2004

Distributed Uniform Sampling in Real-World Networks

Asad Awan

Ronaldo A. Ferreira

Suresh Jagannathan
Purdue University, suresh@cs.purdue.edu

Ananth Y. Grama
Purdue University, ayg@cs.purdue.edu

Report Number:
04-029
DISTRIBUTED UNIFORM SAMPLING
IN REAL-WORLD NETWORKS

Asad Awan
Ronaldo A. Ferreira
Suresh Jagannathan
Ananth Y. Grama
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD TR #04-029
October 2004
Distributed Uniform Sampling in Real-World Networks

Asad Awan  Ronalda A. Ferreira  Suresh Jagannathan  Ananth Grama
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
Email: {awan,rf,suresh,ayg}@cs.purdue.edu

Abstract
Uniform sampling in networks is at the core of a wide variety of randomized algorithms. Random sampling can be performed by modeling the system as a graph with associated transition probabilities and defining a corresponding Markov chain (MC). A random walk of prescribed minimum length, performed on this graph, yields a stationary distribution, and the corresponding random sample. This sample, however, is not uniform when network nodes have a non-uniform degree distribution. This poses a significant practical challenge since typical large scale, real-world, unstructured networks tend to have non-uniform degree distributions, e.g., power-law degree distribution in unstructured peer-to-peer networks.

In this paper we present a distributed algorithm that enables efficient uniform sampling in large real-world networks. Specifically, we prescribe necessary conditions for uniform sampling in such networks and present distributed algorithms that satisfy these requirements. We empirically evaluate the performance of our algorithm in comparison to known algorithms. We also quantify, in context of the presented algorithms, the performance parameters in uniform sampling that are most relevant in a distributed setting—computational complexity, number of network messages, and the uniformity of the sampling. Detailed experimental results are used to support our claims relating to performance improvements of our algorithm.

1 Introduction
Uniform sampling in networks is an important substrate that provides the basis for a variety of randomized algorithms. These algorithms address problems such as leader election, duplicate elimination and controlled replication, search and routing, and group communications. The emergence of peer-to-peer (P2P) networks, where frequent node arrivals and departures make it difficult to maintain accurate network state, provide strong motivation for this class of algorithms. This paper addresses the critical problem of efficient distributed uniform sampling via random walks in large unstructured networks.

The uniform sampling problem can be formally defined as follows:

Definition 1.1 (Uniform random sampling) An algorithm samples uniformly at random from a set of nodes in a connected network if and only if it selects a node i belonging to the network, with probability 1/n, where n is the number of nodes in the network.

Notice that this problem is analogous to the problem of selecting a number uniformly at random in a given range. A trivial approach to this problem would be to collect the entire set of node identifiers at each node and index randomly into this table of identifiers. This simple approach, however, does not work for our target applications because the overhead of frequently updating system state at each node (if at all possible) would be extremely high. An alternate approach this problem relies on the notion of a random walk. Starting from an initial node, a random walk (of predetermined length) transitions through a sequence of intermediate nodes with probabilities defined for each link and ends at a destination node. The likelihood of terminating a random walk at any node determines whether the walk is a uniform sampling random walk or not. Formally, we define a uniform sampling random walk as follows:

Definition 1.2 (Uniform random sampling using random walk) A random walk of a given length samples uniformly at random from a set of nodes of a connected network if and only if the walk terminates at a node i belonging to the network, with probability 1/n, where n is the number of nodes in the network.

A number of researchers, over the years, have studied properties of random walks. Lovasz [11] provides an excellent survey on these techniques. The simplest random walk algorithm selects an outgoing edge at every node with
equal probability, e.g., if a node has degree four, each of the edges is traversed with a probability 0.25. It can be shown that the probability distribution associated with the second largest eigenvalue corresponds to a network topology parameter. These concepts are discussed in greater detail in Section 2. The main drawback of the simple random walk is that, while it reaches a stationary distribution, this distribution is not uniform for typical networks. In fact, it can be shown that the probability of terminating a random walk at a node is directly proportional to the degree of the node. In the context of conventional networks, where node degrees can vary significantly, this does not correspond to an acceptable uniform sample. In Figure 1, we plot the probability of terminating at a node on a power-law graph, with 30,000 nodes. Note that the variability in sampling is close to an order of magnitude!

