u to u' in the backward process.

Algorithm 1 UNBIASED-ESTIMATE

```
Input: Node $u$, Starting node $w$, Length of walk $t$
If $t = 0$ and $u == w$ then return 1
If $t = 0$ and $u$ != $w$ then return 0
return $\frac{p_{uu'}}{q_{u'u}}$ · UNBIASED-ESTIMATE($u'$, $w$, $t - 1$)
```

Analysis of Estimation Variance: While the above UNBIASED-ESTIMATOR produces an unbiased estimations of the sampling probability, it also has an important problem: a high estimation variance which leads to a high estimation error (unless the estimator is repeatedly executed to reduce variance - which would lead to a large query cost nonetheless). Specifically, the estimation variance on the last few steps of the recursive process (i.e., with the smallest subscript in $p_t(u)$) are amplified significantly in the final estimation. To see this, consider a simple example of a k-regular graph. With UNBIASED-ESTIMATOR, the estimation of $p_t(u)$ is either $\tilde{p}_t(u) = 1$ (when the node w encountered at $p_0(w)$ is the staring node) or 0 otherwise. As a result, the relative standard error for the estimation of $p_t(u)$ is exactly $\sqrt{(1 - p_t(u))/p_t(u)}$. Since $p_t(u)$ is usually extremely small for a large graph, the relative standard error can be very high for UNBIASED-ESTIMATOR.

Our main idea for reducing the estimation variance is two-fold: *initial crawling* and *weighted sampling* - which we discuss in the next two subsections, respectively, before combing UNBIASED-ESTIMATE and the two heuristics to produce the practical ESTIMATE algorithm.

5.2 Variance Reduction: Initial Crawling

Our first idea is to *crawl* the h-hop neighborhood of the starting point, so for each node v in the neighborhood, we can precisely compute its sampling probability $p_t(v)$ for $t \leq h$. For example, if simple random walk is used in WALK, then all nodes v in the immediate 1-hop neighborhood of starting node s have $p_1(v) = \frac{1}{|N(s)|}$. In practice, h should be set to a small number like 2 or 3 to minimize the query cost caused by the crawling process - note that the query cost is likely small because many nodes in the neighborhood may already be accessed by the WALK part, especially when multiple walks are performed to obtain multiple samples.

One can see that, with this initial crawling step, we effectively reduce the number of backward steps required by ESTIMATE because the backward estimation process can terminate as soon as it hits a crawled node. This shortened process, in turn, leads to a lower estimation variance and error.

5.3 Variance Reduction: Weighted Sampling

Our second idea for variance reduction is weighted sampling - i.e., instead of picking u' uniformly at random from $N(u)$ as stated above (for estimating $p_t(u)$ from $p_{t-1}(u')$), we design the probability distribution based on the knowledge we already have about the underlying graph (e.g., through the random walks and backward estimations already performed). The key motivation behind this idea is the following observation on UNBIASED-ESTIMATE: When estimating $p_t(u)$, the values of $p_{t-1}(u')$ for all neighbors of u (i.e., $u' \in N(u)$) tend to vary widely - i.e., some neighbors might have much higher sampling probability than others. This phenomenon is evident from the fact that, even after reaching the stationary distribution of say the simple random walk, the sampling probability can vary by $d_{\max}/d_{\min}$ times, where $d_{\max}$ and $d_{\min}$ are the maximum and minimum degree of a node, respectively. On the other hand, without the initial crawling process, when $t = 1$, all but one neighbors of u have $p_0(u') = 0$, while the other one has $p_0(u') = 1$ - also a significant variation.

