On Sampling Nodes in a Network

Flavio Chierichetti[*]
Sapienza University
Rome, Italy
flavio@di.uniroma1.it

Anirban Dasgupta[†]
IIT
Gandhinagar, India
anirbandg@iitgn.ac.in

Ravi Kumar
Google
Mountain View, CA
ravi.k53@gmail.com

Silvio Lattanzi
Google
New York, NY
silviolat@gmail.com

Tamás Sarlós
Google
Mountain View, CA
stamas@gmail.com

ABSTRACT

Random walk is an important tool in many graph mining applications including estimating graph parameters, sampling portions of the graph, and extracting dense communities. In this paper we consider the problem of sampling nodes from a large graph according to a prescribed distribution by using random walk as the basic primitive. Our goal is to obtain algorithms that make a small number of queries to the graph but output a node that is sampled according to the prescribed distribution. Focusing on the uniform distribution case, we study the query complexity of three algorithms and show a near-tight bound expressed in terms of the parameters of the graph such as average degree and the mixing time. Both theoretically and empirically, we show that some algorithms are preferable in practice than the others. We also extend our study to the problem of sampling nodes according to some polynomial function of their degrees; this has implications for designing efficient algorithms for applications such as triangle counting.

Keywords. random walks, stationary distribution, mixing time, uniform sampling, Metropolis–Hastings.

1. INTRODUCTION

Sampling large networks is a fundamental data mining problem. When the network is huge and it is costly or infeasible to process the network in its entirety, sampling is often the most realistic option to infer properties of the network and to obtain estimates about basic quantities. Sampling has been successfully used for many network-related measurements, ranging from estimating the size of the network, the number of edges and the average degree, or even higher-order properties such as the clustering coefficient, the number of triangles and small subgraphs, and more. Sampling in general, and network sampling in particular, has been studied extensively in both theoretical and applied research communities, and there is a rich body of literature on these topics.

By the very nature of the sampling process, the obtained estimate is approximate and can sometimes be incorrect. A principled way to sample is thus critical in order to ensure that the obtained estimate has any statistical quality guarantee. Theoretical results on sampling especially emphasize this aspect: they focus on the number of samples sufficient to estimate a quantity to within an additive or multiplicative error and with high probability.

Sampling in networks. One of the basic primitives for sampling in networks is the problem of generating an independent and uniformly distributed network node. This primitive is very powerful since estimating many key network properties can be expressed in terms of a statistical estimator based on a uniform node sample. For example, the fraction of left-leaning members in an online social network can be estimated by querying a uniform sample of members (i.e., nodes) about their political alignments. Likewise, the size of the network can be estimated by generating several independent uniform nodes and counting the collisions among them. Efficient and non-trivial estimators for several other properties such as the average degree and clustering coefficient can also be expressed as statistical estimators that are based on uniform samples.

An operational issue in sampling that is unique to networks is the question of how to access nodes and edges. The following oracle model is widely used: given any node, in a unit time, one can obtain all the neighbors of the node. This access model is realistic given the way large networks are typically accessed in practice. For instance, by crawling a web page, one can access all the web pages linked to this web page; similarly, by crawling a social profile, one can access all the (publicly-visible) friends of a user. With these in mind, an algorithm to generate a uniform node from a network must have the following desirable properties: (i) it has to be resource-efficient, i.e., it has to query as few nodes of the network as possible, (ii) it has to be provably good, i.e., the approximation guarantees must be quantifiable and tunable, (iii) it must be simple to implement using the oracle query model and efficient to run.

Existing work for uniform node generation are based on a random walk in the network: invoke the access oracle to get the neighborhood of the current node and pick a random neighbor to continue the exploration for a while and output a node as the sample. Evaluation of these algorithms fall into one of two categories: (i) propose natural heuristics for the problem and demonstrate empirically that produce good quality samples or (ii) use some method for sampling but focus on the eventual quantity that is estimated using the samples instead of on the quality of the samples themselves. To the best of our knowledge, the query complexity of uniform node generation in a network has not been formally addressed.

[*] Supported in part by a Google Focused Research Award, by the ERC Starting Grant DMAP 680153, and by the SIR Grant RBSI14Q743.

[†] Supported in part by a Google Faculty Research Award.

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872427.2883045.

Our contributions. In this paper, we study the uniform node generation problem. In particular, we ask: is it possible to relate the query complexity of generating a uniform node in a network to the structural properties of the underlying network? Intuitively, we expect the problem to be easier on well-connected networks than on poorly connected networks. Our work proceeds to formalize this intuition and quantify the resources required for uniform generation. However, this is non-obvious since specific measures of connectivity such as the mixing time could be a weak bound on the number of queries, and unlike mixing time, the query complexity of a walk does not immediately relate to any spectral properties.

For the problem of generating a uniform sample in a network, we study two simple and efficient algorithms, namely, the well-known rejection sampling and an algorithm based on the maximum degree of the network. We show that the query complexity of both these algorithms is the product of the average degree of the network and the mixing time of the naive random walk. This characterization is desirable because both the average degree and the mixing time of the naive random walk are known to be small for real networks. To prove these bounds we use a variational characterization of the mixing time; this technique leads us to tighter bounds in comparison with the ones obtainable using conductance-based methods. We also show that these bounds are asymptotically tight for the algorithms. While the theoretical upper bounds between the two algorithms is near-indistinguishable, our empirical analysis on several real-world network shows that the maximum degree algorithm is at least as good as, and is often superior to rejection sampling; based on this, we advocate the former.

We then study the complexity of an algorithm based on the classical Metropolis–Hastings method. This is a popular practical algorithm to generate a uniform node in a network. However, we show that this method is inferior to the other two algorithms, both in theory and in practice—instead of the average degree, the query complexity depends on the maximum degree of the network. We also show that this dependence is inevitable and the only way to salvage is for the oracle to provide more information about the neighborhood, specifically, the degrees of each of the neighbors.

We also show that the sum of the average degree and the mixing time is a lower bound on any sampling algorithm for the uniform generation problem. We believe that this lower bound is loose and conjecture that the product dependence between these two quantities is the asymptotically optimal bound for the problem.

Finally we study a particular version of the non-uniform sampling question: given a parameter α, what is the complexity of generating a node in the network with probability proportional to the degree of the node raised to the power of α? For $\alpha = 2$, this question has implications for more efficient algorithms (than ones using uniform samples) for estimating higher-order properties such as the clustering coefficient.

2. PRELIMINARIES

We consider undirected connected graphs in this paper. Let $G = (V, E)$ be an undirected graph where V is the set of *nodes* and $E \subseteq \binom{V}{2}$ is the set of *edges*. The *neighborhood* $\Gamma(u)$ of a node u is the set of nodes v such that $\{u, v\} \in E$; the *degree* of a node $u \in V$ is equal to

$$d(u) := |\Gamma(u)| = |\{v \mid \{u, v\} \in E\}|.$$

Let $n = |V|$ denote the number of nodes and let $m = |E|$ denote the number of edges in the graph G. Let

$$d_{\text{avg}} := \frac{1}{n} \sum_u d(u) = \frac{2m}{n},$$

denote the *average* degree, and let

$$d_{\min} := \min_u \{d(u)\}, \quad d_{\max} := \max_u \{d(u)\},$$

denote the *minimum* and the *maximum* degrees in G. For simplicity, we use E also as an adjacency matrix: $E(u, v) = 1$ if $(u, v) \in E$ and is 0 otherwise.

For a function $f : V \to \mathbb{R}$, let $|f| := \sum_u |f(u)|$ denote its one-norm and let $\|f\|_2 := \sqrt{\sum_u f(u)^2}$ denote its two-norm.

Random walk basics. Let $P = \{P(u, v)\}$ be an $n \times n$ transition matrix of a Markov chain on state space V, where the entry $P(u, v)$ gives the probability of transitioning from node u to node v. Let π be the *stationary distribution* of P, i.e.,

$$\sum_u P(u, v)\pi(u) = \pi(v), \text{ for all } v \in V.$$

If P is ergodic (i.e., irreducible and aperiodic, see, for e.g., [16] for formal definitions), then the stationary distribution π is well-defined and unique.