Many applications of random walks are extremely sensitive to the quality of uniform sampling. Biases in sampling may result in poor performance of randomized algorithms, congestion in underlying networks, or sub-optimal utilization of storage resources. This provides the underlying motivation for our work. In addition to the quality of uniform sampling, a key performance parameter is the length of the random walk. Since the length of the random walk directly corresponds to the number of network messages, it is highly desirable to minimize the length of the walk. Consequently, the focus of this paper is on random walk techniques capable of uniform sampling, while minimizing the length of the walk. The paper makes the following specific contributions:

- It identifies sufficient and necessary conditions on a transition matrix for a uniform sampling random walk over the corresponding network.
- It presents an algorithm, called Random Weight Distribution (RWD), for achieving these conditions while reducing the length of the random walk.
- It provides detailed empirical evaluation of the performance characteristics of RWD in comparison to existing methods, namely Metropolis-Hastings (MH) and Maximum-Degree (MD).

The structure of this paper is as follows – in Section 2, we provide the background and foundations for our algorithm. In Section 3, we present necessary and sufficient conditions for uniform sampling. We also present two known algorithms and our new algorithm, which enables uniform sampling in irregular networks. In Section 4 we empirically evaluate the performance of our algorithm, and present a simulation-based comparison of the presented algorithms. In Section 5, we present related work, followed by conclusions in Section 6.

2 Background

In this section, we provide necessary background on random sampling using random walks in large networks. The abstraction of random walks as Markov chains is used to set up the notation and concepts that are used in the rest of the paper. Using a Markov chain model, we show, based on known results, that a simple random walk cannot achieve uniform sampling unless each node in the network has an identical number of connections. We also discuss various parameters that determine the length of the random walk required to achieve a stationary sample distribution.

Let $(G, E)$ be a simple connected undirected graph representing a distributed system with $|V| = n$ nodes and $|E| = m$ links. The degree, or number of links, of a node $i, 1 \leq i \leq n$, is given by $d_i$, and $\max_{1 \leq i \leq n} d_i$ denotes the maximum degree. The set of neighbors of a node $i$ is given by $\Gamma(i)$, where edge $(i,j) \in E, v_j \in \Gamma(i)$. The $n \times n$ adjacency matrix of $G$ is given by $A = \{a_{ij}\}$, where $1 \leq i, j \leq n, a_{ij} = 1$ if the edge $(i, j) \in E$, and $a_{ij} = 0$ otherwise. The corresponding $n \times n$ transition probability matrix is given by $P = \{p_{ij}\}$, where $0 \leq p_{ij} \leq 1$ is the probability of moving from node $i$ to node $j$ in one message hop (or time step). Furthermore, it is easy to see that $\sum_j p_{ij} = 1$, which implies that $P$ is a row-stochastic matrix.

Random walks: A simple random walk on $G$ is a sequence of nodes visited at each step of the walk. The transition from node $i$ to its neighbor is governed by the transition probability matrix $P$, where $p_{ij} = 1/d_i, p_{ij} = 0, v_j \not\in \Gamma(i)$.
The sequence of nodes can be denoted as \( \{X_t, X_{t+1}, \ldots\} \), where \( X_i \) implies that at step \( t \) the walk is at node \( i \).

If we consider nodes in a graph as states in a finite state space, then the random walk represents a discrete-time stochastic process, \( \{X_t\}_{t \geq 0} \). For this stochastic process we have,

\[
P_r(X_{t+1} = j | X_t = i) = p_{ij}
\]

(1)

Equation (1) simply implies that during a random walk the probability of moving to node \( j \) from node \( i \) in one step only depends on \( i \) and is independent of \( t \). This is known as the memoryless or Markov property. A random walk can be conveniently modeled as a Markov chain, more specifically a homogeneous Markov chain, since the right-hand side of Equation (1) is independent of \( t \). Such a Markov chain has the following properties: it is irreducible if the graph \( G \) is connected and is aperiodic if \( G \) is aperiodic. A graph \( G \) is aperiodic if the greatest common divisor of the length of all cycles in the graph is 1. In particular, an undirected aperiodic graph cannot be bipartite, which is a reasonable assumption for real networks in which connections are established randomly.

