Node Sampling using Random Centrifugal Walks

Andrés Sevilla[a,*], Alberto Mozo[a], Antonio Fernández Anta[b]

[a] *Dpto. Sistemas Informáticos, Universidad Politécnica de Madrid, Madrid, Spain*
[b] *Institute IMDEA Networks, Madrid, Spain*

Abstract

A distributed algorithm is proposed for sampling networks, so that nodes are selected by a special node (source), with a given probability distribution. We define a new class of random walks, that we call Random Centrifugal Walks (RCW). A RCW starts at the source and always moves away from it.

The algorithm assumes that each node has a weight, so the nodes are selected with a probability proportional to its weight. It requires a preprocessing phase before the sampling of nodes. This preprocessing is done only once, regardless of the number of sources and the number of samples taken from the network. The length of RCW walks are bounded by the network diameter.

The RCW algorithms that do not require preprocessing are proposed for grids and networks with regular concentric connectivity, for the case when the probability of selecting a node is a function of its distance to the source.

Keywords: Node sampling, Random walks, Randomized algorithms, Distributed algorithms

1. Introduction

Sampling a network with a given distribution has been identified as a useful operation in many contexts. For instance, sampling nodes with uniform probability is the building block of epidemic information spreading [14, 13]. Similarly, sampling with a probability that depends on the distance to a given node [3, 20] is useful to construct small world network topologies [16, 7, 2]. Other applications that can benefit from distance-based node sampling are landmark-less network positioning systems like NetICE9 [19], which does sampling of nodes with special properties to assign synthetic coordinates to nodes. In a different context, currently there is an increasing interest in obtaining a representative (unbiased) sample from the users of online social networks [9]. In this paper we propose a distributed algorithm for sampling networks with a desired probability distribution.

*Corresponding author
Email addresses: asevilla@eui.upm.es (Andrés Sevilla), amozo@eui.upm.es (Alberto Mozo), antonio.fernandez@imdea.org (Antonio Fernández Anta)

1.1. Related Work

One technique to implement distributed sampling is to use gossiping between the network nodes. Jelasity et al. [13] implemented a uniform sampling service using gossip-based epidemic algorithms. Kermarrec et al. [15] analyze a generic peer uniform sampling service with small views and independence. Bertier et al. [2] implement uniform sampling and DHT services using gossiping. As a side result, they sample nodes with a distribution that is close to Kleinberg's harmonic distribution (one instance of a distance-dependent distribution). Another gossip-based sampling service that gets close to Kleinberg's harmonic distribution has been proposed by Bonnet et al. [3]. As far as we know, there is no gossip-based sampling algorithm that is able to sample with an arbitrary probability distribution. Moreover, when using gossip-based uniform distributed sampling as a service, it has been shown by Busnel et al. [5] that only partial independence (ϵ-independence) between views (the subsets of nodes held at each node) can be guaranteed without re-executing the gossip algorithm. They show that, in order to achieve ϵ-independence between two consecutive samples at the same node, at least $\Omega(\log(1/\epsilon))$ shuffle rounds must be performed. Each shuffle round involves exchanging $\Theta(n)$ messages (where n is the network size). Gurevich and Keidar [11] give an algorithm that achieves ϵ-independence between uniform samples in $O(ns \log n)$ transformations (i.e., shuffle operations), even in the presence of messages loss, where s is the view size.

Another popular distributed technique to sample a network is the use of random walks [23]. Most random-walk based sampling algorithms do uniform sampling [1, 9], usually having to deal with the irregularities of the network. Sampling with arbitrary probability distributions can be achieved with random walks by re-weighting the hop probabilities to correct the sampling bias caused by the non-uniform stationary distribution of the random walks. Lee et al. [17] proposed two new algorithms based on Metropolis-Hastings (MH) random walks for sampling with any probability distribution. These algorithms provide an unbiased graph sampling with a small overhead, and a smaller asymptotic variance of the resulting unbiased estimators than generic MH random walks.

Sevilla et al. [20] have shown how sampling with an arbitrary probability distribution can be done without communication if a uniform sampling service is available. In that work, as in all the previous approaches, the desired probability distribution is reached when the stationary distribution of a Markov process is reached. The number of iterations (or hops of a random walk) required to reach this situation (the warm-up time) depends on the parameters of the network and the desired distribution, but it is not negligible. For instance, Zhong and Sheng [23] found by simulation that, to achieve no more than 1% error, in a torus of 4096 nodes at least 200 hops of a random walk are required for the uniform distribution, and 500 hops are required for a distribution proportional to the inverse of the distance. Similarly, Gjoka et al. [10] show that a MHRW sampler needs about 6K samples (or 1000-3000 iterations) to obtain the convergence to the uniform probability distribution. In the light of these results, Markovian

approaches seem to be inefficient to implement a sampling service, specially if multiple samples are desired.

1.2. Contributions

In this paper we present efficient distributed algorithms to implement a sampling service. The basic technique used for sampling is a new class of random walks that we call *Random Centrifugal Walks* (RCW). A RCW starts at a special node, called the *source*, and *always* moves *away* from it.

All the algorithms proposed here are instances of a generic algorithm that uses the RCW as basic element. This generic RCW-based algorithm works essentially as follows. A RCW always starts at the source node. When the RCW reaches a node x (the first node reached by a RCW is always the source s), the RCW stops at that node with a *absorption probability*. If the RCW stops at node x, then x is the node selected by the sampling. If the RCW does not stop at x, it jumps to a neighbor of x. To do so, the RCW chooses only among neighbors that are farther from the source than the node x. (The probability of jumping to each of these neighbors is not necessarily the same.) In the rest of the paper we will call all the instances of this generic algorithm as *RCW algorithms.*

Firstly, we propose a RCW algorithm that samples *any* connected network with *any* probability distribution (given as weights assigned to the nodes). Before starting the sampling, a preprocessing phase is required. This preprocessing involves building a minimum distance spanning tree (MDST) in the network[1], and using this tree for efficiently aggregating the node's weights. As a result of the weight aggregation, each node has to maintain one numerical value per link, which will be used by the RCW later. Once the preprocessing is completed, any node in the network can be the source of a sampling process, and multiple independent samplings with the exact desired distribution can be efficiently performed. Since the RCW used for sampling follow the MDST, they take at most D hops (where D is the network diameter).

Secondly, when the probability distribution is distance-based and the nodes are at integral distances from the source, RCW algorithms without preprocessing (and only a small amount of state data at the nodes) are proposed. In a *distance-based probability distribution* all the nodes at the same distance from the source node are selected with the same probability. (Observe that the uniform and Kleinberg's harmonic distributions are special cases of distance-based probability distributions.) In these networks, each node at distance $k > 0$ from the source has neighbors (at least) at distance $k - 1$. We can picture nodes at distance k from the source as positioned on a ring at distance k from the source. The center of all the rings is the source, and the radius of each ring is one unit larger than the previous one. Using this graphical image, we refer the networks of this family as *concentric rings networks.*

[1] Using, for instance, the algorithm proposed by Bui et al. [4] whose time complexity is $O(n)$ and $O(n \cdot m)$ message complexity.