Given this observation, one can see that we should allocate the queries we spend according to the value of $p_{t-1}(u')$ rather than simply at a uniform basis - specifically, we should spend more queries estimating a larger $p_{t-1}(u')$, simply because its estimation accuracy bears more weight on the final estimation error of $p_t(u)$. To this end, we adjust the random selection process of u' from $N(u)$ to the following *weighted sampling* process: First, we assign a minimum sampling probability ϵ to all nodes in $N(u)$ - to maintain the unbiasedness of the estimation algorithm. For the remaining $1 - \epsilon$ probability, we assign them proportionally to the total number of historic random walks which hit node u' at Step $t - 1$. More specifically, during the estimation process, all our random walks start from the same starting node. Suppose we have performed n_{hw} random walks and currently performing the next one. Also suppose that we are at node u at step t. Let u' be a neighbor of u (i.e $u' \in N(u)$). Among the n_{hw} random walks, we compute the number of times u' is reached at step $t - 1$. Let it be $n_{u',t-1}$. i.e. $0 \leq n_{u',t-1} \leq n_{hw}$. The ratio $\frac{n_{u',t-1}}{n_{hw}}$ has some impact on how often node u' is picked as part of the random walk. Algorithm 2 depicts the pseudocode of this weighted sampling scheme.

Algorithm 2 WeightedSamplingBackward (WS-BW)

```
Input: Node $u$, starting node $w$, Length of walk $t$, $\epsilon$
if $t = 0$ and $u = w$ then return $1$
if $t = 0$ and $u \neq w$ then return $0$
$\forall u' \in N(u), \pi_{u'} = \epsilon/|N(u)|$
$\forall u' \in N(u), \pi_{u'} = \pi_{u'} + (1 - \epsilon)(n_{u',t-1}/n_{hw})$
Choose node $v$ from $N(u)$ according to distribution $\pi$
return $\frac{1}{|N(v)| \cdot \pi_v} \cdot$ WS-BW$(v, w, t - 1)$
```

5.4 Algorithm ESTIMATE

We now combine UNBIASED-ESTIMATE with our two heuristics for variance reduction, initial crawling and weighted sampling, to produce Algorithm ESTIMATE. Note that there is one additional design in ESTIMATE which aims to further reduce the estimation error: For each $p_t(u)$ we need to estimate, we can repeatedly execute ESTIMATE (and take the average of estimations) to reduce the estimation error. The number of executions we take, of course, depends on the overall query budget. In addition, instead of running the same number of executions for all u, we next allocate the budget, once again, according to the estimations we have obtained so far for all nodes to be estimated. Specifically, we assign the number of executions in proportion to the estimation variance for each node. Figure 3 depicts the pseudo code of Algorithm ESTIMATE.

Algorithm 3 ESTIMATE

```
Input: Starting node $w$, length of walk $t$, number of crawling
    steps $h$, forward random walks issued $F$
Crawl $h$-hop neighborhood of $w$ and compute their exact sam-
    pling probability
Let $V_F$ be the set of nodes hit by random walks in $F$
for $u \in V_F$ do
    $p_t(u)$ = WS-BW$(u, w, t)$
    Compute estimation variance of estimations of $p_t(u)$
end for
Use remaining budget to reduce variance by invoking Algo-
    rithm 2. Choose nodes randomly proportional to their variance.
```

6. LIMITATIONS OF WALK-ESTIMATE - GRAPH DIAMETER

Before concluding the technical discussion of WALK-ESTIMATE, we would like to point out its limitations - i.e., when it should *not* be applied for sampling a graph with local neighborhood access limitations. Specifically, we note that WALK-ESTIMATE should not be applied over graphs with long diameters. Note that while our results for IDEAL-WALK (Theorem 1) appear to demonstrate efficiency enhancements regardless of the graph diameter, the performance of our ESTIMATE step is significantly worse when the graph diameter is large. The reason behind this is straightforward