Equation (1) can be written more generally as \( p(x(t) = j | x(t-1) = i) = p_{ij} \), where \( p_{ij} \) is the transpose of the vector \( \{p_{ij} \} \) for the \( t \)-step probability transition matrix. Therefore, we have:

\[
p(x(t) = j | x(t-1) = i) = p_{ij}^t
\]

(2)

It is well known that an irreducible and aperiodic Markov chain has a stationary distribution \( \pi^* = \pi^* P \), and \( \pi^* = \pi^* P^t \) follows. It is easy to show \((15)\), page 132\()\), that \( \pi \), the component corresponding to node \( i \), \( 1 \leq i \leq n \), is \( \pi_i = d_i/2m \).

Eigenvalues of \( P \): From \( P^n \approx \pi^* P^n \) we see that \( \pi \) is a left eigenvector of \( P \) with eigenvalue 1. Also, \( \pi^* P = 1 \) (\( P \) is row-stochastic, and \( \pi \) is a vector with all entries equal to 1) implies that \( 1 \) is a right eigenvector with eigenvalue 1. It follows that \( P^n \approx \pi^* \). This implies that a very long walk converges to the stationary distribution \( \pi \) irrespective of the initial distribution. Since \( P \) is a non-negative primitive \( n \times n \) matrix (i.e., irreducible and aperiodic), from basic linear algebra we also know that \( P \) has \( n \) distinct eigenvalues \( \lambda_1 > \lambda_2 > \ldots > \lambda_n \) [13].

Random sampling: The above results indicate that a long enough random walk converges to a random sample irrespective of where the walk started. Thus, random walk is a good candidate for random sampling in a network. However, we also know that the resulting sample distribution is dependent on the degree of the node: \( \pi_i = d_i/2m \).

This last result implies that the random sample is uniform \((\pi_{\text{uniform}} = 1/n)\) only if the graph \( G \) is regular (i.e., the degrees of all nodes are equal). Since typical large-scale, real-world, unstructured networks tend to have non-uniform degree distributions (e.g., power-law degree distribution of unstructured P2P networks [19], and in-degree distributions due to irregular placement of sensors in a sensor network [4]), uniform sampling in practical scenarios poses a significant challenge.

Length of walk for random sampling: The sample distribution at step \( t \) of the walk depends on \( P^t \), which in turn depends on the eigenstructure of \( P \). From the Perron-Frobenius theorem, we have \( P^t = \lambda_1^t \pi^* (1 + o(m^{-1/2} \lambda_1^t)) \), where \( \lambda_1 \) is the right eigenvector corresponding to eigenvalue \( \lambda_1 \) and \( \pi^* \) is the left eigenvector, and \( o(\cdot) \) is the algebraic multiplicity of \( \lambda_1 \) (see, [3] Chapter 6). Rewriting the above equation, we have \( P^t = \lambda_1^t \pi^* + O(m^{-1/2} |\lambda_1|^t) \). These results simply imply that

\[
P^t = \lambda_1^t \pi^* + O(m^{-1} |\lambda_1|^t).
\]

(3)

As \( |\lambda_2| < 1 \), when \( t \) is large, \( |\lambda_2|^t \approx 0 \). Therefore, the smaller the second largest eigenvalue modulus (SLEM), the faster the convergence to stationary distribution. As a result, a walk of smaller length is required for random sampling. The number of steps required to converge to the stationary distribution is called the mixing time of the Markov chain.

3 Distributed Uniform Sampling Algorithms

In this section, we describe necessary and sufficient conditions for uniform sampling using random walks. We present known distributed algorithms, which change transition probabilities between neighboring nodes such that a random walk of a given minimum length can be used for uniform sampling. One of the shortcomings of these algorithms is that the minimum length of the random walk required to reach stationary distribution is significant. We present a new distributed algorithm, called Random Weight Distribution (RWD) that allows uniform sampling, while shortening the required minimum length of the random walk. We analytically show that our algorithm outperforms known distributed algorithms in terms of the mini- mum length of the random walk, and that the setup overhead of the algorithm is minimal.