This concentric rings topology can be naturally found in real networks. For instance, consider a wireless sensor network in which each node has a fixed known position assigned (e.g., via GPS). Then, fixing a source node, the nodes in the kth concentric rings can be the nodes whose (Euclidean) distance to the source is in the interval $(k-1, k]$. If the communication radius is reasonably large, the requirements of the concentric rings topology model will be satisfied.

The first distance-oriented RCW algorithm we propose samples with a distance-based distribution in a network with grid topology. The grid topology has been identified as an efficient deployment pattern in wireless sensor networks [22]. In this network topology, the source node is at position $(0, 0)$ and the lattice (Manhattan) distance is used. This grid contains all the nodes that are at a distance no more than the radius R from the source (the grid has hence a diamond shape[2]). The algorithm we derive assigns an absorption probability to each node, that only depends on its distance from the source. However, the hop probabilities depend on the position (i, j) of the node and the position of the neighbors to which the RCW can jump to. We formally prove that the desired distance-based sampling probability distribution is achieved. Moreover, since every hop of the RCW in the grid moves one unit of distance away from the source, the sampling is completed after at most R hops.

We have proposed a second distance-oriented RCW algorithm that samples with distance-based distributions in concentric rings networks *with uniform connectivity*. These are networks in which all the nodes in each ring k have the same number of neighbors in ring $k-1$ and the same number in ring $k+1$. Like the grid algorithm, this variant is also proved to finish with the desired distribution in at most R hops, where R is the number of rings.

Unfortunately, in general, concentric rings networks have no uniform connectivity. This case is faced by creating, on top of the concentric rings network, an overlay network that has uniform connectivity. In the resulting network, the algorithm for uniform connectivity can be used. We propose a distributed algorithm that, if it completes successfully, builds the desired overlay network. We have found via simulations that this algorithm succeeds in building the overlay network in a large number of cases (see Table 1).

In summary, RCW can be used to implement an efficient sampling service because, unlike previous Markovian (e.g., classical random walks and epidemic) approaches, (1) it always finishes in a number of hops bounded by the network diameter, (2) selects a node with the *exact probability distribution*, and (3) does not need warm-up (stabilization) to converge to the desired distribution. Additionally, in the case that preprocessing is needed, this only has to be executed once, independently on the number of sources and the number of samples taken from the network.

For instance, gossiping algorithms require a time complexity of $\Omega(\log(1/\epsilon))$ rounds in which a total number of $\Omega(n \log(1/\epsilon))$ messages are exchanged to

[2]A RCW algorithm for a square grid can be derived from the one presented.

obtain two consecutive ϵ-independent uniform samples at a given node [5]. On the other hand, RCW can start multiple independent samples in parallel, if desired, and each sample involves at most $D + 1$ rounds and messages.

The rest of the paper is structured as follows. In Section 2 we introduce concepts and notation that will be used in the rest of the paper. In Section 3 we present the RCW algorithm for a connected network. In Sections 4, 5, 6 and 7 we describe the RCW algorithm on grids and concentric rings networks with and without uniform connectivity. In Section 8 we present the simulation results for all described algorithms. Finally, we conclude the paper in Section 9.

2. Definitions and Model

2.1. Connected Networks

In this paper we only consider connected networks. This family includes most of the potentially interesting networks we can find. In every network, we use N to denote the set of nodes and $n = |N|$ the size of that set. When convenient, we assume that there is a special node in the network, called the *source* and denoted by s. We assume that each node $x \in N$ has an associated weight $w(x) > 0$. Furthermore, each node *knows* its own weight. The weights are used to obtain the desired probability distribution p, so that the probability of selecting a node x is proportional to $w(x)$. Let us denote $\eta = \sum_{j \in N} w(j)$. Then, the probability of selecting $x \in N$ is $p(x) = w(x)/\eta$. (In the simplest case, weights are probabilities, i.e., $w(x) = p(x), \forall x$ and $\eta = 1$.)

As mentioned, in order to use RCW to sample connected networks, some preprocessing is done. This involves constructing a spanning tree in the network and performing a weight aggregation process. After the preprocessing, RCW is used for sampling. A RCW starts from the source. When the RCW reaches a node $x \in N$, it selects x as the sampled vertex with probability $q(x)$, which we call the *absorption probability*. If x is not selected, a neighbor y of x in the tree is chosen, using for that a collection of *hop probabilities* $h(x, y)$. The values of $q(x)$ and $h(x, y)$ are computed in the preprocessing and stored at x. The probability of reaching a node $x \in N$ in a RCW is called the *visit probability*, denoted $v(x)$.

2.2. Concentric Rings Networks

We also consider a subfamily of the connected networks, which we call *concentric rings networks.* These are networks in which the nodes of N are at integral distances from s. In these networks, no node is at a distance from s larger than a radius R. For each $k \in [0, R]$, we use $\mathbb{R}_k \neq \emptyset$ to denote the set of nodes at distance k from s, and $n_k = |\mathbb{R}_k|$. (Observe that $\mathbb{R}_0 = \{s\}$ and $n_0 = 1$) These networks can be seen as a collection of concentric rings at distances 1 to R from the source, which is the common center of all rings. For that reason, we call the set $\mathbb{R}_k$ the *ring at distance* k. For each $x \in \mathbb{R}_k$ and $k \in [1, R]$, $\gamma_k(x) > 0$ is the number of neighbors of node x at distance $k - 1$ from s (which is only 1

if $k = 1$), and $\delta_k(x)$ is the number of neighbors of node x at distance $k+1$ from s (which is 0 if $k = R$).

The concentric rings networks considered must satisfy the additional property that the probability distribution is *distance based*. This means that, for all $k \in [0, R]$, every node $x \in \mathbb{R}_k$ has the same probability p_k to be selected. We assume that each node $x \in \mathbb{R}_k$ knows its own p_k. These properties allow, in the subfamilies defined below, to avoid the preprocessing required for connected networks.

2.2.1. Grids

A first subfamily of concentric rings networks considered is the grid with lattice distances. In this network, the source is at position $(0, 0)$ of the grid, and it contains all the nodes (i, j) so that $i, j \in [-R, R]$ and $|i| + |j| \leq R$. For each $k \in [0, R]$, the set of nodes in ring k is $\mathbb{R}_k = \{(i, j) : |i|+|j| = k\}$. The neighbors of a node (i, j) are the nodes $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$ (that belong to the grid).

2.2.2. Uniform Connectivity

The second subfamily considered is the set of concentric rings networks with *uniform connectivity*. These networks satisfy that

$$\forall k \in [1, R], \forall x, y \in \mathbb{R}_k, \delta_k(x) = \delta_k(y) \wedge \gamma_k(x) = \gamma_k(y). \quad (1)$$

In other words, all nodes of ring k have the same number of neighbors δ_k in ring $k+1$ and the same number of neighbors γ_k in ring $k-1$.

The behavior of a generic RCW was already described. In the algorithm that we will present in this paper for concentric rings networks we guarantee that, for each k, all the nodes in $\mathbb{R}_k$ have the same visit probability v_k and the same absorption probability q_k. A RCW starts from the source. When it reaches a node $x \in \mathbb{R}_k$, it selects x as the sampled vertex with absorption probability q_k. If x is not selected, a neighbor $y \in \mathbb{R}_{k+1}$ of x is chosen.