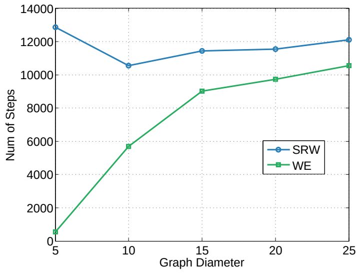


Figure 4: Cycle graphs with long diameter

- in our backward walk for ESTIMATE, the probability of hitting the starting node (or the starting neighborhood crawled by the initial crawling process) decreases rapidly when the graph diameter becomes larger. This in turn leads to more backward walks being required for the estimations of sampling probability and, as a result, worst sampling efficiency. Figure 4 demonstrates an example of how the average number of walk steps taken (for WALK-ESTIMATE, both forward and backward) per sample changes on cycle graphs for simple random walk (SRW) and our WALK-ESTIMATE algorithm (WE, with SRW as input) when the graph diameter grows from 5 to 25. The cycle graphs' sizes are $11, 21, \ldots, 51$. One can see that unlike SRW which is barely affected by the growing diameter, the expected cost of WALK-ESTIMATE increases dramatically as the diameter becomes longer. It is important to understand that graphs with long diameters are *not* the intended target of this paper, because it is well known that online social networks, even the very large ones, have small diameters ranging from 3 to 8 [20].

7. EXPERIMENTAL EVALUATION

7.1 Experimental Setup

Hardware and Platform: All our experiments were performed on a quad-core 2 GHz AMD Phenom machine running Ubuntu 14.04 with 8 GB of RAM. The algorithms were implemented in Python.

Datasets: In this section, we test real-world data crawled from online social networks and also those which are publicly available. Specifically, we use three different popular social graphs, i.e. Google Plus, Yelp, and Twitter. The detail of each dataset is described bellow. Moreover, we use small synthetic data to find the exact bias of the obtained samples.

Google Plus Social Graph: Google Plus[3] is the second largest social networking site with more than 500 million active users. For our experiments, we crawled a subset of the graph by starting from a number of popular users and recursively collecting information about their followers. We model this dataset as an undirected graph where the users correspond to nodes and an edge exist between two users if at least one of them has the other in their circles. We collected 16,405 users with more than 4.5 million connections between them. The average degree of the graph is 560.44. We also collected each user's self description and used it in our tests.

Yelp Social Graph: Yelp is a crowd-sourced local business review and social networking site with 132 million monthly visitors and 57 million reviews. Yelp Academic dataset[4] provides all the data and reviews of the 250 local businesses. For our experiments, we considered the largest connected component of user-user graph where nodes are the users and an edge exists between two users if they review atleast one similar business. Moreover, for each user there exists different information such as, review text, star rating, count of useful votes, count of funny votes, and count of cool votes. This graph has approximately 120,000 nodes and more than 954,000 edges.

Twitter Social Graph: Twitter is an online social network which is popular among millions of users who generate huge numbers of tweets, posts, and reviews every day. We used the Twitter dataset from Stanford's SNAP dataset repository[5] which is a directed graph crawled from public sources and has close to 80,000 nodes and more than 1.7 million edges. In our experiments, we assume it is an undirected graph and we consider in and out degrees of a node as its attributes.

Synthetic Graph: We generated scale-free network of size 1000 nodes and 6951 edges using the Barabasi-Albert model [4], with the number of edges to attach from a new node (to existing nodes) set by default to 7.

Algorithms Evaluated: We evaluated two traditional random walks - simple random walk (SRW) and Metropolis Hastings random walk (MHRW) - and the application of our WALK-ESTIMATE (WE) algorithm over each of them. Additionally, in order to evaluate the effect of the variance reduction heuristics, initial crawling and weighted sampling, proposed in Section5, we compared the performance of our main algorithm (WE) with three variations WE-None, WE-Crawl, and WE-Weighted. WE-None uses neither heuristics, WE-Crawl uses initial crawling only, while WE-Weighted uses weighted sampling only.

Parameter Settings: For SRW and MHRW, we used the Geweke convergence monitor [7] with threshold $Z \leq 0.1$. For our WALK component, we set the walk length to $2d + 1$ where d is the (estimated) graph diameter (set to $d = 7$ for Google Plus). For initial crawling, we set $h = 1$ for Google Plus and $h = 2$ for the synthetic graphs, Yelp, and Twitter. For weighted sampling, we set $\epsilon = 0.1$. We also considered 10th percentile of the estimation of sampling probabilities as the scale factor, $\min_{v \in V}(p(v)/q(v))$, for the acceptance rejection. In each graph, we run all the algorithms from the same starting point. For each obtained data point in the results we reported average value of the 100 runs. In WE, we varies number of walks from 100 to 2000 in every run.