The notion of mixing time quantifies how fast a random walk approaches its limiting distribution. Informally, it is the number of steps needed before the random walk converges to its stationary distribution, regardless of the node from which the walk originated. Formally, let e_u denote the unit vector with 1 in the uth position, let P^t denote the transition matrix raised to the power t, and let

$$\Delta(t, u) := |\pi - e_u P^t|/2,$$

denote the *total variation distance* between the stationary distribution and the distribution after t steps when the walk starts at u. The *mixing time*, with respect to a parameter , is then defined as:

$$\begin{aligned} t_{\text{mix}}(\ , u) &:= \min_t \{\Delta(t', u) \leq \ , \text{ for all } t' \geq t\}, \\ t_{\text{mix}}(\ ) &:= \max_u \{t_{\text{mix}}(\ , u)\}, \\ t_{\text{mix}} &:= t_{\text{mix}}(1/n^2). \end{aligned}$$

For a transition matrix P, let $\{\lambda_i(P)\}$ be the set of eigenvalues such that $|\lambda_1(P)| \geq |\lambda_2(P)| \geq \cdots$. It is easy to see that $\lambda_1(P) = 1$ and since P is aperiodic, $\lambda_2(P)$ is real and non-negative. The second eigenvalue $\lambda_2 = \lambda_2(P)$, in particular, its gap to the first (i.e., $1 - \lambda_2$) is intimately related to the mixing time of G [22]:

$$\frac{\lambda_2}{2(1 - \lambda_2)} \log \frac{1}{2} \leq t_{\text{mix}}(\ ) \leq \frac{1}{1 - \lambda_2} \left(\log \frac{1}{\ \cdot \pi_{\min}} \right), \quad (1)$$

where $\pi_{\min} = \min_u \pi(u)$.

The second eigenvalue also satisfies the following *variational characterization*, which will be useful in our analysis:

$$1 - \lambda_2 = \inf_f \frac{\sum_{u,v} (f(u) - f(v))^2 \pi(u) P(u, v)}{\sum_{u,v} (f(u) - f(v))^2 \pi(u)\pi(u)}, \quad (2)$$

over all non-constant f.

Notation. For the rest of the paper, we use different superscripts to denote different types of random walks considered in this work for quantities such as the transition matrix, the stationary distribution, and the mixing time associated with the walk. For $\alpha \geq 0$, let Π^α denote the distribution in which the degrees are raised to the power of α, i.e., $\Pi^\alpha(u) \propto d(u)^\alpha$.

Uniform random walk. Recall the *uniform random walk* (urw) on G: at each step of the walk, if the current node is u, pick a node v uniformly at random from $\Gamma(u)$ and move to v. The transition matrix P^{urw} associated with this random walk on G is given by

$$P^{\text{urw}}(u, v) = \begin{cases} \frac{1}{d(u)} & \text{if } \{u, v\} \in E, \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

Since G is undirected and connected, it follows that the Markov chain defined by P^{urw} is recurrent; for simplicity, we further assume that it is also aperiodic and hence ergodic. It follows that

$$\pi^{\mathsf{urw}}(u) = \frac{d(u)}{2m}, \tag{4}$$

for $u \in V$. In our notation, this distribution on V is denoted Π^1. The mixing time $t^{\mathsf{urw}}_{\mathrm{mix}}(\ )$ will be an important factor in all our algorithms; for simplicity, we abbreviate it as $t_{\mathrm{mix}}(\ ) = t^{\mathsf{urw}}_{\mathrm{mix}}(\ )$. Since $\pi^{\mathsf{urw}}_{\min} = \Omega(1/\mathrm{poly}(n))$, (1) becomes

$$\frac{c\lambda_2}{(1-\lambda_2)} \log\frac{1}{\ } \le t_{\mathrm{mix}}(\ ) \le \frac{1}{1-\lambda_2}\left(\log n + \log\frac{1}{\ }\right), \tag{5}$$

for some constant c.

Access model and parameters. We assume the following *neighborhood oracle* model for accessing the graph: given any node $u \in V$, in unit time, one can obtain the neighborhood $\Gamma(u)$ of u. Our goal is to design algorithms to produce a sample node according to a specified distribution on V. Quantitatively, however, we will settle for an approximation: an algorithm is said to generate an *-approximate sample* from a distribution if the sample produced by the algorithm is -close in total variation distance to the desired distribution. Note that the neighborhood oracle can be used to easily implement the uniform random walk (urw). This random walk realizes the distribution Π^1 on V given by (4), i.e., a node is output with probability proportional to its degree. However, the main focus in our work is on the uniform distribution Π^0 on V, i.e., each node u should be generated with probability (close to) to $1/|V|$. Later we consider more general distributions Π^α, $\alpha \neq 0, 1$.

There are two parameters of interest for an algorithm $\mathcal{A}$ operating in this model, namely, the number of random walks steps, denoted $\mathtt{steps}(\mathcal{A})$, and the number of *distinct* nodes in the graph accessed, denoted $\mathtt{queries}(\mathcal{A})$, to generate one sample from the desired distribution. Clearly, $\mathtt{queries}(\mathcal{A}) \le \mathtt{steps}(\mathcal{A})$. From the definition of mixing time, $t_{\mathrm{mix}}(\ )$ steps (and queries) are needed to generate a node according to π^{urw} by using the uniform random walk urw. Hence, informally, to generate a node with probability proportional to its degree, the algorithm urw doing the uniform random walk (for a worst case graph) incurs $\mathtt{queries}(\mathsf{urw}) = \Theta(\mathtt{steps}(\mathsf{urw})) = \Theta(\ t_{\mathrm{mix}}(\ ))$.

We assume that the algorithm knows the approximate values of n, $d_{\max}$, d_{avg}, $d_{\min}$, and t_{mix} of the graph. With this assumption, it does not have to explicitly identify a stopping condition, which by itself is non-trivial [18]. We also assume that the algorithm has access to some arbitrary node in the graph to begin the walk.

3. UNIFORM SAMPLING

In this section we discuss three algorithms for sampling a node from the graph uniformly at random, i.e., for generating samples according to Π^0. The first algorithm will serve as the baseline: it is rejection sampling on top of a uniform random walk. The second algorithm is based on a modification of the uniform random walk, taking the maximum degree into account. The third algorithm is the Metropolis–Hastings random walk for generating a sample, using the uniform random walk as the proposal distribution. For each of these algorithms, we show bounds on $\mathtt{steps}(\cdot)$ and $\mathtt{queries}(\cdot)$.

3.1 Rejection sampling

The idea behind rejection sampling (rej) is to do uniform random walk until it mixes, sample a node, and accept it with probability inversely proportional to its degree, and continue. Algorithm 1 contains a formal description.

Algorithm 1 Rejection sampling (rej).

```
Input: Graph $G$, a starting node $u$
  while no node is output do
    for $t_{\mathrm{mix}}(\ )$ steps do
      do a step of the uniform random walk ($P^{\mathsf{urw}}$)
    With probability $d_{\min}/d(u)$, output $u$ and stop
```

Note that after $t_{\mathrm{mix}}(\ )$ steps the walk is at node u with probability $\frac{d(u)}{2m} \pm\ $, furthermore u is returned with probability $d_{\min}/d(u)$. Hence the probability that u is selected is $\frac{d_{\min}}{2m} \pm\ '$, with $'<\ $. Thus the algorithm outputs an almost uniform sample.

THEOREM 1. *To generate an -approximate sample from Π^0, with $\le \frac{d_{\min}}{4m}$, the expected number of steps and queries are*

$$E[\mathtt{queries}(\mathsf{rej})] \le E[\mathtt{steps}(\mathsf{rej})] \in O\left(t_{\mathrm{mix}}(\ ) \cdot \frac{d_{\mathrm{avg}}}{d_{\min}}\right).$$

PROOF. Since rej performs a uniform random walk for $t_{\mathrm{mix}}(\ )$ steps, an equivalent view is that it generates an -approximate sample u according to Π^1, which is then accepted with probability $d_{\min}/d(u)$. Let U be the random node at the end of $t_{\mathrm{mix}}(\ )$ steps. The combined probability of successfully obtaining an -approximate sample according to Π^0 can be calculated as

$$\begin{aligned}\sum_{u\in V}\Pr[U=u]\cdot\frac{d_{\min}}{d(u)} &= \sum_{u\in V}\left(\frac{d(u)}{2m} \pm\ \right)\cdot\frac{d_{\min}}{d(u)} \\ \ge\ d_{\min}\sum_{u\in V}\frac{1}{2m} -\ n &= \frac{d_{\min}}{d_{\mathrm{avg}}} -\ n.\end{aligned}$$

Thus, by repeating this process at most $\frac{d_{\mathrm{avg}}}{d_{\min}-2\epsilon m}$ times, in expectation, we can obtain an -approximate sample from Π^0. □

Even though the sampling algorithm is simple and efficient, it is not efficient in the number of queries, since a node cannot be rejected until it is queried and until a node is output as a sample, all the nodes examined en route are discarded. This seems wasteful. The goal of the the next two algorithms is to address this issue, where they try to distinguish between $\mathtt{steps}(\cdot)$ and $\mathtt{queries}(\cdot)$, with the goal of reducing the latter, regardless of the former.