The desired *distance-based probability distribution* is given by the values p_k, $k \in [0, R]$, where it must hold that $\sum_{k=0}^{R} n_k \times p_k = 1$. The problem to be solved is to define the absorption and hop probabilities so that the probability of a node $x \in \mathbb{R}_k$ is p_k.

Observation 1. *If for all $k \in [0, R]$ the visit v_k and absorption q_k probabilities are the same for all the nodes in $\mathbb{R}_k$, the RCW samples with the desired probability iff $p_k = v_k \cdot q_k$.*

3. Sampling in a Connected Network

In this section, we present a RCW algorithm that can be used to sample any connected network. As mentioned, in addition to connectivity, it is required that each node knows its own weight. A node will be selected with probability proportional to its weight.

```
1   task Weight_Aggregation(i)
2   if i is a leaf then
3     send WEIGHT(w(i)) to neighbor x
4     receive WEIGHT(p) from neighbor x
5     T_i(x) ← p
6   else
7     repeat
8       receive WEIGHT(p) from x ∈ neighbors(i)
9       T_i(x) ← p
10      foreach y ∈ neighbors(i) \ {x} do
11        if received WEIGHT(·) from neighbors(i) \ {y}
12        then
13          w ← WEIGHT(w(i) + Σ_{z∈neighbors(i)\{y}} T_i(z))
14          send w to y
15      end foreach
16    until received WEIGHT(·) from every x ∈ neighbors(i)
```

Figure 1: Weight aggregation algorithm. Code for node i.

3.1. Preprocessing for the RCW Algorithm.

The RCW algorithm for connected networks requires some preprocessing which will be described now. This preprocessing has to be done only once for the whole network, independently of which nodes act as sources and how many samples are taken.

3.1.1. Building a spanning tree

Initially, the algorithm builds a spanning tree of the network. A feature of the algorithm is that, if several nodes want to act as sources for RCW, they can all share the same spanning tree. Hence, only one tree for the whole network has to be built. The algorithm used for the tree construction is not important for the correctness of the RCW algorithm, but the diameter of the tree will be an upper bound on the length of the RCW (and hence possibly on the sampling latency). There are several well known distributed algorithms (see, e.g., [6] and the references therein) that can be used to build the spanning tree. In particular, it is interesting to build a minimum diameter spanning tree (MDST) because, as mentioned, the length of the RCW is upper bounded by the tree diameter. There are few distributed algorithms in the literature to build a MDST. One possible candidate to be used in our context is the one proposed by Bui et al. [4]. Additionally, if link failures are expected, the variation of the former algorithm proposed by Gfeller et al. [8] can be used.

3.1.2. Weight aggregation

Once the spanning tree is in place, the nodes compute and store aggregated weights using the algorithm of Figure 1. The algorithm executes at each node $i \in N$, and it computes in a distributed way the aggregated weight of each subtree that can be reached following one of the links of i. In particular, for each node x that is in the set of neighbors of i in the tree, $neighbors(i)$, the

algorithm computes a value $T_i(x)$ and stores it at i. Let (i, x) be a link of the spanning tree, then by removing the link (i, x) from the spanning tree there are two subtrees. We denote by $stree(x, i)$ the subtree out of them that contains node x.

Theorem 1. *After the completion of the Weight Aggregation algorithm (of Figure 1), each node $i \in N$ will store, for each node $x \in neighbors(i)$, in $T_i(x)$ the value $\sum_{y \in stree(x,i)} w(y)$.*

PROOF. Consider $stree(x, i)$ a tree rooted at x. We prove the claim by induction in the depth of this tree. The base case is when the tree has depth 1. In this case x is a leaf and, from the algorithm, it sends to i its weight $w(x)$, which is stored at i as $T_i(x)$. If the depth is $k > 1$, by induction hypothesis x ends up having in $T_x(y)$ the sum of the weight of the subtree $stree(x, y)$, for each $y \in neighbors(x) \setminus \{i\}$. These values plus $w(x)$ are added up and sent to i, which stores the resulting value as $T_i(x)$.

The values $T_i(x)$ computed in this preprocessing phase will later be used by the RCW algorithm to perform the sampling. The complexity of this process in terms of messages exchanged is $2(n - 1)$, and the time complexity in rounds is D. We assume that all nodes start running the Weight Aggregation algorithm simultaneously, that the transmission of messages takes one step, and that computation time is negligible.

Theorem 2. *The Weight Aggregation algorithm (of Figure 1) requires $2(n-1)$ messages to be exchanged, and completes after D steps, where D is the diameter of the tree.*

PROOF. It is easy to observe in the algorithm that one message is sent across each link in each direction. Since all spanning trees have $n - 1$ links, the first part of the claim follows.

The second claim can be shown to be as follows. Let us consider any node i as the root of the spanning tree. Let d be the largest distance in the tree of any node from i. We show by induction on k that all nodes at distance $d - k$ from i have received the aggregated weight of their corresponding subtrees by step k. The base case is $k = 1$, which follows since the leaves at distance d send their weights to their parents in the first step. Consider now any (non-leaf) node j at distance $d-k+1$ from i. Assume that y is the parent (at distance $d-k$) of j in the tree rooted at i. By induction hypothesis j has received all the aggregated weights of the subtrees by step $k - 1$. Then, when the latest such value was received from a neighbor x (Line 8), the foreach loop (Lines 10-13) is executed. In this execution, the condition of the if statement at Line 11 is satisfied for y. Then, the aggregated weight $w(j) + \sum_{z \in neighbors(j) \setminus \{y\}} T_j(z)$ is sent to y by step $k - 1$. This value reaches y in one step, by step k. Then, i receives all the aggregated weights by step d. Since the largest value of d is D, the proof is complete.

1 **task** $RCW(i)$
2 **when** $RCW\_MSG(s)$ **received from** x
3 $candidates \leftarrow neighbors(i) \setminus \{x\}$
4 **with** *probability* $q(i) = \frac{w(i)}{w(i)+\sum_{z \in candidates} T_i(z)}$ **do**
5 *select node* i *and report to source* s
6 **otherwise**
7 *choose a node* $y \in candidates$
8 **with** *probability* $h(i,y) = \frac{T_i(y)}{\sum_{z \in candidates} T_i(z)}$
9 **send** $RCW\_MSG(s)$ *to* y

Figure 2: RCW algorithm for connected networks. Code for node i.

3.2. RCW Sampling Algorithm

In this RCW algorithm (Figure 2) any node can be the source. The spanning tree and the precomputed aggregated weights are used by any node to perform the samplings (as many samples as needed). The sampling process in the RCW algorithm works as follows. To start the process, the source s sends a message $RCW\_MSG(s)$ to itself. When the $RCW\_MSG(s)$ message is received by a node i from a node x, it computes a set of candidates for next hop in the RCW, which are all the neighbors of i except x. Then, the RCW stops and selects that node with an *absorption probability* $q(i) = \frac{w(i)}{w(i)+\sum_{z \in candidates} T_i(z)}$ (Line 4). If the RCW does not select i, it jumps to a neighbor of i different from x. To do so, the RCW chooses only among nodes y in the set of candidates (that move away from s) using $h(i,y) = \frac{T_i(y)}{\sum_{z \in candidates} T_i(z)}$ as hop probability (Line 8).

3.3. Analysis

We show now that the algorithm proposed performs sampling with the desired probability distribution.