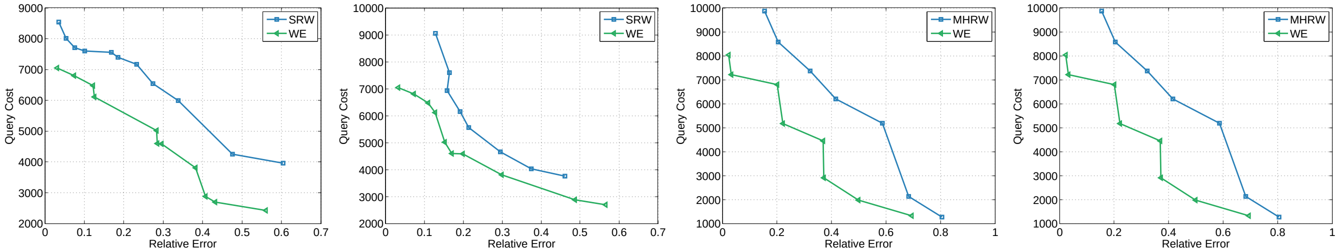
Performance Measures: Given the large sizes of the graph being tested, it is impractical to precisely measure the bias of obtained samples. Thus, for the large graphs we indirectly measured the sample bias by the relative error of AVG aggregate estimations generated from the samples (i.e., $|\tilde{x} - x|/x$ where x and $\tilde{x}$ are the precise and estimated values of the aggregate, respectively). We used arithmetic and harmonic mean for the uniform and non-uniform samples respectively. Node and edge sampling has been used to measure variety of network metrics [11, 26, 27]. Among them we focus on the measures which can be computed from node sample. Specifically, we evaluate AVG aggregate of the measures related to the topological properties (such as degree, shortest path length, local clustering coefficient) as well as measures associated with a node attribute (such as number of stars in Yelp, and user's self-description in Google Plus). Specifically, for Google Plus, we considered two aggregates: the AVG degree and the AVG number of words in a user's self-description. For Yelp and Twitter we considered topological properties, i.e., degree, shortest path length, local clustering coefficient. In Twitter we estimate number of followers and followees using the in and out degrees of the node as its attributes. We also estimate average number of stars in Yelp.

Moreover, we computed exact bias of our algorithms over a small graph. Recall that sample bias is defined as the distance between

[3] http://plus.google.com

[4] www.yelp.com/academic_dataset

[5] snap.stanford.edu/data/egonets-Twitter.html



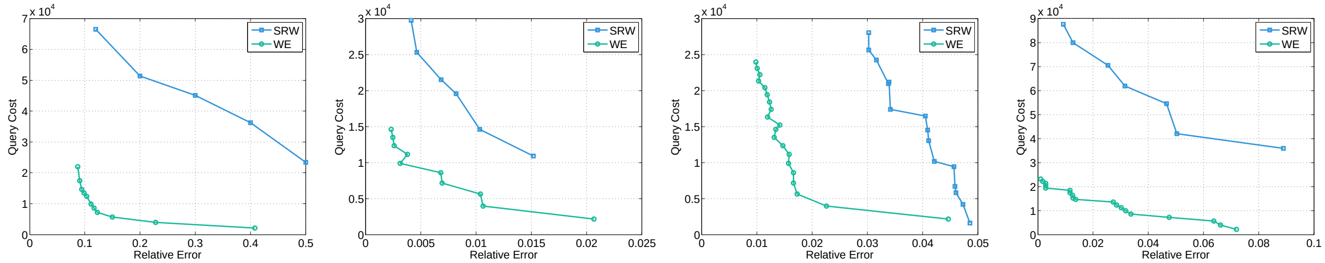
(a) Average Degree (SRW)

(b) Average Self-description Length (SRW)

(c) Average Degree (MHRW)

(d) Average Self-description Length (MHRW)

Figure 5: Relative Error of the Average Estimations vs Query Cost in Google Plus.