3.2 Maximum-degree sampling

In maximum-degree sampling (md), the main idea is to use the skeleton of the graph and adjust the transition probabilities by using the maximum degree [1, 2], in order to make the random walk spend more time at low degree nodes (i.e., have more mass on the self-loop edges). The intuition being that, since a uniform random walk on a regular graph converges to the uniform distribution, by modifying the random walk this way we make the graph regular and hence a uniform random walk on this modified graph would naturally converge to the uniform distribution. A benefit of this modification is that while the mixing time of the new random walk might be much more than the original walk, since the walk is now forced to spend more time at a node before transitioning out, the number of distinct nodes queried can be much lower. In other words, $\mathtt{queries}(\mathsf{md})$ can be much lower than $\mathtt{steps}(\mathsf{md})$.

A formal description of md is given by a uniform random walk on the Markov chain given by the following transition matrix P^{md}.

$$P^{\mathsf{md}}(u,v) = \begin{cases} \frac{1}{d_{\max}} & \{u,v\} \in E, \\ 1-\frac{d(u)}{d_{\max}} & u = v, \\ 0 & \text{otherwise.}\end{cases} \tag{6}$$

LEMMA 2. *The random walk* md *satisfies the following properties: (i)* $\pi^{\mathsf{md}} = \Pi^0$*, the uniform distribution; (ii) the expected number of steps spent in self-loop at a node* u *is* $d_{\max}/d(u)$*; and (iii) the distinct nodes queried in a walk according to* md *corresponds to a valid walk according to* urw.

PROOF. The first two properties follow from the definition of P^{md}. The last property follows from the observation that the repeated nodes in a walk corresponding to md arise from the self-loops and by eliminating the duplicates, a node v following u in a walk would have been chosen uniformly among $\Gamma(u)$, which would be the behavior of urw. □

Now, we proceed to analyzing the mixing time and the number of queries for this sampling algorithm. Our goal is to express these in terms of the parameters of G. This turns out to be a delicate task since the graph itself has been modified.

We first show a lower bound on a useful quantity in terms of the minimum and average degrees of the graph.

LEMMA 3. *Let* $\mathcal{F}$ *denote the set of all real-valued functions on the nodes* V *of* G*. Then,*

$$\inf_{f\in\mathcal{F}} \frac{\sum_{u,v\in V}(f(u)-f(v))^2 d(u)d(v)}{\sum_{u,v\in V}(f(u)-f(v))^2} \geq \frac{d_{\min}d_{\text{avg}}}{2}.$$

PROOF. For $f\in\mathcal{F}$, $f : V\to\mathbb{R}$, let

$$Q(f) = \frac{\sum_{u,v}(f(u)-f(v))^2 d(u)d(v)}{\sum_{u,v}(f(u)-f(v))^2}.$$

We will show that for any $f\in\mathcal{F}$, $Q(f)\geq d_{\min}(d_{\min}+d_{\text{avg}})/2$. We first note that without loss of generality, $\sum_u f(u) = 0$. Indeed, we can define the function $f'\in\mathcal{F}$ to be $f'(u) = f(u) - \sum_u f(u)/n$ and it follows from definition that $Q(f') = Q(f)$. Furthermore, note that for any non-zero $f\in\mathcal{F}$, it holds that $Q(f) = Q(f/\|f\|_2)$ and hence we can also assume that f has unit norm, i.e., $\|f\|_2^2 = \sum_u f(u)^2 = 1$.

Now, with these assumptions on f,

$$\begin{aligned} Q(f) &= \frac{\sum_{u,v}(f(u)-f(v))^2 d(u)d(v)}{\sum_{u,v}(f(u)-f(v))^2} \\ &\geq d_{\min}\cdot\frac{\sum_{u,v}(f(v)-f(v))^2 d(u)}{\sum_{u,v}(f(u)-f(v))^2}. \qquad (7)\end{aligned}$$

The denominator in (7) evaluates to

$$\begin{aligned}\sum_{u,v}(f(u)-f(v))^2 &= 2n\sum_u f(u)^2 - 2\sum_u f(u)\sum_v f(v) \\ &= 2n\sum_u f(u)^2 \quad = \quad 2n,\end{aligned}$$

using the assumptions $\sum_u f(u) = 0$ and $\sum_u f(u)^2 = 1$. The numerator in (7) can similarly be simplified as

$$\begin{aligned}&\sum_{u,v}(f(u)-f(v))^2 d(u) \\ &= n\sum_u f(u)^2 d(u) + \sum_u d(u)\sum_v f(v)^2 \\ &\quad -2\sum_u f(u)d(u)\sum_v f(v) \\ &= n\sum_u f(u)^2 d(u) + \sum_u d(u).\end{aligned}$$

Hence, (7) evaluates to

$$\begin{aligned} Q(f) &\geq d_{\min}\frac{n\sum_u f(u)^2 d(u) + \sum_u d(u)}{2n} \\ &= d_{\min}\frac{(\sum_u f(u)^2 d(u)) + d_{\text{avg}}}{2} \\ &\geq d_{\min}\frac{d_{\min}+d_{\text{avg}}}{2} \\ &\geq \frac{d_{\min}d_{\text{avg}}}{2}.\end{aligned}$$

Since this lower bound holds for any $f\in\mathcal{F}$, the proof follows. □

Notice that the bound in Lemma 3 is near-tight: it is achieved by $f(u) = 1$ for some node u such that $d(u) = d_{\min}$ and $f(v) = o(1)$ for all nodes $v\neq u$. Using Lemma 3, we can obtain an upper bound on the number of queries and the mixing time of md.

THEOREM 4. *To generate an -approximate sample from* Π^0*, the number of steps needed by* md *is given by:*

$$\texttt{steps}(\mathsf{md}) \quad = \quad c_1(n,\ )\cdot\frac{d_{\max}}{d_{\min}}\cdot t_{\text{mix}}(\ )$$

where $c_1(n,\ ) = 2c_0(\log n + \log(1/\ ))$ *for some constant* c_0*. Furthermore, with probability* $1 - 1/K$*, this many steps are taken if* md *is run until it does*

$$\texttt{queries}(\mathsf{md}) \quad = \quad K(c_1(n,\ )+1)\cdot\frac{d_{\text{avg}}}{d_{\min}}\cdot t_{\text{mix}}(\ ),$$

many (distinct) queries, for any constant $K\geq 1$.

PROOF. Applying the variational characterization (2) to P^{md}, using the definition in (6) and the fact that $\pi^{\mathsf{md}}(u) = 1/n$ for all u, we obtain

$$\begin{aligned} 1-\lambda_2(P^{\mathsf{md}}) &= \inf_f \frac{\sum_{u,v}(f(u)-f(v))^2\pi^{\mathsf{md}}(u)P^{\mathsf{md}}(u,v)}{\sum_{u,v}(f(u)-f(v))^2\pi^{\mathsf{md}}(u)\pi^{\mathsf{md}}(v)} \\ &= \frac{n}{d_{\max}}\cdot\inf_f\frac{\sum_{u,v}(f(u)-f(v))^2E(u,v)}{\sum_{u,v}(f(u)-f(v))^2}. \qquad (8)\end{aligned}$$

Similarly, for the simple walk P^{urw}, using $\pi^{\mathsf{urw}}(u) = d(u)/(2m)$, we have that

$$1-\lambda_2(P^{\mathsf{urw}}) = 2m\cdot\inf_f\frac{\sum_{u,v}(f(u)-f(v))^2E(u,v)}{\sum_{u,v}(f(u)-f(v))^2 d(u)d(v)}. \qquad (9)$$

Hence, (8) leads to

$$1-\lambda_2(P^{\mathsf{md}}) = \frac{n}{d_{\max}}\cdot\inf_f \sum_{u,v}(f(u)-f(v))^2E(u,v)$$

Now, using (5), it then follows

$$t^{\mathsf{md}}_{\mathrm{mix}}(\ ) \leq \frac{1}{1-\lambda_2(P^{\mathsf{md}})}\left(\log n + \log\frac{1}{\ }\right)$$
$$\leq 2\frac{d_{\max}}{d_{\min}}\cdot\frac{1}{1-\lambda_2(P^{\mathsf{urw}})}\left(\log n + \log\frac{1}{\ }\right)$$

Now note that, when $\lambda_2(P^{\mathsf{urw}}) \leq \frac{1}{2}$, $\frac{1}{1-\lambda_2(P^{\mathsf{urw}})} \leq 2$, and when $\lambda_2(P^{\mathsf{urw}}) > \frac{1}{2}$, $\frac{1}{1-\lambda_2(P^{\mathsf{urw}})} \leq \frac{t_{\mathrm{mix}}(\epsilon)}{c\lambda_2 \log\frac{1}{\epsilon}} \leq \frac{2t_{\mathrm{mix}}(\epsilon)}{c\log\frac{1}{\epsilon}}$. Since $t_{\mathrm{mix}}(\ ) \geq 1$, we bound both the cases using $c_0 t_{\mathrm{mix}}(\ )$ for some appropriate constant $c_0 \geq 2$.