Theorem 3. *Each node $i \in N$ is selected by the RCW algorithm with probability $p(i) = \frac{w(i)}{\eta}$.*

Proof. If a node i receives the $RCW\_MSG(s)$ from x, let us define $candidates = neighbors(i) \setminus \{x\}$, and $T(i) = w(i) + \sum_{z \in candidates} T_i(z)$. We prove the following stronger claim: Each node $i \in N$ is visited by the RCW with probability $v(i) = \frac{T(i)}{\eta}$ and selected by the RCW algorithm with probability $p(i) = \frac{w(i)}{\eta}$.

We prove this claim by induction on the number of hops from the source s to node i in the spanning tree. The base case is when the node i is the source s. In this case x is also s, $candidates = neighbors(s)$, and $T(s) = \eta$. Hence, $v(s) = \frac{T(s)}{\eta} = 1$ and $q(s) = \frac{w(s)}{\eta}$, yielding $p(s) = \frac{w(s)}{\eta}$.

The induction hypothesis assumes the claim is true for a node x at distance k from s, and proves the claim for i which is at distance $k+1$. We have that $\Pr[\text{visit } i] = v(x)\,(1-q(x))\,\frac{T(i)}{T(x)-w(x)}$, where $1-q(x)$ is the probability of not selecting node x when visiting it, and $\frac{T(i)}{T(x)-w(x)}$ is the probability of

choosing the node i in the next hop of the RCW. The absorption probability of x and i are $q(x) = w(x)/T(x)$ and $q(i) = w(i)/T(i)$, respectively (Line 4). Then, $v(i) = \frac{T(x)}{\eta}\left(1 - \frac{w(x)}{T(x)}\right)\frac{T(i)}{T(x)-w(x)} = \frac{T(x)}{\eta}\left(\frac{T(x)-w(x)}{T(x)}\right)\frac{T(i)}{T(x)-w(x)} = \frac{T(i)}{\eta}$ and $\Pr[\text{select } i] = v(i)q(i) = \frac{T(i)}{\eta}\frac{w(i)}{T(i)} = \frac{w(i)}{\eta}$.

4. Sampling in a Grid

If the algorithm for connected networks is applied to a grid, given its regular structure, the construction of the spanning tree could be done without any communication among nodes, but the weight aggregation process has to be done as before. However, we show in this section that all preprocessing and the state data stored in each node can be avoided if the probability distribution is based on the distance. RCW sampling process was described in Section 2, and we only redefine absorption and hop probabilities. From Observation 1, the key for correctness is to assign absorption and hop probabilities that guarantee visit and absorption probabilities that are homogenous for all the nodes at the same distance from the source.

4.1. Absorption probability

For $k \in [0, R]$, the absorption probability of every node $(i, j) \in \mathbb{R}_k$ is defined as

$$q_k = \frac{n_k \cdot p_k}{\sum_{j=k}^{R} n_j \cdot p_j} = \frac{n_k \cdot p_k}{1 - \sum_{j=0}^{k-1} n_j \cdot p_j}. \tag{2}$$

As required by Observation 1, all nodes in $\mathbb{R}_k$ have the same q_k. Note that $q_0 = p_0$ and $q_R = 1$, as one may expect. Since the value of p_k is known at $(i, j) \in \mathbb{R}_k$, n_k can be readily computed[3], and the value of $\sum_{j=0}^{k-1} n_j \cdot p_j$ can be piggybacked in the RCW, the value of q_k can be computed and used at (i, j) without requiring precomputation nor state data.

4.2. Hop probability

In the grid, the hops of a RCW increase the distance from the source by one unit. We want to guarantee that the visiting probability is the same for each node at the same distance, to use Observation 1. To do so, we need to observe that nodes (i, j) over the axes (i.e., with $i = 0$ or $j = 0$) have to be treated as a special case, because they can only be reached via a single path, while the other nodes can be reached via several paths. To simplify the presentation, and since the grid is symmetric, we give the hop probabilities for one quadrant only (the one in which nodes have both coordinates non-negative). The hop probabilities in the other three quadrants are similar. The first hop of each RCW chooses one of the four links of the source node with the same probability 1/4. We have three cases when calculating the hop probabilities from a node (i, j) at distance k, $0 < k < R$, to node (i', j').

[3] $n_0 = 1$, while $n_k = 4k$ for $k \in [1, R]$.

- *Case A:* The edge from (i,j) to (i',j') is in one axis (i.e., $i = i' = 0$ or $j = j' = 0$). The hop probability of this link is set to $h_k((i,j),(i',j')) = \frac{i+j}{i+j+1} = \frac{k}{k+1}$.
- *Case B:* The edge from (i,j) to (i',j') is not in the axes, $i' = i+1$, and $j' = j$. The hop probability of this link is set to $h_k((i,j),(i+1,j)) = \frac{2i+1}{2(i+j+1)} = \frac{2i+1}{2(k+1)}$.
- *Case C:* The edge from (i,j) to (i',j') is not in the axes, $i' = i$, and $j' = j+1$. The hop probability of this link is set to $h_k((i,j),(i,j+1)) = \frac{2j+1}{2(i+j+1)} = \frac{2j+1}{2(k+1)}$.

It is easy to check that the hop probabilities of a node add up to one.

4.3. Analysis

Below we prove that the RCW that uses the above absorption and hop probabilities selects nodes with the desired sample probability.

Lemma 4. *All nodes at the same distance $k \geq 0$ to the source have the same visit probability v_k.*

PROOF. The proof uses induction. The base case is $k = 0$, and obviously $v_k = 1$. When $k = 1$, the probability of visiting each of the four nodes at distance 1 from the source s is $v_i = \frac{1-q_0}{4}$, where $1-q_0$ is the probability of not staying at source node. Assuming that all nodes at distance $k > 0$ have the same visit probability v_k, we prove the case of distance $k+1$. Recall that the absorption probability is the same q_k for all nodes at distance k.

The probability to visit a node $x = (i',j')$ at distance $k+1$ depends on whether x is on an axis or not. If it is in one axis it can only be reached from its only neighbor (i,j) at distance k. This happens with probability (case A) $\Pr[\text{visit } x] = v_k(1-q_k)\frac{i+j}{i+j+1} = v_k(1-q_k)\frac{k}{k+1}$. If x is not on an axis, it can be reached from two nodes, $(i'-1,j')$ and $(i',j'-1)$, at distance k (Cases B and C). Hence, the probability of reaching x is then $\Pr[\text{visit } x] = v_k(1-q_k)\frac{2(i'-1)+1}{2(i'+j')} + v_k(1-q_k)\frac{2(j'-1)+1}{2(i'+j')} = v_k(1-q_k)\frac{k}{k+1}$. Hence, in both cases the visit probability of a node x at distance $k+1$ is $v_{k+1} = v_k(1-q_k)\frac{k}{k+1}$. This proves the induction and the claim.