3.1 Uniform Sampling via Random Walks

As mentioned in Section 1, a random walk of a given minimum length converges to a stationary distribution \( \pi \). If the stationary distribution \( \pi_{\text{uniform}} \) is such that \( \pi_{\text{uniform}} = (1/n) \), the random walk will terminate at any node in the network with equal probability (c.f. Definition 1.2).

\[\text{(1)}\]
To achieve a uniform stationary distribution in an irregular graph, we need to modify its probability transition matrix. Recall that, if the graph is not regular the probability transition matrix introduced for simple random walks will not suffice. As we shall see, it is straightforward to define probability transition matrices that have a stationary distribution.

Let $P$ be a probability transition matrix of a Markov chain, then $P_{\text{new}} = P_{\text{old}}^2 P$. Rewriting this as $(1/n)I = (1/n)P^2$, shows that $P = I$ for such a matrix. This means that the sum of columns vectors of $P$ is equal to 1 ($\sum_j P_{ij} = 1$) for each column $j$ of the matrix, i.e., $P$ is column stochastic. A probability transition matrix which is column stochastic in addition to being row stochastic is called doubly stochastic. Therefore, we can state that a doubly stochastic matrix has a stationary distribution $\pi_{\text{new}}$.

The following observation will be used to prove that the algorithms presented next result in uniform sampling random walks.

**Observation 3.1** Symmetric probability transition matrices ($P^T = P$) are doubly stochastic. $P$ is row stochastic because it is a probability transition matrix, and $P$ is column stochastic in virtue of symmetry. Therefore, a Markov chain defined over a symmetric probability transition matrix has a stationary distribution $\pi_{\text{new}}$.

### 3.2 Existing Distributed Random Walk Algorithms

#### 3.2.1 Maximum-Degree Algorithm (MD)

In the distributed adaptations of this algorithm, each node can perform a local computation to set up its transition probabilities. The main problem with the algorithm is that it requires the knowledge of the maximum degree, $\Delta_{\text{max}}$, among all nodes in the network. The maximum degree is a dynamic and global parameter of the network, and its dissemination to all nodes at run-time in a large distributed system is difficult. The algorithm sets up the transition matrix $P_{\text{MD}}$ as follows:

$$p_{ij}^{\text{MD}} = \begin{cases} 1/\Delta_{\text{max}} & \text{if } i \neq j \cap j \in \mathcal{T}(i) \\ 1 - d_i/\Delta_{\text{max}} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Note that the self-transition probability maintains the doubly-stochastic property of the matrix. In this algorithm, $p_{ij}^{\text{MD}} = p_{ji}^{\text{MD}} = 1/\Delta_{\text{max}}$, therefore, the resulting probability transition matrix is symmetric and doubly stochastic. It follows from Observation 3.1 that a random walk using these transition probabilities will select a node uniformly at random if the random walk is long enough.

#### 3.2.2 Metropolis-Hastings Algorithm (MH)

This algorithm is an adaptation for uniform sampling of the classical Metropolis-Hastings algorithm [14, 6, 2]. In this distributed algorithm each node $i$ sends a message, stating its degree, $d_i$, to each of its neighbors $j \in \mathcal{T}(i)$. Once this information is received from each of the neighbors, the transition probability matrix $P_{\text{MH}}$ is set up as follows:

$$p_{ij}^{\text{MH}} = \begin{cases} \min(1, d_i/d_{\text{max}}) & \text{if } i \neq j \cap j \in \mathcal{T}(i) \\ 1 - \sum_{j \neq i} (p_{ij}^{\text{MH}}) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

As in the previous algorithm, self-transition probability maintains the row-stochastic property. Clearly, $P_{\text{MH}}$ is symmetric and hence will enable uniform sampling via random walks (Observation 3.1).