(a) Yelp: Average Degree

(b) Yelp: Average Stars

(c) Yelp: Average Shortest Path

(d) Yelp: Average Local Clustering Coefficient

Figure 6: Relative Error of the Average Estimations vs Query Cost in Yelp.

actual sampling distribution (i.e., the probability distribution according to which each node is drawn as a sample) and a predetermined target distribution. We used two distance measures variation distance (ℓ_∞), and K-L divergence to quantify the bias.

7.2 Experimental Results

Aggregate estimation: We started by testing how WE performs against the baseline SRW and MHRW on the fundamental tradeoff in social network sampling - i.e., sample bias vs. query cost. The results over Google Plus are shown in Figure 6. Specifically, subgraphs (a) and (b) depict SRW and WE with SRW as input random walk, while (c) and (d) are corresponding to MHRW. The AVG aggregate used to measure sample bias is AVG degree for (a) and (c) and AVG self-description length for (b) and (d). As one can see from the figure, our algorithm significantly outperforms SRW and MHRW - i.e., offers substantially smaller relative error for the same query cost - on both aggregates tested. Figure 6 shows the results over the Yelp dataset. Specifically, subgraph (a) shows the AVG aggregate of the node attributes, i.e. star rating while subgraphs (b), (c), and (d) is the results of AVG aggregate of the topological properties, i.e, degree, shortest path length, and clustering coefficient. The results confirm the fact that WE provides smaller relative error with the same query cost. We also test our algorithm in Twitter dataset and the results in Figure 7 shows that AVG of the in-degree, out-degree, shortest path length, and clustering coefficient of the samples retrieved by the WE has smaller relative error than SRW for the same query cost.

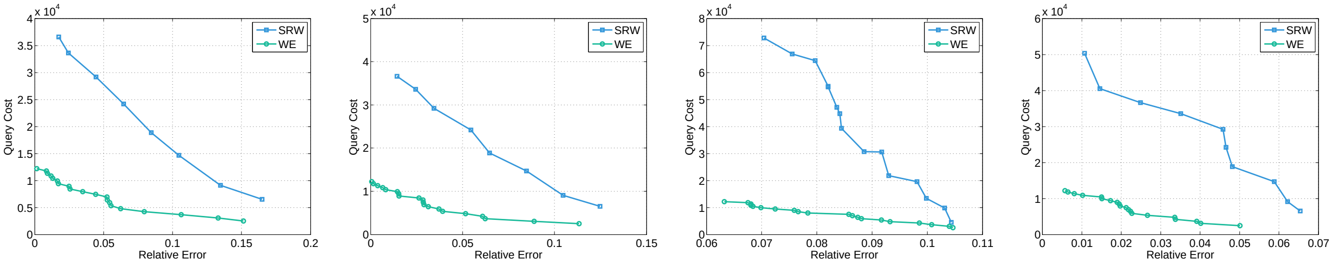
We also study how our proposed variance reduction techniques improves the efficiency of our algorithm by comparing the performance of WE, WE-None, WE-Crawl and WE-Weighted, again according to how the relative error of aggregate estimation changes with the query cost. Figure 8 depicts the result for Google Plus, according to the same subgraph setup (i.e., random-walk/aggregate combination) as Figure 6. One can see that, as expected in all cases, WE outperforms the single-heuristics variants, which in turn outperform the theoretical variant of the algorithm.

Finally, we tested the quality of samples obtained by WE, in order to verify that the above-tested performance enhancements are not merely from walks being shorter, but from an equal or higher quality sample as well. To this end, Figure 8 depicts how the relative errors on AVG estimations change with the number of samples produced by SRW, MHRW, and the corresponding WEs, respectively on Google Plus - again according to the same subgraph setup as Figure 6. One can see that in all cases, the samples produced by WE achieves smaller relative error than the corresponding input random walks (with Geweke convergence monitor), indicating the smaller sample bias achieved by WE. The results for the Yelp and Twitter are similar to those of Google Plus and, due to space limitations, are not included in the paper. More experimental results can be found in the technical report [25].