Theorem 5. *Every node at distance $k \in [0,R]$ from the source is selected with probability p_k.*

PROOF. If a node is visited at distance k, it is because no node was selected at distance less than k, since a RCW always moves away from the source. Hence, $\Pr[\exists x \in \mathbb{R}_k \text{ visited}] = 1 - \sum_{j=0}^{k-1} n_j p_j$. Since all the n_k nodes in $\mathbb{R}_k$ have the same probability to be visited (see the previous lemma), we have that $v_k = \frac{1-\sum_{j=0}^{k-1} n_j p_j}{n_k}$. Now, since all the n_k nodes in $\mathbb{R}_k$ have the same absorption probability is q_k, the probability of selecting a particular node x at distance k

from the source is $\Pr[\text{select } x] = v_k q_k = \frac{1-\sum_{j=0}^{k-1} n_j p_j}{n_k} \frac{n_k p_k}{\sum_{j=k}^{R} n_j p_j} = p_k$, where it has been used that $(1 - \sum_{j=0}^{k-1} n_j p_j) = \sum_{j=k}^{R} n_j p_j$.

5. Sampling in a Concentric Rings Network with Uniform Connectivity

In this section we derive a RCW algorithm to sample a concentric rings network with uniform connectivity, where all preprocessing is avoided, and only a small (and constant) amount of data is stored in each node. Recall that uniform connectivity means that all nodes of ring k have the same number of neighbors δ_k in ring $k+1$ and the same number of neighbors γ_k in ring $k-1$.

5.1. Distributed algorithm

The general behavior of the RCW algorithm for these networks was described in Section 2. In order to guarantee that the algorithm is fully distributed, and to reduce the amount of data a node must know a priori, a node at distance k that sends the RCW to a node in ring $k+1$ piggybacks some information. In more detail, when a node in ring k receives the RCW from a node of ring $k-1$, it also receives the probability v_{k-1} of the previous step, and the values p_{k-1}, n_{k-1}, and δ_{k-1}. Then, it calculates the values of n_k, v_k, and q_k. After that, the RCW algorithm uses the absorption probability q_k to decide whether to select the node or not. If it decides not to select it, it chooses a neighbor in ring $k+1$ with uniform probability. Then, it sends to this node the probability v_k and the values p_k, n_k, and δ_k, in the message of the RCW.

The RCW algorithm works as follows. The source s selects itself with probability $q_0 = p_0$. If it does not do so, it chooses one node in ring 1 with uniform probability, and sends it the RCW message with values $v_0 = 1$, $n_0 = 1$, p_0, and δ_0. Figure 3 shows the code of the RCW algorithm for nodes in rings $\mathbb{R}_k$ for $k > 0$. Each node in ring k must only know initially the values δ_k, γ_k and p_k. Observe that n_k (number of nodes in ring k) can be locally calculated as $n_k = n_{k-1}\delta_{k-1}/\gamma_k$. The correctness of this computation follows from the uniform connectivity assumption (Eq. 1).

5.2. Analysis

The uniform connectivity property can be used to prove by induction that all nodes in the same ring k have the same probability v_k to be reached. The absorption probability q_k is defined as $q_k = p_k/v_k$. Then, from Observation 1, the probability of selecting a node x of ring k is $p_k = v_k q_k$. What is left to prove is that the value v_k computed in Figure 3 is in fact the visit probability of a node in ring k.

Lemma 6. *The values v_k computed in Figure 3 are the correct visit probabilities.*

```
1   task RCW(x, k, δ_k, γ_k, p_k)
2   when RCW_MSG(s, v_{k-1}, p_{k-1}, n_{k-1}, δ_{k-1}) received:
3       n_k ← n_{k-1} δ_{k-1}/γ_k
4       v_k ← n_{k-1} (v_{k-1} − p_{k-1})/n_k
5       q_k ← p_k/v_k
6       with probability q_k do select node x and report to s
7       otherwise
8           choose a neighbor y in ring k + 1
9                       with uniform probability
10          send RCW_MSG(s, v_k, p_k, n_k, δ_k) to y
```

Figure 3: RCW algorithm for concentric rings with uniform connectivity (code for node $x \in \mathbb{R}_k$, $k > 0$).

PROOF. Let us use induction. For $k = 1$ the visit probability of a node x in ring $\mathbb{R}_1$ is $\frac{1-q_0}{n_1} = \frac{1-p_0}{n_1}$. On the other hand, when a message RCW_MSG reaches x, it carries $v_0 = 1$, $n_0 = 1$, p_0, and δ_0 (Line 2). Then, v_1 is computed as $v_1 = n_0 \frac{v_0 - p_0}{n_1} = \frac{1-p_0}{n_1}$ (Line 4). For a general $k > 1$, assume the value v_{k-1} is the correct visit probability of a node in ring $k-1$. The visit probability of a node in ring k is $v_{k-1} n_{k-1}(1 - q_{k-1})/n_k$, which replacing $q_{k-1} = p_{k-1}/v_{k-1}$ yields the expression used in Figure 3 to compute v_k (Line 4).

The above lemma, together with the previous reasoning, proves the following.

Theorem 7. *Every node at distance k of the source is selected with probability p_k.*

6. Sampling in a Concentric Rings Network with Uniform Connectivity Up to Distance 2

In this section we generalize RCW for concentric rings with uniform connectivity to allow that each node could have neighbors at distances 1 and 2. Intuitively, this generalization will reduce on average the number of steps taken in each random walk. Then, we assume that two nodes in ring k have the same number of neighbors in rings $k-2$, $k-1$, $k+1$, and $k+2$. Also, as before we assume that each node has at least one neighbor in the rings at distance 1.

In this case a node in ring k that receives the RCW and is not selected has to decide whether to send the walk to a neighbor at distance 1 or at distance 2 (among the neighbors at each distance it uses uniform probability). To decide on this, we use a parameter $s_k \in (0, 1]$ that is the probability of choosing a neighbor at distance 1 and $(1 - s_k)$ is the probability of choosing a neighbor at distance 2. The set of values s_k has to be large enough so that the desired probability distribution can be achieved (in particular, it must guarantee that the visit probability v_k is no smaller than the selection probability p_k).

```
1  task RCW(x, k, δ_k, γ_k, p_k, s_k, δ_{k+1}, γ_{k+1}, p_{k+1}, s_{k+1})
2  when RCW_MSG(s, v_{k-1}, p_{k-1}, n_{k-1}, δ_{k-1}, s_{k-1},
3                          v_{k-2}, p_{k-2}, n_{k-2}, δ_{k-2}, s_{k-2}) received:
4      n_k  ← n_{k-1} δ_{k-1}/γ_k
5      v_k  ← n_{k-1} (v_{k-1} − n_{k-1} p_{k-1})/n_k s_{k-1} +
6             n_{k-2} (v_{k-2} − n_{k-2} p_{k-2})/n_k (1 − s_{k-2})
7      q_k  ← p_k/v_k
8
9      with probability q_k do select node x and report to s
10     otherwise
11       if Distance (s_k) = 1 then
12         choose a neighbor y in ring k + 1
13                          with uniform probability
14         send RCW_MSG(s, v_k, p_k, n_k, δ_k, s_k,
15                          v_{k-1}, p_{k-1}, n_{k-1}, δ_{k-1}, s_{k-1})
16       else
17         choose a neighbor y in ring k + 2
18                          with uniform probability
19         n_{k+1}  ← n_k δ_k/γ_{k+1}
20         v_{k+1}  ← n_k (v_k − p_k)/n_{k+1}
21         send RCW_MSG(s, v_{k+1}, p_{k+1}, n_{k+1}, δ_{k+1}, s_{k+1}
22                                  , v_k, p_k, n_k, δ_k, s_k)
```

Figure 4: RCW algorithm for concentric rings with uniform connectivity Up to Distance 2 (code for node $x \in \mathbb{R}_k$, $k > 0$).