#### 3.2.3 Performance

Conditioned on the non-uniformity of the number of links per node (i.e., high variance in degree distributions), both MD and MH might have high self-transition probabilities for nodes with low degrees. Comparing MD and MH algorithms, we can observe that the self-transition probability of the MH algorithm is lower-bounded by the self-transition probability of the MD algorithm. Intuitively, high self-transition probability implies that the length of the walk must be higher to attain sufficient mixing (a comprehensive experimental evaluation is provided in Section 4), otherwise the walk might be biased.
towards low degree nodes. This bias is certainly not desired because often times low degree has a correlation with low importance of the node in the network (for e.g., in P2P systems low degree nodes generally stay in the network for smaller periods of time and have fewer resources than higher degree nodes).

3.3 Random Weight Distribution Algorithm

In this section, we present our distributed algorithm, referred to as the Random Weight Distribution (RWD) algorithm. RWD is a completely decentralized algorithm that sets up transition probabilities in a connected network to enable efficient uniform sampling via random walks.

The algorithm proceeds as follows. In the initialization phase each node, locally, sets transitions probability as:

\[
p_{ij} = \begin{cases} 
\frac{1}{p} & \text{if } i \neq j, j \in V(i), \text{ where } p \geq d_{max} \\
1 - d_{i}/p & \text{if } i = j \\
0 & \text{otherwise}
\end{cases}
\]

Here, \( p \) is a static system parameter with the constraint that it should be greater than \( d_{max} \). This parameter is static because we can sufficiently overestimate \( d_{max} \), knowing system properties (e.g., popular P2P clients have a maximum connection limit [10]). Furthermore, as shown subsequently, overestimating \( d_{max} \) does not affect the performance of our algorithm. Note that this phase results in a high self-transition probability for low degree node. Also note that the resulting transition probability matrix is symmetric.

After the initialization is complete, each node attempts to distribute its self-transition probability randomly and symmetrically to its neighbors. The algorithm runs at each node in the network. In the following sections the terminology weight of a node refers to the self-transition probability of the node at any given time during the execution of the algorithm. \( j \) is a node, \( i \) is the algorithm terminates when either the weight of the node becomes zero or the weight of all nodes \( j \in V(i) \) becomes zero. The pseudo-code for the complete RWD algorithm is shown in Figure 2.

3.3.1 Discussion

A node \( i \) keeps a set \( N \) of its neighbors that have non-zero weights (self-transitions). It selects a neighbor \( j \) with equal probability from this set and sends an \textsc{increase} message. Selecting neighbors uniformly at random is an important characteristic of this algorithm and will be used later in its analysis. Node \( j \), on receiving the \textsc{increase} message, checks to see if its weight is greater than or equal to \( \delta \). \( \delta < 1 \) is a global quantum parameter. If the weight of \( j \) is greater than or equal to \( \delta \), it accepts the \textsc{increase} request, reduces its self-transition probability by \( \delta \), and increases the transition probability of the \((j,j)\) link by \( \delta \). Node \( i \) is notified of the success by an \textsc{ack} message. On receiving the \textsc{ack} message, node \( i \) reduces its self-transition probability and increases the transition probability of the \((i,j)\) link. Observe that after this operation the sums of transition probabilities at nodes \( i \) and \( j \) remain equal to one. Conversely, if the weight of node \( j \) is less than \( \delta \), it replays with a \textsc{decrease} message. On receiving the \textsc{decrease} message, node \( i \) removes node \( j \) from its set \( N \), and does not change its weight. Note that both operations preserve the symmetry of the transition probability on the link between \( i \) and \( j \). The following remark follows:

Remark 3.1: Each step in the RWD algorithm maintains symmetry in the global transition probability matrix \( p_{\text{global}} \). Therefore, the transition probability matrix remains symmetric when the algorithm terminates. Using Observation 3.1, we see that a random walk based on \( p_{\text{global}} \) will have stationary distribution \( \text{tr}_\text{global} \).
The termination condition of our algorithm, restated below, is used in the analysis in the subsequent section. **Termination condition:** the algorithm terminates if either the self-transition probability of the node becomes zero, i.e., it has no more weight to distribute, or if the set $N$ becomes empty, i.e., the weight of all of its neighbors is zero.

**Bound on the number setup of messages:** At each step in the algorithm the weight is reduced by $\delta$. This parameter determines how many INCREASE messages it takes for the algorithm to reach the final transition probabilities. The following lemma provides a bound on the number of messages.