Exact bias: We used a synthetic graph to find the exact bias of the obtained samples. We compared three different sampling distributions: (1) theoretical target distribution denoted as *theo* (2) WE sampling distribution and (3) SRW sampling distribution. We run the sampling algorithm with a large query budget 1,000,000 for both WE and SRW, we got 36600 samples from WE and 1101 samples from SRW. The *Probability density function (PDF)* is then estimated based on these samples, see Figure 10. One can see that WE produces more accurate PDF than SRW (i.e. closer to the theoretical PDF curve). Table 1 provides the result of variation distance (ℓ_∞) and K-L divergence to quantify the bias of the samples. The results show that the sampling distribution of WE is much closer than that of SRW to the theoretical target distribution.

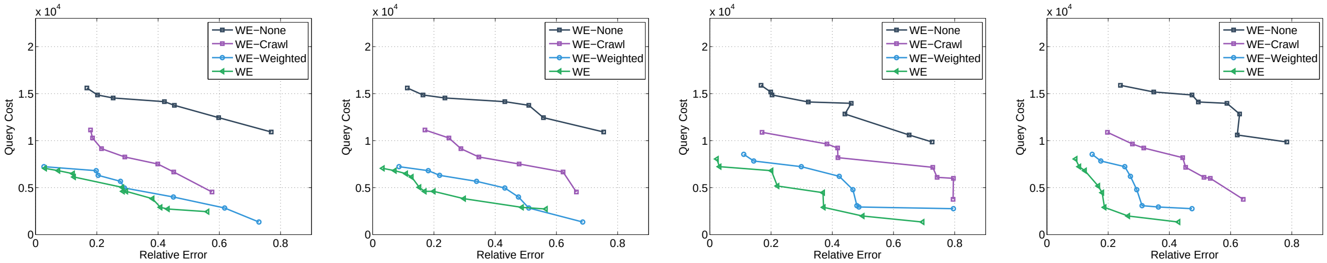
8. RELATED WORK

Random Walks: As discussed in Section 2, random walk is an MCMC based sampling method extensively studied in statistics (e.g, [10]). Besides the traditional random walk designs described



(a) Twitter: Average In-Degree (b) Twitter: Average Out-Degree (c) Twitter: Average Local Clustering Coefficient (d) Twitter: Average Local Clustering Coefficient

Figure 7: Relative Error of the Average Estimations vs Query Cost in Twitter (from SNAP repository).



(a) Average Degree (SRW) (b) Average Self-description Length (SRW) (c) Average Degree (MHRW) (d) Average Self-description Length (MHRW)

Figure 8: Improvement Trend: Relative Error of the Average Estimations vs Query Cost in Google Plus.

Distance Measure	Dist(Theoretical, SRW)	Dist(Theoretical, WE)
ℓ_∞	0.0081	0.00549
K-L Divergence	0.47529	0.01834