6.1. Analysis

As in Section 5, the absorption probability is set to $q_k = p_k/v_k$. The expression to compute v_k now depends on parameters that can come from the previous two rings. When the RCW is sent to a node at ring k, must contain the values this node needs to compute v_k, namely n_{k-1}, v_{k-1}, p_{k-1}, s_{k-1}, n_{k-2}, v_{k-2}, p_{k-2}, and s_{k-2}. If the message is sent to a node at distance 2 (Line 19), the sender (at ring $k-2$) must compute those values that it does not know (like v_{k-1}) before sending them.

Lemma 8. *The values v_k computed in Figure 4 are the correct visit probabilities.*

Proof. The proof using induction is very similar to the proof of Lemma 6. For $k = 1$ the visit probability of a node x in ring $\mathbb{R}_1$ is $\frac{1-q_0}{n_1} = \frac{1-p_0}{n_1}$. On the other hand, when a message RCW_MSG reaches x, it carries $n_0 = 1$, $v_0 = 1$, p_0, $s_0 = 1$, $n_{-1} = 0$, $v_{-1} = 0$, $p_{-1} = 0$, and $s_{-1} = 0$. Then, v_1 is computed as $v_1 = n_0 \frac{v_0 - p_0}{n_1} s_0 + n_{-1} \frac{v_{-1} - p_{-1}}{n_1}(1 - s_{k-1}) = \frac{1-p_0}{n_1}$ (Line 6). For a general $k > 1$, assume the values v_{k-1} and v_{k-2} are the correct visit probabilities of a node in ring $k-1$ and $k-2$ respectively. The visit probability of a node in ring

```
1  function AssignAttachmentPoints(x, k)
2  ap ← LCM(n_k, n_{k+1}) / n_k
3  C ← N_{k+1}(x) /* neighbors of x in ring k + 1 */
4  A_x ← ∅ /* A_x is a multiset */
5  loop
6    choose c from C
7    send ATTACH_MSG to c
8    receive RESPONSE_MSG from c
9    if RESPONSE_MSG = OK then
10     ap ← ap − 1
11     add c to A_x /* c can be in A_x several times */
12   else C ← C \ {c}
13 until (ap = 0) ∨ (C = ∅)
14 if (ap = 0) then return A_x
15 else return FAILURE
```

Figure 5: Assignment Attachment Points (AAP) Function.

k is $(v_{k-1}n_{k-1}(1-q_{k-1})s_{k-1} + v_{k-2}n_{k-2}(1-q_{k-2})(1-s_{k-2}))/n_k$, replacing $q_{k-1} = p_{k-1}/v_{k-1}$ and $q_{k-2} = p_{k-2}/v_{k-2}$ yields the expression used in Figure 4 to compute v_k (Line 6).

7. Concentric Rings Networks without Uniform Connectivity

In general, concentric rings networks do not guarantee uniform connectivity. For instance, two nodes in a ring k may have different number of neighbors in rings $k-1$ or $k+1$. (Although it is expected that the nodes' degrees follow roughly a normal probability distribution if randomly deployed in the rings.) As a consequence, the RCW algorithm for uniform connectivity can not be used directly, since it would show a biased behavior. We propose now an algorithm that creates an *overlay* uniform connectivity on top of the existing connectivity between rings.

7.1. The Assignment Attachment Points (AAP) Algorithm

To eliminate the errors observed when there is no uniform connectivity, we propose a simple distributed algorithm to transform the concentric rings network without uniform connectivity into an overlay network with uniform connectivity.

To preserve the property that the visit probability is the same for all the nodes in a ring, nodes will use different probabilities for different neighbors. Instead of explicitly computing the probability for each neighbor, we will use the following process. Consider rings k and $k+1$. Let $r = LCM(n_k, n_{k+1})$, where LCM is the *least common multiple* function. We assign $\frac{r}{n_k}$ *attachment points* to each node in ring k, and $\frac{r}{n_{k+1}}$ attachment points to each node in ring $k+1$. Now, the problem is to connect each attachment point in ring k to a different attachment point in ring $k+1$ (not necessarily in different nodes). If

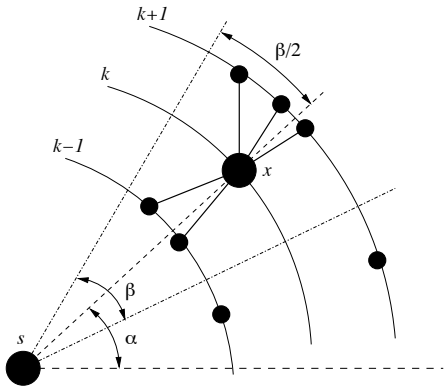

Figure 6: Node deployment and connectivity used in the simulations.

this can be done, we can use the algorithm of Figure 3, but when a RCW is sent to the next ring, an attachment point (instead of a neighbor) is chosen uniformly. Since the number of attachments points is the same in all nodes of ring k and in all nodes of ring $k+1$, the impact in the visit probability is that it is again the same for all nodes of a ring.

The connection between attachment points can be done with the simple algorithm presented in Figure 5, in which a node x in ring k contacts its neighbors to request available attachment points. If a neighbor that is contacted has some free attachment point, it replies with a response message *RESPONSE_MSG* with value *OK*, accepting the connection. Otherwise it replies to x notifying that all its attachment points have been connected. The node x continues trying until its $\frac{r}{n_k}$ attachment points have been connected or none of its neighbors has available attachment points. If this latter situation arises, then the process failed. The algorithm finishes in $O(\max_k\{n_k\})$ communication rounds. (Note that $r \leq n_k \cdot n_{k+1}$ and that, initially, C is $\mathbb{N}_{k+1}(x)$ and hence $|C| = n_{k+1}$.) Combining these results with the analysis of Section 5, we can conclude with the following theorem.

Theorem 9. *Using attachment points instead of links and the distributed RCW-based algorithm of Figure 3, it is possible to sample a concentric rings network without uniform connectivity with any desired distance-based probability distribution p_k, provided that the algorithm of Figure 5 completes (is successful) in all the nodes.*

In general, the algorithm of Figure 5 may not be successful. However, it can be shown that it is always successful if there is uniform connectivity. In fact, the success of AAP does not depend on the random decisions made in the algorithm, but on structural properties of the connectivity between all consecutive rings. Let us abuse the notation and use $\mathbb{N}_{k+1}(W)$ to denote the neighbors from $\mathbb{R}_{k+1}$ of the nodes in set $W \subseteq \mathbb{R}_k$. Recall that, when connecting rings k and $k+1$, each

Angle	% success
15 °	0%
30 °	0%
45 °	3%
60 °	82%
75 °	99%
90 °	100%
150 °	100%
180 °	100%
360 °	100%

Table 1: Success rate of the AAP algorithm as a function of the connectivity angle.

node in ring k is assigned $\frac{r}{n_k}$ attachment points, while each node in ring $k+1$ is assigned $\frac{r}{n_{k+1}}$ attachment points. Then, we characterize the cases in which AAP is going to be successful as described in the following theorem, which is a direct application of Hall's Theorem [12].