Hence, $t^{\mathsf{md}}_{\mathrm{mix}}(\ ) \leq 2\frac{d_{\max}}{d_{\min}} \cdot t_{\mathrm{mix}}(\ )c_0\left(\log n + \log\frac{1}{\epsilon}\right) = \frac{d_{\max}}{d_{\min}} \cdot t_{\mathrm{mix}}(\ ) \cdot c_1(n,\ )$. Thus, the mixing time of md could be more than that of urw. However, as we mentioned earlier, our focus is on queries(md). To bound this, we use the observations in Lemma 2. If we let md run until it makes $(c_1(n,\ )+1)\cdot d_{\mathrm{avg}}/d_{\min}\cdot t_{\mathrm{mix}}(\ )$ queries, then by Lemma 2(ii), the expected number of steps is at least $c_1(n,\ )\cdot d_{\max}/d_{\min}\cdot t_{\mathrm{mix}}(\ )$. Using Markov inequality, by accruing K times those many queries, with probability at least $1-\frac{1}{K}$ we have taken the necessary number of steps. □

3.3 Metropolis–Hastings sampling

The Metropolis-Hastings algorithm is used to compute samples from any distribution Π starting from an arbitrary transition matrix Q. The only condition needed on Q is symmetry, i.e., $Q(u,v) = Q(v,u)$. The algorithm [7, 17, 19, 23] works in rounds and in each round does the following: if the current state is u, it generates a candidate $v \in V$ with probability proportional to $Q(u,v)$, and accepts and moves to v with probability $\min(1, \Pi(v)/\Pi(u))$. The stationary distribution of this process can be shown to be Π. For more details on the Metropolis–Hastings algorithm, see [10].

In our case, $\Pi = \Pi^0$. Ideally, we would like to define Q using P^{urw}. Unfortunately, this is not easy to do since P^{urw} is not necessarily symmetric (due to normalization by the degree of a node). Therefore, we define the following transition matrix P^{MH} by using the adjacency matrix E of the graph.

$$P^{\mathsf{MH}}(u,v) = \begin{cases} \frac{1}{\max(d(u),d(v))} & \{u,v\} \in E, \\ 1 - \sum\limits_{w\in\Gamma(u)} \frac{1}{\max(d(u),d(w))} & u = v, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

It is easy to see that $\pi^{\mathsf{MH}} = \Pi^0$, the uniform distribution. It also easy to see that the first case of (10) is equivalent to $1/d(u) \cdot \min(1, d(u)/d(v))$, which is the way MH is typically written.

THEOREM 5. *To generate an -approximate sample from Π^0, the number of steps and queries needed are given by*

$$\mathtt{steps}(\mathsf{MH}) \quad = \quad c_3(n,\ )\cdot\frac{d_{\max}}{d_{\min}}\cdot t_{\mathrm{mix}}(\ ),$$

where $c_3(n,\ ) = c_2(\log n + \log(1/\ ))$ for some constant c_2.

PROOF SKETCH. The proof proceeds along similar as that of Theorem 4. We only provide the differences. As before, using the variational characterization in (2), $\pi^{\mathsf{MH}}(u) = 1/n$ for all u, and the definition of P^{MH} in (10), we obtain

$$1-\lambda_2(P^{\mathsf{MH}}) \quad = \quad n\cdot\inf_f \frac{\sum_{u,v}(f(u)-f(v))^2\frac{E(u,v)}{\max(d(u),d(v))}}{\sum_{u,v}(f(u)-f(v))^2}$$
$$\geq \quad \frac{n}{d_{\max}}\inf_f\frac{\sum_{u,v}(f(u)-f(v))^2E(u,v)}{\sum_{u,v}(f(u)-f(v))^2},$$

and the rest of the proof continues as in Theorem 4. □

Note that the number of steps is also an upper bound on the number of queries.

4. LOWER BOUNDS

In this section we will prove an $\Omega(d_{\mathrm{avg}} + t_{\mathrm{mix}})$ query lower bound for algorithms that try to generate a uniform at random node from a graph. Then, we will also show that the bounds we obtain for md algorithm is tight. For the lower bounds, we will assume the neighborhood oracle model for accessing the graph. The general flavor of our lower bound methodology is to create families of (random) graphs where we use combinatorial arguments and simple properties of random walks to establish the desired bounds.

While we are currently unable to show, we conjecture that $\Omega(d_{\mathrm{avg}}\cdot t_{\mathrm{mix}})$ is the lower bound on the number of queries for generating a uniform node in a graph for which only the average degree and the mixing time are known.

4.1 An $\Omega(d_{\mathrm{avg}})$ lower bound

To show an $\Omega(d_{\mathrm{avg}})$ query lower bound, let d be large enough and let $n < e^{o(\sqrt{d})}$; thus, $d = \omega(\log^2 n)$. We generate a random graph G according to the following process. Let $H \sim G\left(n, \frac{d}{n}\right)$ be an Erdős–Rényi random graph and let c be a fair two-sided coin. If $c = \mathtt{head}$, then $G = H$. If $c = \mathtt{tail}$, then we add more nodes and edges to H in the following way to create G: consider each node v in G independently, and with probability $1/d$, add d new nodes and connect all of them to v. Those v's that acquire these new nodes are considered *changed* and the rest, *unchanged*. Let $s \in V$ be a uniform unchanged node.

LEMMA 6. *With high probability, G will have the following properties: (i) $d_{\mathrm{avg}} = (1\pm o(1))\cdot d$; (ii) if $c = \mathtt{head}$, then G has no degree 1 nodes and $|V| = n$; (iii) if $c = \mathtt{tail}$, then the number of degree 1 nodes in G is $(1\pm o(1))\cdot n$, and $|V| = (2\pm o(1))n$; and (iv) $t_{\mathrm{mix}} \leq O\left(\frac{\log n}{\log d}\right)$.*

PROOF. (i) and (iii) follow from a simple application of the Chernoff bound and (ii) follows from the construction. For (iv), the mixing time can be upper bounded by $O\left(\frac{\log n}{\log d}\right)$, using known results on mixing time on simple random graphs (see, for example, [11]) and an application of the Chernoff bound. □

Let $\mathcal{A}$ be an algorithm that aims to return a uniform node from G with $\mathtt{queries}(\mathcal{A}) = o(d)$. We strengthen $\mathcal{A}$ by assuming that it knows the process that generated G (but that does not know G itself) and starts exploring G from s. Our plan would be to see if $\mathcal{A}$ can reasonably guess the coin c with $o(d)$ queries since the uniform distribution on G when $c = \mathtt{head}$ is far away from that when $c = \mathtt{tail}$.

THEOREM 7. *Algorithm $\mathcal{A}$ can distinguish whether the coin $c = \mathtt{head}$ or $c = \mathtt{tail}$ with probability at most $\frac{1}{2}+o(1)$. Thus, the total variation distance between the distribution of the output of $\mathcal{A}$ and Π^0 is at least $\frac{1}{2}-o(1)$, with probability at least $\frac{1}{2}-o(1)$.*

PROOF. By definition, an unchanged node has exactly the same neighborhood regardless of the outcome of c. Since the probability of the process changing a node is at most $1/d$, if $\mathtt{queries}(\mathcal{A}) = o(d)$, then it will query zero changed nodes with probability at least $1-o(1)$. Indeed, since the changes are independent, the probability that any given set $S \subseteq V$ of size $|S| \leq o(d)$ nodes will contain at least one changed node is at most

$$1-\left(1-\frac{1}{d}\right)^{|S|} \leq 1-\left(1-\frac{1}{d}\right)^{o(d)} \leq 1-e^{-o(1)} = o(1).$$

Thus, regardless of the outcome of c, with probability $1 - o(1)$, none of the nodes queried by $\mathcal{A}$ is changed; in other words, the view of the algorithm is oblivious to the outcome of c. However, the total variation distance between Π^0 conditioned on $c = \texttt{head}$ and Π^0 conditioned on $c = \texttt{tail}$ is at least $\frac{1}{2} - o(1)$, since the fraction of changed nodes is at least $\frac{1}{2} - o(1)$ if $c = \texttt{tail}$. The proof is complete. □

4.2 An $\Omega(t_{\mathrm{mix}})$ lower bound

Let k be large enough and let $t \le k^{\Theta(1)}$. Let $H \sim G\left(k, \frac{\log^2 k}{k}\right)$, be an Erdős–Rényi random graph. Replace each edge $e = \{u, v\}$ of H by two new edges $\{u, w_e\}$ and $\{v, w_e\}$, and attach t new degree-one nodes to w_e. Let G be the resulting graph. Note that although the graph is bipartite, we can add self-loops with probability $1/2$ to make the walk aperiodic. The following are easy properties of G (we omit the proof in this version).