**Lemma 3.1** The number of INCREASE messages is strictly less than $\left(1 - \frac{d_i}{\delta + d_i}\right)\delta + d_i$.

**Proof:** Note that irrespective of $\delta$, we can provide a strict bound on the number of messages as follows: $p_{d,\delta} < 1$ for the node to be connected (because there must be a non-zero transition probability to a neighboring node). Therefore, the number of INCREASE messages per node is strictly less than $1/(\delta + d_i)$. Note that $d_i$ is added because INCREASE messages which result in ACK messages also need to be counted. There can be at most $d_i$ such messages. Now using the knowledge that $p_{d,\delta} = 1 - d_i/p$, we can see that the number of INCREASE messages is strictly less than $\left(1 - \frac{d_i}{p}\right)/(\delta + d_i)$. The inequality is strict because the algorithm stops when either a node reduces its weight to zero, in which case there will be no ACKs and the $d_i$ term can be removed, or the node is not able to reduce its weight to zero and receives ACKs from all its neighbors. In the second case the number of messages is exactly $1 - \frac{d_i}{p} + \frac{d_i}{\delta + d_i}$, which is smaller than the given inequality. Observe that the total number of reply messages (ACK+NACK) is the same as the number of INCREASE messages.

Note that setting the value of quantum to be as low as 0.025 results in strictly less than 40 $\times$ $d_i$ increase messages per node. This can be considered a small constant overhead. Furthermore, the following optimization can be applied. We know that the neighboring nodes in real-world distributed systems frequently exchange messages, e.g., search queries, heartbeats, and ping-pong messages. All messages from a node are always routed through its neighbors. By reserving a few bits in each message, the communication required for our algorithm can be piggybacked on existing messages. This minimizes the network overhead of our algorithm.

**4 Experimental Results**

In this section, we study the performance of the Random Weight Distribution (RWD) algorithm in comparison with the distributed adaptations of the Maximum-Degree (MD) and Metropolis-Hastings (MH) algorithms. First, using exact calculation of SLEM (second-largest eigenvalue magnitude) and $r_{\text{rwd}}$ transition probability matrix evaluation (using matrix multiplication), we quantify that $p_{\text{rwd}}$ has a lower SLEM and hence shorter mixing time compared to $p_{\text{md}}$ and $p_{\text{mh}}$. We also use the RWD algorithm, with varying values of system parameters $\delta$ (increment quantum) and $\rho$ (inverse of initial edge transition weight). Next, we perform random walks using transition probability matrices computed by the three algorithms. In the rest of this section, we abbreviate the previous state--transition probability matrices computed by the three algorithms.

**4.1 Experimental Setup**

Our experiments are based on power-law topologies. In a power-law random graph the node degree distribution follows a power-law distribution, i.e., if the nodes are sorted in descending order of degree then the $i$-th node has degree $D_i = \delta i^\alpha$, where $\delta$ is a constant. Such graphs are often used in literature to model large non-uniform network topologies. For example, it is believed that P2P networks conform to such power-law topologies. The parameter $\alpha = 0.8$ is used for our results, unless stated otherwise. This value of $\alpha$ is popularly used in evaluation studies of P2P networks. The topology is constructed by first selecting the degree for each node using the power-law distribution and then connecting them randomly. Motivated by real-world systems [10], we limit the maximum degree of any node in the network to 100. In typical P2P clients such as Limewire [10], such restrictions are often applied to restrict the number of connections of a given node in order to limit the load on the node. For our simulation of random walks, we use 50,000 nodes in the network. To study the properties of the transition probability matrix, which involves the exact calculation via matrix multiplication of SLEM ($\mu$) and number of steps to convergence, we use a topology with 5,000 nodes.

**4.2 Convergence to Stationarity**

In this section we present our study of the characterization of the transition probability matrices, $p_{\text{rwd}}$, $p_{\text{md}}$, and
4.3 Length of Random Walks and Uniform Sampling

Figure 4. Characterization of $p_{\text{rew}}$ with varying values of $p$, with quantum $q$ fixed at 0.025. The values corresponding to the MH and MD algorithms are also shown.