Theorem 10. *Given a concentric rings network, the AAP algorithm is successful and finishes in* $O(\max_k\{n_k\})$ *rounds iff* $\forall k \in [1, R-1], \forall W \subseteq \mathbb{R}_k$, $\frac{|W|}{n_k} \leq \frac{|\mathbb{N}_{k+1}(W))|}{n_{k+1}}$.

In order to evaluate empirically the success of the algorithm, we use a geometric deployment. A node x in ring k is assigned a position in the ring. This position can be given by an angle α. Then, each network studied will have associated a connectivity angle β, the same for all nodes. This means that x will be connected to all the nodes in rings $k-1$ and $k+1$ whose position (angle) is in the interval $[\alpha - \beta/2, \alpha + \beta/2]$ (see Figure 6). Observe that the bigger the angle β is, the more neighbors x has in rings $k-1$ and $k+1$.

It is shown in Table 1 the success rate of the algorithm for different connectivity angles. It can be observed that the success rate is high as long as the connectivity angles are not very small (at least 60°). (For an angle of 60° the expected number of neighbors in the next ring for each node is less than 17.) For small angles, like 15° and 30°, the AAP algorithm is never successful.

For the cases in which AAP fails, the algorithm for connected network presented in Section 3 can be used. Observe that, the complexity of building the spanning tree rooted at the source s is $S \in \Omega(D)$,[4] where D is the largest distance from s to any node. On the other hand, independently of whether it succeeds, the AAP algorithm finishes in $O(\max_k\{n_k\})$ rounds. Therefore, if $S \in \omega(\max_k\{n_k\})$, then it is worth trying AAP before using the approach of Section 3, due to the potential savings.

[4]For instance, the algorithm of Bui et al. [4] is linear in the number of rounds.

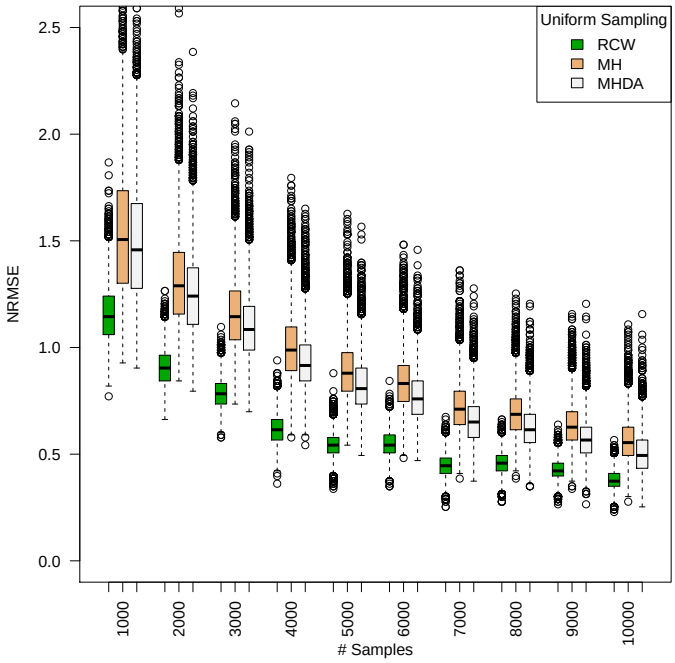

Figure 7: The NRMSE average over all possible degrees for RCW, MH and MHDA algorithms sampling on the AS-733 graph.

8. RCW Simulation Results

In this section, we present the simulation results of the RCW algorithm when sampling is done on the previously described topologies (connected networks, grid and concentric rings). We implement two sampling distributions: uniform (UNI) and proportional to the inverse of the distance (PID). The objective of these simulations is to measure, in each topology, the RCW error with respect to a uniform and a PID random sampling simulator. Also, we compare the RCW algorithm against some well known Markovian based algorithms.

8.1. Connected Networks

In this experiment, we compare the RCW algorithm with the Markovian approach described in [17]. Firstly, we use the normalized root mean square error (NRMSE), described in that paper, to compare the algorithms. Secondly, we use the relative error to obtain a more accurate evaluation of the RCW algorithm error. To perform these experiments we choose a real-world network dataset used in [17]. This network is the undirected graph *AS-733* [18], with 6474 nodes and 13233 edges. Each node represents an AS (Autonomous System) and the edges show the node peering relationships. In this graph, the maximum degree and the diameter are 1460 and 9, respectively.

8.1.1. Comparing the RCW algorithm with the MH and MH-DA algorithms

In this subsection, we compare RCW algorithm with the classical MH algorithm [21] and the MH-DA algorithm described in [17]. Before using the RCW algorithm, we do the required preprocessing (see Figure 1) for building a spanning tree and running a weight aggregation process on this graph. For the graph of the dataset *AS-733*, we obtain a spanning tree of diameter 11.

We use the NRMSE in order to compare the accuracy of the RCW sampling with the MH and MH-DA algorithms. The NRMSE for the source node i is defined as

$$E_i(d,r) = \frac{\sqrt{(\mathbb{E}(d,r) - (\frac{n_d}{n})^2}}{\frac{n_d}{\sum_j n_j}} \tag{3}$$

where d is the degree of node i, r is the number of samples, n_d is the number of nodes with degree d, and $\mathbb{E}(d,r)$ is the expected value of r samples with degree equal to d. The average NRMSE for all possible degrees is defined as

$$E_i(r) = \sum_d E_i(d,r)\frac{n_d}{n} \tag{4}$$

To obtain the NRMSE for every graph node, we use the RCW, MH and MH-DA algorithms to collect samples starting from each graph node. In Figure 7, we show the NRMSE average for different numbers of samples per node, ranging from $1,000$ to $10,000$.

These results show that the RCW algorithm outperforms, in accuracy, the MH and MH-DA algorithms. The RCW algorithm reaches a mean NRMSE error below 1.0 requiring only $2,000$ samples, while the MH and MH-DA algorithms need at least $6,000$ samples to obtain similar results. Moreover, RCW does not need warm-up to offer samples with a high accuracy from the first time. On average, RCW only needs 4.67 hops per sample to obtain accurate results in our dataset.

8.1.2. Evaluating the RCW algorithm using the relative error

We compare now the RCW algorithm with a uniform probability distribution simulator[5] using the average relative error. We define the *relative error (RE)* e_i for a node i in a collection C of s samples as

$$e_i = \frac{|fRCW_i - f_i|}{f_i},$$

where $fRCW_i$ is the number of instances of i in collection C obtained by the RCW algorithm, and $f_i = p_i \cdot s$ is the expected number of instances of i with the uniform probability distribution (p_i). We use the RE, instead of the NRMSE, because the NMRSE is aggregated by the node degree, and hence it does not give information of the error distribution.

[5] Pseudorandom number generator Mersenne Twister.