Table 1: Distance between Theoretical Sampling Distribution and that of SRW/WE

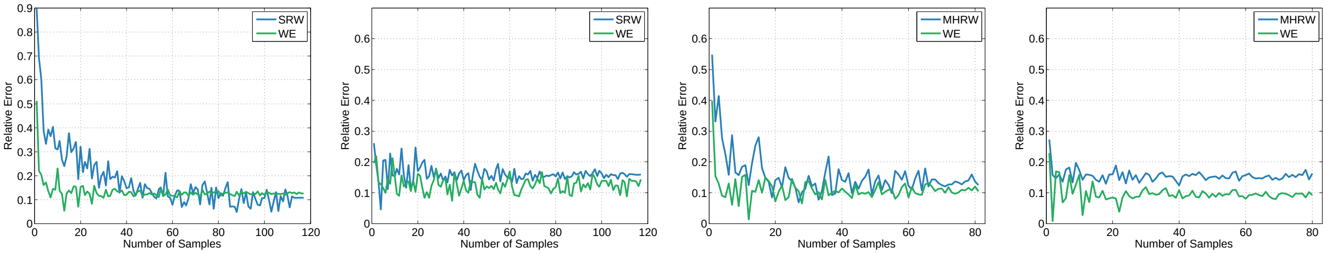
in Section 2, two key related concepts used in this paper are *burn-in period*, which captures the number of steps a random walk takes before converging to its stationary distribution [22]; and convergence monitors, heuristic techniques for measuring on-the-fly how long the burn-in period should be (i.e., determining when a random walk should be stopped and a sample taken). Examples here include Geweke, Raftery and Lewis, Gelman and Rubin convergence monitors (see [7] for a comprehensive review).

Random Walks on Social Networks: There have been extensive studies (e.g., [1, 15, 18]) on the sampling of online social networks which feature graph browsing interfaces [29] that enforce the aforementioned local-neighborhood-only access limitation. [18] introduces a taxonomy of sampling techniques - specifically, node sampling, edge sampling and subgraph sampling. For the problem studied in this paper - i.e., sampling nodes from online social networks - the usage of multiple parallel random walks is studied in [2], while several studies (e.g., [18]) demonstrates the superiority of random walk techniques such as Simple Random Walk (SRW), Metropolis-Hastings Random Walk (MHRW) over baseline solutions such as Breadth First Search (BFS) and Depth First Search (DFS). An interesting issue studied in the literature is the comparison between SRW and MHRW over real-world social networks - the finding in [11] is that MHRW is less efficient than SRW because MHRW mixes more slowly. While our technique discussed in the paper is transparent to the input random walk, a similar comparison result can be observed from our experimental results as well.

Improving the Efficiency of Random Walks: Most related to this paper are the previous studies on improving the efficiency of random walks over online social networks. To this end, [13] combines random jump and MHRW to efficiently retrieve uniform random node samples from an online social networks. Nonetheless, in order to enable random jumps, this study assumes access to an *ID generator* which can sample a node uniformly at random with a high hit rate - an assumption that is not satisfied by many online social networks and not assumed in this paper. Another study [26] considers frontier sampling which converts input samples with uniform distribution to output samples with arbitrary target distribution. Our study in the paper is transparent to this work - as we address the problem of generating sample nodes rather than assuming access to samples with pre-determined distributions. Also related to efficiency enhancements are [17] which introduces a non-backtracking random walk that converges faster with less asymptotic variance than SRW and [30] which modifies the topology of the underlying graph on-the-fly in order to get a faster random walk on the modified graph. A key difference between WALK-ESTIMATE and all these existing studies is that while all existing techniques still wait for convergence to the target distribution, we do not wait for convergence, but rather proactively estimate the sampling distribution and then use rejection sampling to achieve the target distribution.

9. FINAL REMARKS

In this paper, we developed WALK-ESTIMATE, a general purpose technique for faster sampling of nodes over an online social network with any target (sampling) distribution. Our key idea is to conduct a random walk for a small number of steps, and follow it with a proactive estimation of the sampling distribution of the node encountered before applying acceptance-rejection sampling to achieve the target distribution. Specifically, we presented two main components of WALK-ESTIMATE, WALK which determines the number of steps to walk, and ESTIMATE which enables an unbiased estimation of the sampling distribution. Theoret-



(a) Average Degree (SRW)

(b) Average Self-description Length (SRW)

(c) Average Degree (MHRW)

(d) Average Self-description Length (MHRW)

Figure 9: Relative Error of the Average Estimations vs Number of Samples in Google Plus.