LEMMA 8. *With high probability,* $|V| = tk\log^2 k$ *and* $t_{\mathrm{mix}} = \Theta\left(t \cdot \frac{\log k}{\log\log k}\right)$.

Let s be a uniformly chosen node of H. Let $\mathcal{A}$ be an algorithm that aims to return a uniform node from G with $\texttt{queries}(\mathcal{A}) = o(t_{\mathrm{mix}})$. We strengthen $\mathcal{A}$ by assuming that it knows the process that generated G (but that does not know G itself) and starts exploring G from s. The main idea is to show that since most nodes of G are far from any starting point (in H), the algorithm $\mathcal{A}$ will be unable to obtain a uniform node of G.

THEOREM 9. *The total variation distance between the distribution output by* $\mathcal{A}$ *and* Π^0 *is at least* $1 - o(1)$.

PROOF. Observe that if edge $e = \{u, v\}$ is incident on node v in H, and if algorithm $\mathcal{A}$ knows v and w_e but not u (i.e., the other end point of e in H), then it will require $\Omega(t)$ queries in expectation to know which of the neighbors of w_e in G is u. Since $\texttt{queries}(\mathcal{A}) = o\left(t \cdot \frac{\log k}{\log\log k}\right)$, by the Chernoff bound, $\mathcal{A}$ will query at most $o\left(\frac{\log k}{\log\log k}\right)$ nodes of H with high probability.

Hence, with probability $1 - o(1)$, none of the nodes that are at distance more than $\Omega\left(\frac{\log k}{\log\log k}\right)$ from s will be reached. Since the number of these nodes is a $1 - o(1)$ fraction of the total number of nodes of H, the proof follows. □

4.3 A tight example for rej

Let $D = D(n)$ be such that $\omega(\log n) < D(n) < o\left(2^{\log^{1-\epsilon} n}\right)$. Sample $H \sim G(n, D/n)$, then add to it a new node v, and connect it to a single node of H; let G be the resulting graph. Observe that each node of H will have degree $\Theta(D)$ with high probability; hence $d_{\mathrm{max}}, d_{\mathrm{avg}} = \Theta(D)$, $d_{\mathrm{min}} = 1$ and $t_{\mathrm{mix}} = \Theta\left(\frac{\log n}{\log D}\right)$.

The following lemma prove that our analysis of rej (with $\ = o(n^{-2})$) is tight.

LEMMA 10. *On* G, $\texttt{queries}(\mathsf{rej}) \ge \Omega(d_{\mathrm{avg}} \cdot t_{\mathrm{mix}})$ *with high probability.*

PROOF SKETCH. If rej rejects $r < O(d_{\mathrm{avg}})$ nodes in total, since $d_{\mathrm{avg}} \cdot t_{\mathrm{mix}} = o(n)$, we will have that a constant fraction of the nodes visited by rej will result in unique queries. Hence, rej will query at least $\Omega(r \cdot t_{\mathrm{mix}})$ nodes. Now, observe that the probability that $r < o(d_{\mathrm{avg}}) = o(D)$ is at most

$$O\left(\frac{r}{Dn}\right) + \left(1 - \left(1 - \frac{1}{\Theta(D)}\right)^r\right) = O\left(\frac{r}{Dn}\right) + O\left(\frac{r}{D}\right) = o(1).$$

The proof follows. □

4.4 A tight example for md

We now give an example showing the tightness of our analysis of the md sampling of Section 3.2. Let k be a parameter and let $D > k$ be large enough. Consider the following graph $G = G_{D,k}$: a clique of D nodes, and each of the clique's nodes attached to a distinct path of k nodes. Let $\ > 0$ be a fixed constant.

LEMMA 11. G *has the following properties: (i)* $|V| = D \cdot (k+1)$*; (ii)* $d_{\mathrm{avg}} = \Theta(D/k)$, $d_{\mathrm{max}} = D$, *and* $d_{\mathrm{min}} = 1$*; and* $t_{\mathrm{mix}}(\ ) = \Theta(k^2)$.

PROOF SKETCH. The first two properties are easy to see. For the mixing time, we first start with the lower bound. If we begin a walk from one of the nodes of degree 1, i.e., one of the furthest nodes from the clique, in $o(k^2)$ steps, the probability that we will reach the clique is $o(1)$. This follows from mixing time bounds on the a uniform random walk on the line. Thus, $t_{\mathrm{mix}}(1/3) \ge \Omega(k^2)$. For the upper bound, we can show that $1 - \lambda_2 \ge \Omega(1/k^2)$ (this follows from Cheeger's inequality [4] and the fact that the conductance of G is at least $\Omega(1/k)$; we omit the details in this version). Thus, using (1), we have $t_{\mathrm{mix}}(\ ) \le O(k^2)$. □

We now show that one cannot decrease the bound on $\texttt{steps}(\mathsf{md})$ given by Theorem 4 below $\Omega(t_{\mathrm{mix}}(\ ) \cdot d_{\mathrm{max}}/d_{\mathrm{min}})$.

LEMMA 12. *On* G*, to get to within a total variation distance* $o(1)$ *from uniformity, one needs to have* $\texttt{steps}(\mathsf{md}) \ge \Omega(t_{\mathrm{mix}}(1/3) \cdot d_{\mathrm{max}})$.

PROOF SKETCH. Observe that if we do only $o(k^2)$ steps of the P^{md} walk towards nodes of degree 2 in $G_{D,k}$ (i.e., towards the nodes in the paths), the probability that we will ever reach a node that is at distance $\ge k/2$ from the clique in $G_{D,k}$ is $o(1)$. This follows from the standard analysis of the uniform random walk on a line. Furthermore, observe that the number of nodes that are at distance $\ge k/2$ from the clique is a constant fraction of $|V|$.

Thus, for the total variation distance of the sample to be $o(1)$ from uniformity, we need to make at least $\Omega(k^2)$ steps towards nodes of degree 2 in G. The expected number of self-loop steps of P^{md} that we do when on a node of degree 2 is $\Omega(D)$ (derived by calculating the expectation of a geometric random variable). Therefore, the number of steps of P^{md} that we need to make to be within total variation distance $o(1)$ from uniformity is at least $\Omega(k^2) \cdot \Omega(D) = \Omega(k^2 D)$. Since $t_{\mathrm{mix}}(\ ) \cdot d_{\mathrm{max}} = \Theta(k^2 D)$, the claim follows. □

We then show that the bound on $\texttt{queries}(\mathsf{md})$ given by Theorem 4 is also near-tight. We use a different family of graphs for showing this lower bound. Note that this bound is purely a function of the definition of P^{md} and does not really use the fact that we are aiming to sample according to Π^0.

LEMMA 13. *Suppose* md *runs for* $\Omega(t_{\mathrm{mix}}(\ )d_{\mathrm{max}})$ *steps. Then, there are graphs for which* $\texttt{queries}(\mathsf{md}) \ge \Omega(t_{\mathrm{mix}}(\ )d_{\mathrm{avg}})$.

PROOF. Pick a large enough n, pick $\omega(\log n) < T < o(\sqrt{n})$, and $\omega(\log n) < D < o(\sqrt{n})$. Let $H \sim G(n, D/n)$ be an Erdős–Rényi random graph. We choose as G the graph obtained by a copy of H connected via a single edge to a clique of $\Theta(\sqrt{T})$ nodes. The graph G will have $\Theta(n)$ nodes, average and maximum degree $d_{\mathrm{avg}}, d_{\mathrm{max}} = \Theta(D)$, and mixing time $t_{\mathrm{mix}}(\ ) \ge \Omega(T)$ (indeed, if the walk starts in the clique, one will have to wait $\Omega(T)$ steps, in expectation, to leave the clique).

Now, suppose that we run md on G, starting, for simplicity, from a uniform at random node in H. Recall that $TD \le O(t_{\mathrm{mix}} d_{\mathrm{max}})$, and consider the first $\Theta(TD)$ steps of the walk. The probability that the walk will cross the edge connecting H to the clique is $o(1)$. By the Chernoff bound, with high probability, the maximum absolute difference between the degree of a node in H from D will be $o(D)$. Therefore, the expected number of times that the walk will pass through a self-loop in the first $\Theta(TD)$ steps of the walk can be upper bounded by $o(TD)$. Therefore this prefix of the walk can be coupled with a simple random walk that crosses $\Theta(TD)$ edges of H. Since $\Theta(TD) = o(n)$, this implies that the number of distinct visited nodes is $\Omega(TD) = \Omega\left(t_{\mathrm{mix}}(\ ) \cdot d_{\mathrm{avg}}\right)$. □

4.5 A lower bound for MH

We finally give a lower bound for MH. We will show that, in some graphs having constant average degree (i.e., $d_{\mathrm{avg}} = O(1)$), MH requires a number of queries larger than $\Omega\left(t_{\mathrm{mix}} \cdot d_{\mathrm{max}}^{\frac{1}{2}-\epsilon}\right)$, for any choice of $d_{\mathrm{max}} = \Theta\left(n^c\right)$, with $0 < c < \frac{2}{3}$.