In this section, we evaluate the effect of the length of random walks and the resulting uniformity of the samples. A practical hindrance for uniform sampling via random walks is that it is difficult to estimate the minimum required TTL or the length of the walk. As stated in [11], the random walk length necessary to achieve stationary distribution has order $O(\log N)$.

However, the constant in this bound is dependent on the SLEM of the network. As shown in Section 4.2, our algorithm achieves a lower SLEM and hence requires a shorter walk. For a 50,000 node network, we evaluate the constant associated with the required length of the walk for the transition probability matrix generated using RWD, MH, and MD algorithms. In [8], Horowitz and Malkhi propose an algorithm to estimate the network size using only local information. This algorithm provides a good estimation of the logarithm of the size of the network. Convergence to stationarity can be found using coupling methods [16]. Once, the length of the walk of a given random graph is known, it is not expected to change drastically for stable network topologies.

We evaluate the uniformity of the samples by running random walks of increasing length for each algorithm. As a measure of uniformity, we calculate the standard deviation from the expected probability $1/N = 1/50000 = 0.00002$ (c.f. Definition 12). As a comparison, we also present corresponding results for a uniform sample generated using the C function `drand48()`. We use random walks of lengths different values of $p$, using our algorithm. The values observed for the ME and MH algorithms are also plotted as horizontal lines. We see that our algorithm performs better for all values of $p$. However, $p > 150$ yields poorer performance compared to that with higher values of $p$. Note that it is required that $p > d_{\text{max}}$. If the value of $p$ is close to $d_{\text{max}}$, the algorithm has less degree of freedom in the weight distribution phase. Therefore, simply overestimating $p$ is a good heuristic for using our algorithm.

In Figure 3, we show the plots of the resulting eigenvalues and steps to convergence when using different values of the quantum. The values observed for the MD and MH algorithm are also plotted as horizontal lines. We see that our algorithm performs better, for all but one value of quantum, in terms of the required number of steps to reach convergence. Furthermore, note that the trend for the required length of walk follows the trend of the SLEM plot. The value of quantum for which our algorithm is outperformed by the other two algorithms is 0.1. The reason for this failure is that such a high quantum leads our algorithm to local maxima when the self-transition probabilities are distributed. Secondly, because the value of quantum is fixed at 0.1, self-transition probabilities, which are just slightly lower than 0.1 remain as such. Note that by choosing quantum as 0.1 versus, for example, 0.025 saves at most 30 messages per node irrespective of $p$. This is a small saving, especially if the messages are piggybacked on existing system messages, in which case the saving is negligible. Thus, a quantum set to 0.025 represents a good trade-off between number of messages and efficiency of the algorithm.

Figure 3. Characterization of $p_{\text{rew}}$ with varying values of quantum, while $p$ is fixed at 200. The values corresponding to the MH and MD algorithms are also shown.
Figure 5. The standard deviation vs length of walk for RWD, MH and MD over a 50,000 node graph. The standard deviation of exact uniform sampling is also provided as a reference.

Figure 8. MH converges to uniform distribution when the length of the walk is $10 \log n$ and MD converges when the length is $20 \times \log n$. As a comparison, note that RWD converges to uniform distribution when the length of the random walk is only $5 \times \log n$.

4.4 Number of Network Messages

While the minimum length of the walk, i.e., the TTL necessary for uniform sampling is a good indicator of the performance of RWD in a distributed setting, it is not obvious that it directly implies low network message overhead in comparison with the other two algorithms. Recall that in a random walk, TTL is decremented even if a self-transition is made. We evaluate the network message overhead of the random walks that achieve uniform sampling for RWD, MH, and MD. The results are summarized in Table 1. We note that RWD requires only 38 network messages out of a walk of total length 54, MH algorithm requires 51 messages out of a walk of length 109, while MD requires 57 messages out of 216. The key observation is that the absolute value of the network messages involved in uniform sampling in MH and MD is much higher than for RWD, despite the fact that network messages constitute only a small portion of their walks. Although, the total length of the random walk does not translate exactly to network messages, it has associated computational overhead, which may be significant if the nodes that have low processing capability, e.g., in sensor networks.