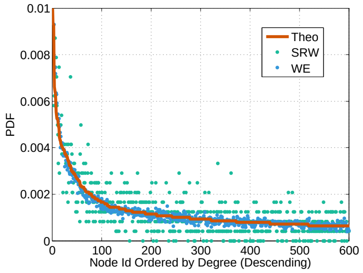


Figure 10: Comparing Sampling Distribution for Scale-Free Graph

ical analysis and extensive experimental evaluations over synthetic graphs and real-world online social networks demonstrated the superiority of our technique over the existing random walks.

10. REFERENCES

[1] E. M. Airoldi. Sampling algorithms for pure network topologies. *SIGKDD Explorations*, 7:13–22, 2005.

[2] N. Alon, C. Avin, M. Koucky, G. Kozma, Z. Lotker, and M. R. Tuttle. Many random walks are faster than one. In *SPAA*, 2008.

[3] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. Four degrees of separation. In *Proc. of the 3rd Annual ACM Web Science Conf.*, pages 33–42. ACM, 2012.

[4] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[5] B. Bollobás and O. Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2004.

[6] R. Cohen and S. Havlin. Scale-Free Networks Are Ultrasmall. *Phys. Rev. Lett.*, 90, 2003.

[7] M. K. Cowles and B. P. Carlin. Markov chain monte carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91:883–904, 1996.

[8] A. Dasgupta, G. Das, and H. Mannila. A random walk approach to sampling hidden databases. In *SIGMOD*, 2007.

[9] C. J. Geyer. Practical markov chain monte carlo. *Statistical Science*, 1992.

[10] W. R. Gilks. *Markov Chain Monte Carlo In Practice*. Chapman and Hall/CRC, 1999.

[11] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *INFOCOM*, 2010.

[12] M. Jerrum and A. Sinclair. Conductance and the rapid mixing property for markov chains: The approximation of permanent resolved. In *STOC*, 1988.

[13] L. Jin, Y. Chen, P. Hui, C. Ding, T. Wang, A. V. Vasilakos, B. Deng, and X. Li. Albatross sampling: robust and effective hybrid vertex sampling for social graphs. In *MobiArch*, 2011.

[14] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. of the Thirty-second Annual ACM Symposium on Theory of Computing*. ACM, 2000.

[15] M. Kurant, M. Gjoka, C. T. Butts, and A. Markopoulou. Walking on a graph with a magnifying glass: stratified sampling via weighted random walks. In *SIGMETRICS*, 2011.

[16] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proc. of the 19th int. conf. on World wide web*, pages 591–600. ACM, 2010.

[17] C.-H. Lee, X. Xu, and D. Y. Eun. Beyond random walk and metropolis-hastings samplers: Why you should not backtrack for unbiased graph sampling. SIGMETRICS, 2012.

[18] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *SIGKDD*, 2006.

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

[20] D. Liben-nowell and E. D. Demaine. An algorithmic approach to social networks. Technical report, PhD thesis at MIT References 118 Science and Artificial Intelligence Laboratory, 2005.

[21] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2(1):1–46, 1993.

[22] S. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, 2nd edition, 2009.

[23] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *SIGCOMM*, 2007.

[24] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *SIGCOMM*, 2010.

[25] A. Nazi, Z. Zhou, S. Thirumuruganathan, N. Zhang, and G. Das. Walk, not wait: Faster sampling over online social networks. *CoRR*, abs/1410.7833, 2014.

[26] B. Ribeiro and D. Towsley. Estimating and sampling graphs with multidimensional random walks. In *SIGCOMM*, 2010.

[27] B. F. Ribeiro, P. Wang, F. Murai, and D. Towsley. Sampling directed graphs with random walks. In *INFOCOM'12*, pages 1692–1700, 2012.

[28] E. Shyu. Diameter bounds and eigenvalues. Technical report, MIT, 2013.

[29] N. Zhang and G. Das. Exploration of deep web repositories. In *Proc. of the VLDB Endowment (VLDB), Tutorial*, 2011.

[30] Z. Zhou, N. Zhang, Z. Gong, and G. Das. Faster random walks by rewiring online social networks on-the-fly. ICDE, 2013.