To prove the lower bound, construct the following graph $G = G_D$ on n nodes, for any $D = \lfloor n^c \rfloor$ for any $0 < c < \frac{2}{3}$. Start from a constant degree graph having constant conductance on node set V with $|V| = n$ (e.g., a random 3-regular graph on n nodes). Add $d = \left\lfloor \sqrt{D} \right\rfloor$ nodes $w_1, \dots, w_d$. Then, for each $i = 1, \dots, d$, select independently a random subset $S_i \subseteq V$ of cardinality $|S| = D - 1$ and for each $v_j \in S_i$, add the edge $\{w_i, v_j\}$ to the graph. Finally, add a final node u, and add the edge $\{u, w_i\}$ for each $i = 1, \dots, d$.

We now show the following (proof omitted in this version).

LEMMA 14. *G_D has the following properties: (i) $|V| = \Theta\left(n\right)$; (ii) $d_{\mathrm{avg}} = \Theta(1)$, $d_{\mathrm{max}} = D$, and $d_{\mathrm{min}} = \Theta(1)$; and $t_{\mathrm{mix}}(\ ) = \Theta(\log n)$.*

The upper bound on t_{mix} follows from Cheeger's inequality [4], and the fact that the conductance of G_D is a constant. We now prove the lower bound:

LEMMA 15. *MH will query at least $\Omega\left(\sqrt{D}\right)$ nodes before making a single step, if it starts the walk on the node u of G_D. Hence, there exists graphs of constant average degree and polynomial maximum degree on which MH queries $\Omega\left(t_{\mathrm{mix}} \cdot d_{\mathrm{max}}^{\frac{1}{2}-\epsilon}\right)$ nodes.*

PROOF. Let the walk start at u. Each neighbor of u has degree D, while u has degree $\Theta\left(\sqrt{D}\right)$. Therefore, for each neighbor of u, the probability that MH will move from u to that neighbor is at most $\Theta\left(D^{-1/2}\right)$. By a standard coupon collector's argument, MH will then have to query $\Theta(\sqrt{D})$ neighbors of u before finishing the first step. Since for this graph, $t_{\mathrm{mix}} \cdot d_{\mathrm{max}}^{\frac{1}{2}-\epsilon} = o(\sqrt{D})$, the claim follows. □

5. OTHER EXTENSIONS

In this section we analyze the complexity of sampling nodes for a more general family of distributions. We focus on sampling nodes according to Π^{α}, i.e., with probability proportional to $d(u)^{\alpha}$, for constant $\alpha > 0$. Besides it being a natural question, this has some practical applications, for example it can be used to obtain efficient estimators [12, 21] for the clustering coefficient or for the number of triangle (when $\alpha = 2$). In the remaining of this section we present extensions of rejection sampling, md sampling and MH sampling for this setting.

Let

$$S_{\alpha} = \sum_{u} d(u)^{\alpha}.$$

5.1 Rejection sampling

To sample a node u with probability proportional to $d(u)^{\alpha}$ we modify the rejection sampling algorithm as in Algorithm 2.

Algorithm 2 Generalization of rejection sampling (rej_{α}).

```
Input: Graph $G$, a starting node $u$, $\alpha$
  while no node is output do
    for $t_{\mathrm{mix}}(\ )$ steps do
      do a step of the uniform random walk ($P^{\mathrm{urw}}$)
    With probability $\frac{d(u)^{\alpha}}{C_{\alpha} d(u)}$, where $C_{\alpha} = \max(d_{\mathrm{min}}^{\alpha-1}, d_{\mathrm{max}}^{\alpha-1})$,
    output $u$ and stop
```

THEOREM 16. *To generate an -approximate sample from Π^{α}, the expected number of steps and queries are given by*

$$\mathtt{steps}(\mathsf{rej}_{\alpha}) = \mathtt{queries}(\mathsf{rej}_{\alpha}) = O\left(t_{\mathrm{mix}}(\ ) \cdot \frac{2mC_{\alpha}}{S_{\alpha}}\right).$$

PROOF. The proof follows the same line of the one of Theorem 1. In this case the probability of returning a sample can be bounded by:

$$\sum_{x \in V} \Pr[X = x] \frac{d(u)^{\alpha}}{C_{\alpha} d(u)} = \sum_{x \in V} \left(\frac{d(u)}{2m} \pm \ \right) \frac{d(u)^{\alpha}}{C_{\alpha} d(u)}$$

$$\geq$$

PROOF. Applying the variational characterization (2) to P^{md}_{α}, and using its definition, we obtain

$$1-\lambda_2(P^{\mathsf{md}_\alpha}) = \inf_f \frac{\sum_{u,v}(f(u)-f(v))^2\pi^{\mathsf{md}_\alpha}(u)P^{\mathsf{md}_\alpha}(u,v)}{\sum_{u,v}(f(u)-f(v))^2\pi^{\mathsf{md}_\alpha}(u)\pi^{\mathsf{md}_\alpha}(v)}$$
$$= \frac{S_\alpha}{\max(1, d_{\max}^{1-\alpha})} \cdot \inf_f \frac{\sum_{u,v}(f(u)-f(v))^2E(u,v)}{\sum_{u,v}(f(u)-f(v))^2d(u)^\alpha d(v)^\alpha}.$$

Hence,

$$1-\lambda_2(P^{\mathsf{md}}) \geq \quad \frac{S_\alpha \max(1, d_{\min}^{1-\alpha})}{2m\max(1, d_{\max}^{1-\alpha})\max(1, d_{\max}^{\alpha-1})}(1-\lambda_2(P^{\mathsf{urw}}))$$
$$\geq \quad \frac{S_\alpha}{2mC_\alpha\max(1, d_{\max}^{1-\alpha})}(1-\lambda_2(P^{\mathsf{urw}}))$$

Now, it then follows

$$t_{\mathrm{mix}}^{\mathsf{md}_\alpha}(\ ) \leq \frac{1}{1-\lambda_2(P^{\mathsf{md}_\alpha})}\left(\log n + \log\frac{1}{\ }\right)$$
$$\leq \frac{2mC_\alpha\max(1, d_{\max}^{1-\alpha})}{S_\alpha}\cdot\frac{1}{1-\lambda_2(P^{\mathsf{urw}})}\left(\log n + \log\frac{1}{\ }\right)$$
$$\leq \frac{2mC_\alpha\max(1, d_{\max}^{1-\alpha})}{S_\alpha}\cdot t_{\mathrm{mix}}(\ )\cdot c_5(n,\ ).$$

Note that in expectation after the underline urw is mixed we make $\sum_u P[X=u]d(u)^{\alpha-1}\max(1, d_{\max}^{1-\alpha}) = \frac{S_\alpha}{2m}\max(1, d_{\max}^{1-\alpha})$ steps for every query in expectation. So we have that if we run $\left(\frac{2m}{S_\alpha}\right)^2 C_\alpha\cdot t_{\mathrm{mix}}(\ )\cdot(c_5(n,\ )+1)$ queries the expected number of steps is at least
$\frac{2m}{S_\alpha}C_\alpha\max(1, d_{\max}^{1-\alpha})\cdot t_{\mathrm{mix}}(\ )\cdot c_5(n,\ )$. So by Markov inequality by running K times those many queries with probability at least $1-\frac{1}{K}$ we have taken enough steps. □

5.3 Metropolis–Hastings sampling

To sample a node u with probability proportional to $d(u)^\alpha$ using Metropolis–Hastings, we define a new random walk P^{MH_α} as follows:

$$P^{\mathsf{MH}_\alpha}(u,v) = \begin{cases} \min\left(\frac{1}{d(v)}, \frac{d(v)^\alpha}{d(u)^{\alpha+1}}\right) & \{u,v\}, \in E \\ 1 - \sum_{v\in\Gamma(u)} P^{\mathsf{MH}_\alpha}(u,v) & u = v, \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

Now we can upper bound the number of steps and the number of queries for MH_α. We state the following theorem without proof.

THEOREM 19. *To generate an -approximate sample from* Π^0, *the number of steps and queries needed are given by*

$$\texttt{steps}(\mathsf{MH}) \quad = \quad c_7(n,\ )\cdot\frac{2mC_\alpha d_{\max}}{S_\alpha d_{\min}^\alpha}\cdot t_{\mathrm{mix}}(\ ),$$

where $c_7(n,\ ) = c_6(\log n + \log(1/\ ))$ *for an appropriate constant* c_6.