We repeat the above experiments using network topologies with varying levels of non-uniformity. This is done by changing the value of $a$ to 0.7, 0.9, 1.1, and 1.2. The results follow the same trend as discussed above but are not included here due to limitations of space.

As a final remark, the experimental results show the efficacy of RWD algorithm in terms of its non-biased sampling; shorter length of the walk, and fewer network messages required for sampling.
Figure 6. Random sampling achieved with a walk of length $3 \times \log n$ for RWD, MH and MD.

Figure 7. Random sampling with a walk of length $5 \times \log n$ for RWD, MH and MD. Note that RWD achieves a random sample with probability close to uniform ($\text{Unif}_{n/5000} = 1/5000$).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Network Msgs</th>
<th>Percentage of total walk</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWD</td>
<td>38</td>
<td>70.4</td>
</tr>
<tr>
<td>MH</td>
<td>51</td>
<td>46.8</td>
</tr>
<tr>
<td>MD</td>
<td>57</td>
<td>21.6</td>
</tr>
</tbody>
</table>

Table 1. Network messages corresponding to random walk lengths used for uniform sampling.

5. Related Work

Structured peer-to-peer (P2P) networks [21, 22, 18, 13, 17] provide strong guarantees for search by imposing a well-defined topology on the network. Peers and objects are assigned hash-based identifiers and objects are assigned to peers based on these identifiers. Objects in the network are found by performing efficient routing protocols that lead to the peer responsible for storing pointers to the objects. Node identifiers, generated via a hash function, are assumed to be uniformly distributed in the identifier space. A simple algorithm for choosing a random peer in these networks would be to select a random number $u$ in the identifier space and route to the node responsible for $u$. King and Saia [9] show that this simple algorithm leads to biased samples. To solve this problem, they propose a more robust algorithm that always chooses each peer with probability exactly $1/n$ and that has $O(\log n)$ expected message complexity.

In [5], Gkantsidis et al. perform an extensive study of random walks in P2P networks. The authors explore the performance of random walks for searching and uniform sampling. For searching, the authors show that random walks perform better than flooding when the length of the random walk is the same as the number of peers covered by flooding with bounded TTL. Another important result in the paper is that it is possible to simulate the selection of a uniform sample of elements by performing a random walk of required length on a particular class of network topologies. The results presented, however, apply only to expander graphs, where the degree of the nodes is constant. This is in contrast with our work, since our focus is on general topologies, including power-law graphs.

Boyd et al. [2] formulate the problem of finding the fastest mixing Markov chain on a graph as a convex optimization. The problem is expressed as a semidefinite program (SDP), and the solution of the SDP yields the global optimal probabilities in the transition matrix. They also show that the Metropolis-Hasting and the maximum degree algorithms are substantially slower than the optimum. While this technique is useful for finding the optimal mixing time, it cannot be directly used in a distributed setting because of the overheads associated with solving the SDP.

Uniform sampling has also been investigated in different...
large-scale applications. In [7], for example, Henzinger et al. propose a method for sampling web pages with near uniform distribution. The algorithm uses a random walk, with the edge weights changed according to the page rank of the pages. Page rank is a measure of the popularity of a web page and is used by search engines to rank search results. The sample produced by the algorithm is not truly uniform and appears biased toward web pages with high numbers of inbound links. Sampling from web pages, however, poses additional challenges when compared with sampling in a network, since the connections form a directed graph with a non-symmetric adjacency matrix. In [1], Bash et al. investigate the problem of approximating a uniform random sample in sensor networks. Their algorithm uses geographic routing and Voronoi diagrams. While this approach is applicable to sensor networks, it does not directly apply to unstructured peer-to-peer networks, where geographic routing is not feasible.

6 Conclusion

In this paper we address key challenges in uniform sampling using random walks. Based on a Markov chain abstraction we describe the necessary conditions for uniform sampling in non-uniform networks, and examine two known algorithms that result in uniform random sampling via random walks. We present a new algorithm called Random Weight Distribution (RWD), which results in uniform sampling while minimizing the length of the random walk. Using a study of real data, we support claims of superior performance of our algorithm compared to existing algorithms, using parameters most relevant in a distributed setting.

References