6. EXPERIMENTS

In order to study the query complexity empirically, we formulated the following set of questions:

- How does the mixing rate of the three algorithms depend on the number of queries performed by the walk?
- How do the algorithms perform in terms of estimating basic network statistic, e.g., graph size, average degree, and clustering coefficient?

6.1 Datasets

We use the following publicly available datasets, all collected from `snap.stanford.edu`. Enron and Gnutella are undirected network generated from email-exchanges and peering relation in the p2p network. DBLP results from co-authorship of papers, while web-ND is the undirected version of the web graph of an university. Table 6.1 describes the basic statistics of these networks.

Dataset	Nodes	Edges
Enron	34K	180K
Gnutella	62K	148K
DBLP	317K	1049K
web-ND	325K	1117K

Table 1: Network statistics.

6.2 Algorithms

For each of these networks, we run random walks to try to sample from the target distribution Π^0 and Π^2. We run the random walks for rej and md for the two target distributions in the same manner as described in Algorithm 1 and (6) respectively. For each of the walks, we calculate the number of distinct query nodes touched, assuming that a node visited before could have been cached by the algorithm. For MH, we run two variants of the walk. The naive variant, denoted in figures as MH, penalizes all self-loops of MH as queries, since in order to reject a transition, the walk has to visit the proposed candidate in order to obtain its degree. The optimistic variant, denoted in figures as MH+, assumes that neighbor degrees are stored in each node, and so rejecting proposed candidates does not result in additional queries.

For each network, we run 200 random walks of each type, and for at most 1 million `steps`. Each walk (for each type) starts at the same node, but is otherwise independent. Note that since we want to examine empirical convergence rate, the results could possibly depend on the starting point, however, after examining 100 randomly chosen starting points, we did not observe any effective change in the plots and hence report the plots with only one arbitrary starting vertex.

Note that there are a number of heuristic convergence statistics that are often used in practice in order the judge whether a Markov chain has converged. These techniques, however, are useful mostly when the actual stationary distribution is not accessible.

6.3 Results

Figure 1 shows the $_1$ difference between Π^0 and the empirical frequency distributions of each of the walks, when the number of `queries` is as denoted by the x-axis. MH typically performs worse than the other alternatives, which are more or less similar to each other, with md being slightly better than rej in most cases. However, augmenting MH with the neighbor degree information (MH+) makes it converge faster than all the other alternatives.

Figure 1 shows the similar plot for the target Π^2. Note that in this case, even the naive MH often performs better than rej, in spite of the lower bound in Theorem 15. This leaves an open question of whether the lower bound can be refined in terms of a specific graph statistic which, although could be bad in the worst case, does have a "reasonable" value for real networks.

The $_1$ difference between distributions is an extremely strong criteria, and as we see in the above plots, it takes a significant number of steps for this difference to be sufficiently less than 1. Hence,

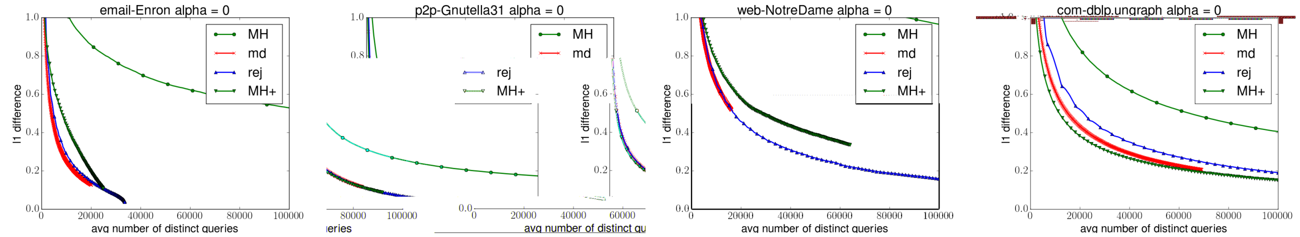

Figure 1: L1 difference from Π^0 for each of the four datasets 1) Enron 2) Gnutella 3) web-ND and 4) DBLP . X-axis is number of queries used.

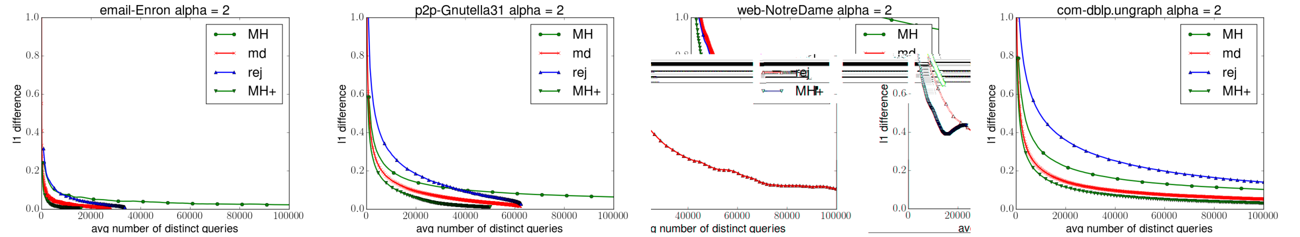

Figure 2: L1 difference from Π^2 for each of the four datasets 1) Enron 2) Gnutella 3) web-ND and 4) DBLP . X-axis is number of queries used.

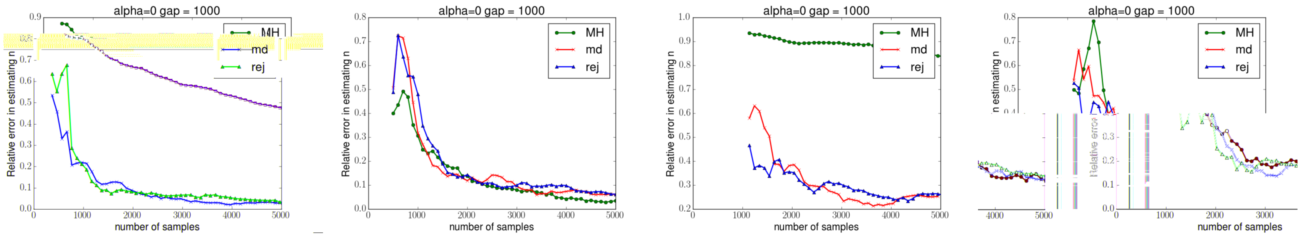

Figure 3: Relative error in estimating $|V|$ for each of the four datasets 1) Enron 2) Gnutella 3) web-ND and 4) DBLP . X-axis is number of queries used, sampled from the walks at a gap of 1000. Note that Y-axis are differently sized.

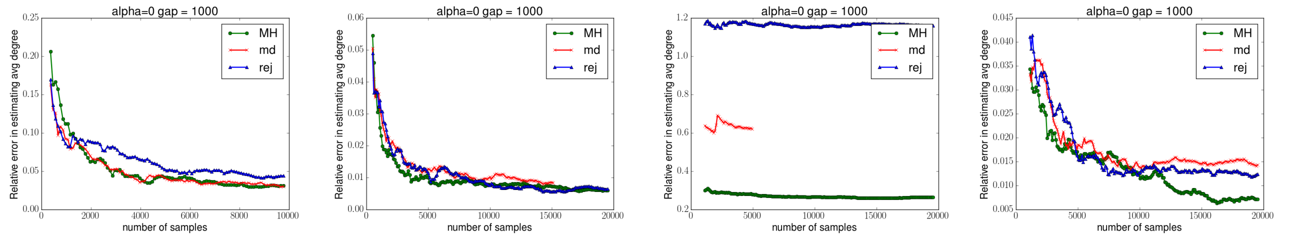

Figure 4: Relative error in estimating average degree for each of the four datasets 1) Enron 2) Gnutella 3) web-ND and 4) DBLP . X-axis is number of queries used. Note that Y-axis are differently sized.

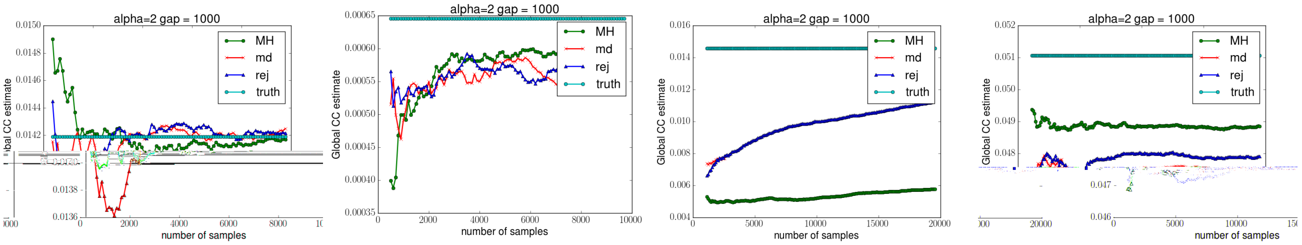

Figure 5: Global clustering coefficient estimate for each of the four datasets 1) Enron 2) Gnutella 3) web-ND and 4)DBLP . X-axis is number of queries used. The walk used was for Π^2. Note that Y-axis are differently sized.

we next compare the performance of the random walks in terms of the quality of samples that they provide after a fixed number of steps. In order to create these samples, we first discard an initial part of each walk. We then take samples after every 1000 queries have been performed by each walk.

In order to estimate the total size of the network, we use the established birthday paradox method. If in a (repeated) sample of size s, we get C collisions, then we estimate $n = |V|$ as $\hat{n} = \frac{s^2}{2C}$. Note that there are two types of errors in the random walk sampling—due to the walks not having converged, and due to the correlation in the samples. Since we discard an initial prefix, we mostly see the effect of the latter. Figure 3 shows the relative error ($\frac{|n-\hat{n}|}{n}$ for estimate $\hat{n}$) for sampling by each walk. Again, the behavior of rej and md are mostly same, with MH occasionally performing much worse than the others.

Estimating average degree is also easy once we have uniform sampling of nodes. Figure 4 plots the relative error incurred by the different walks. Other than the fact all the algorithms perform badly on web-ND , there is little to distinguish them in this metric.

The global clustering coefficient is defined as the ratio of the number of triangles to the number of length 2 paths. In order to estimate the global clustering coefficient, we use walks designed for Π^2 and use the same sampling method described above. Each such node then chooses a pair of neighbors and checks whether it is a triangle, (1/6th of) the fraction that forms a triangle is then returned. The y-axis in Figure 5 plots the absolute estimates rather than the errors. For the two smaller networks we do reach close the number of triangles. For the larger networks, there is a gap indicating that we need more samples for good relative error bound. Again, the performances of the different walks are more or less similar.

7. RELATED WORK

Generating a uniform node in a network has been studied in a number of papers. Most of the work, unfortunately, has been empirical and are not focused on the query complexity aspects. Gjoka et al. [7] modified the simple random walk to induce the uniform distribution on nodes as the stationary distribution; their main idea is to use a version of the Metropolis–Hastings algorithm for unbiasing. See also the work of Stutzbach et al. [23]. Bar-Yossef et al. [1, 2] proposed the md algorithm in order to unbias the distribution induced by the uniform random walk. However, they did not give theoretical bounds on the performance of md. Subsequent work such as that of Li et al. [17] obtained practical algorithms that perform better than MH and md, but once again did not analyze the performance of these algorithms in terms of their query complexity. Very recently, Nazi et al. [19] proposed to combine random walk with a proactive estimation step in order to reduce the long burn-in period typical with random walks.

Sampling in the context of network parameter estimation has been extensively studied in several papers. In particular, for the problem of estimating the size of the networks, sampling and collision counting are classical methods; see the work of Katzir et al. [14] and Hardiman and Katzir [9]. Another problem that has attracted a lot of attention is that of estimating clustering coefficients [9]. All of these works are based on simple uniform random walks on the network. Ye and Wu [24] also consider the social network size estimation problem; while they assume the ability to uniformly sample a node, it is easy to argue that this is an unrealistic assumption. Cooper, Radzik, and Siantos [5] also used random walk methods for estimating network parameters, but they go beyond collisions by actually using the return times for estimation. Ribeiro and Towsley [20] used multidimensional random walks for the sampling and estimation problems. Leskovec and Faloutsos [15] posed the problem of obtaining a representative sample of the network to approximate various properties of the original network such as average shortest path, centrality, etc.

Non-uniform sampling has also been used to estimate certain network parameters. The main reason is that they are sometimes provably more powerful than uniform sampling. Katzir et al. [13] gave a method to estimate the size of networks and size of subpopulations using degree-biased sampling (i.e., according to Π^1) and counting collisions in the samples collected. Dasgupta et al. [6] used degree-biased sampling to estimate the average degree of a graph. They showed that degree-biased sampling can improve the query complexity of estimating the average degree almost exponentially. In a recent work, Seshadri et al. [21] showed that sampling nodes with probability proportional to the square of the degree (i.e., Π^2) is a provably better way for estimating the number of triangles in a network; see also [12].

Zhou et al. [25] modify the simple random walk in order to reduce the mixing time of the walk. Boyd et al. [3] considered the problem of modifying edge probabilities to achieve the fastest mixing time; this turns out to be a semi-definite program. Even though both these works try to reduce the mixing time, it is unclear if they can of any help in improving the query complexity of sampling.

An excellent bibliography of sampling in networks can be found in `www.utdallas.edu/~emrah.cem/links.html`. A comprehensive survey on techniques to bound the mixing time of a Markov Chain is [8].

8. CONCLUSIONS

In this paper we studied the query complexity of sampling in a graph. Focusing on the uniform sampling question, we proved near-tight bounds for three popular random-walk based algorithms. The techniques we use to show the query complexity bound, via the variational inequality, might be of independent interest since typical analysis in random walks focuses more on the number of steps than on the number of queries. Our work poses a natural intriguing open problem: is the query complexity of sampling a uniform at random node $\Omega(d_{\rm avg} \cdot t_{\rm mix})$? An affirmative answer will essentially mean that one has to either make additional assumption on the input graph or resort to heuristics to get any improvements. A negative answer, which seems less plausible, would be very surprising!

Acknowledgments. We thank the reviewers for many useful suggestions.

9. REFERENCES

[1] Z. Bar-Yossef, A. C. Berg, S. Chien, J. Fakcharoenphol, and D. Weitz. Approximating aggregate queries about Web pages via random walks. In *VLDB*, pages 535–544, 2000.

[2] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. *J. ACM*, 55(5), 2008.

[3] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

[4] J. Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. In R. C. Gunning, editor, *Problems in Analysis (Papers dedicated to Salomon Bochner)*, pages 195–199. Princeton Univ. Press, 1970.

[5] C. Cooper, T. Radzik, and Y. Siantos. Estimating network parameters using random walks. In *CASoN*, pages 33–40, 2012.

[6] A. Dasgupta, R. Kumar, and T. Sarlós. On estimating the average degree. In *WWW*, pages 795–806, 2014.

[7] M. Gjoka, M. Kurant, C. Butts, and A. Markopoulou. Walking in Facebook: A case study of unbiased sampling of OSNs. In *INFOCOM*, pages 1–9, 2010.

[8] V. Guruswami. Rapidly mixing Markov chains: A comparison of techniques. *A Survey*, 2000.

[9] S. J. Hardiman and L. Katzir. Estimating clustering coefficients and size of social networks via random walk. In *WWW*, pages 539–550, 2013.

[10] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[11] M. Hildebrand. Random walks on random simple graphs. *Random Structures & Algorithms*, 8(4):301–318, 1996.

[12] M. Jha, C. Seshadhri, and A. Pinar. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *WWW*, pages 495–505, 2015.

[13] L. Katzir, E. Liberty, and O. Somekh. Estimating sizes of social networks via biased sampling. In *WWW*, pages 597–606, 2011.

[14] L. Katzir, E. Liberty, and O. Somekh. Framework and algorithms for network bucket testing. In *WWW*, pages 1029–1036, 2012.

[15] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *KDD*, pages 631–636, 2006.

[16] D. Levin, Y. Peres, and E. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

[17] R.-H. Li, J. X. Yu, L. Qin, R. Mao, and T. Jin. On random walk based graph sampling. In *ICDE*, pages 927–938, 2015.

[18] L. Lovász and P. Winkler. Efficient stopping rules for Markov chains. In *STOC*, pages 76–82, 1995.

[19] A. Nazi, Z. Zhou, S. Thirumuruganathan, N. Zhang, and G. Das. Walk, not wait: Faster sampling over online social networks. *PVLDB*, 8(6):678–689, 2015.

[20] B. Ribeiro and D. Towsley. Estimating and sampling graphs with multidimensional random walks. In *IMC*, pages 390–403, 2010.

[21] C. Seshadhri, A. Pinar, and T. G. Kolda. Wedge sampling for computing clustering coefficients and triangle counts on large graphs. *Statistical Analysis and Data Mining*, 7(4):294–307, 2014.

[22] A. Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, Probability, & Computing 1*, pages 351–370, 1992.

[23] D. Stutzbach, R. Rejaie, N. G. Duffield, S. Sen, and W. Willinger. On unbiased sampling for unstructured peer-to-peer networks. *IEEE/ACM Trans. Netw.*, 17(2):377–390, 2009.

[24] S. Ye and F. Wu. Estimating the size of online social networks. In *SocialCom*, pages 169–176, 2010.

[25] Z. Zhou, N. Zhang, Z. Gong, and G. Das. Faster random walks by rewiring online social networks on-the-fly. In *ICDE*, pages 769–780, 2013.