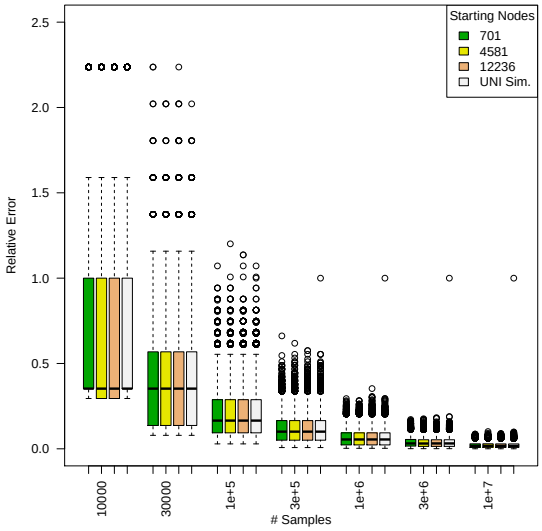

Figure 8: RCW relative error from the three different starting sources on the AS-733 graph. Three different starting nodes are used, namely 701, 4581, and 12236. The error distribution for a simulated uniform random sampling of the nodes is also shown, labelled UNI Sim.

In this experiment we chose three source nodes uniformly at random. Then, we sampled collections of different sizes, ranging from 10^3 to 10^7 samples per collection. Figure 8 shows the distribution of RE obtained for different sample sizes. Observe that for a collection of 10^6 samples (roughly 150 samples per node) the average relative error is below 5%, and that this error is similar for the three source nodes (selected at random) and similar to the one obtained by uniform sampling of the nodes.

8.2. Grids

We evaluate now the RCW algorithm using a grid of 101×101 nodes, with the source placed at the center, so we have a biggest diamond of $5,101$ nodes (recall that the RCW algorithm works in the biggest diamond contained in the grid). In this experiment, we use the relative error (e_i) to evaluate the goodness of the RCW algorithm. In Figure 9 we show the relative error for uniform sampling using several sample sizes. The mean relative error is below 3% for a 10^5 sample size (the frequency of each node is 20 on average). The average relative error in this topology is very close to the error of the previous connected topology. Again, the error distribution is very similar to the one obtained using a uniform sampling of the nodes.

8.3. Concentric Rings

In this section we compare the RCW algorithm with a UNI and PID (pseudo)random generators. To compare them we use the relative error (e_i). In these experi-

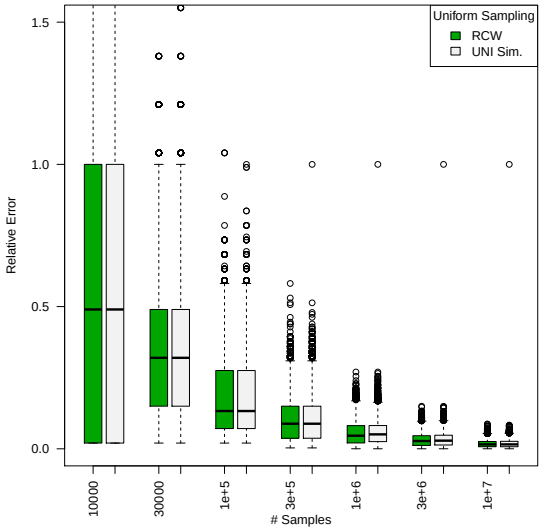

Figure 9: RCW relative error from a source at the center of a 101×101 grid. The error distribution for a simulated uniform random sampling of the nodes is also shown, labelled UNI Sim.

ments, we test two topologies, one with uniform connectivity and another one without uniform connectivity. For the non-uniform topology, we use the AAP algorithm to transform the concentric rings network without uniform connectivity into an overlay network with uniform connectivity. In order to contrast the AAP algorithm behavior we use different node density configurations.

8.3.1. Concentric Rings with uniform connectivity

We use a concentric rings topology of 100 rings with 100 nodes per ring, where the nodes are placed uniformly on each ring. This deployment guarantees the uniform connectivity property (1). We have collected different samples sets (from 10^4 to 10^7 samples). In Figure 10 we show the relative error distribution for UNI and PID sampling using two angles (15° and 30°) to find the neighbor nodes in the network. Observe that with a sample size of 10^6 (100 samples per node on average), the mean relative error is below 8% and the relative error distributions of the UNI and PID generators are very similar to the relative error of the RCW sampling.

8.3.2. Concentric Rings without uniform connectivity

We now focus on evaluating the relative error of the RCW algorithm when it is used on a more realistic topology: a concentric rings network *without uniform connectivity*. In this experiment we have used three different configurations: 100×100 (100 rings of 100 nodes), 100×1000 and 10×1000. In all of them

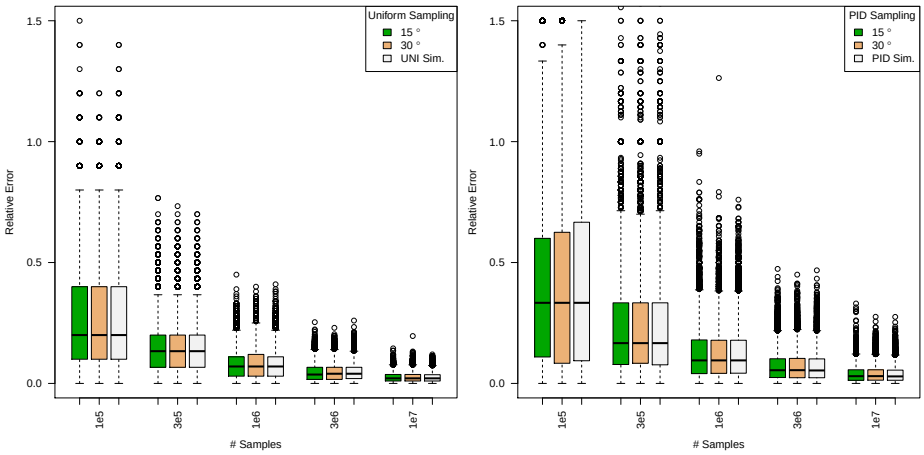

Figure 10: Relative error using different angles in a 100×100 concentric rings topology with uniform connectivity.

we have placed the ring nodes uniformly *at random* on each ring, so that the uniform connectivity is not guaranteed.

Before starting the sampling, we have executed the AAP algorithm with one attachment point per node to guarantee the uniform connectivity and hence, obtaining unbiased results. In the Figures 11 (100 rings and 100 nodes per ring), A.12 (10×1000) and A.13 (100×1000) we show the results of the RCW algorithm doing uniform sampling. In these configurations, the AAP algorithm uses two angles 150° and 180° to find neighbors in the previous and next rings for the topology of 100×100 nodes. These angles may be seem large, but the node density at the rings is low, and using smaller angles it is very likely that the AAP algorithm fails. For the topologies with a higher node density (10×1000 and 100×1000), we have used smaller angles (60° and 90°). In any case, the results in the three topologies show a very similar relative error distributions and a similar behavior to the sampling performed by the uniform simulator (UNI). In the Figures 11 (100 rings and 100 nodes per ring), A.12 (10×1000) and A.13 (100×1000) we show the behavior of the RCW algorithm using the UNI and PID sampling. As corollary, the RCW error behavior is roughly the same independently of the distribution chosen to sampling, and there is no difference between the relative error of the RCW with AAP and the sampling performed by the UNI or PID simulators.

9. Conclusions

In this paper we propose distributed algorithms for node sampling in networks. All the proposed algorithms are based on a new class of random walks called random centrifugal walks. These algorithms guarantee that the sampling completes after a number of hops that is upper bounded by the diameter of the

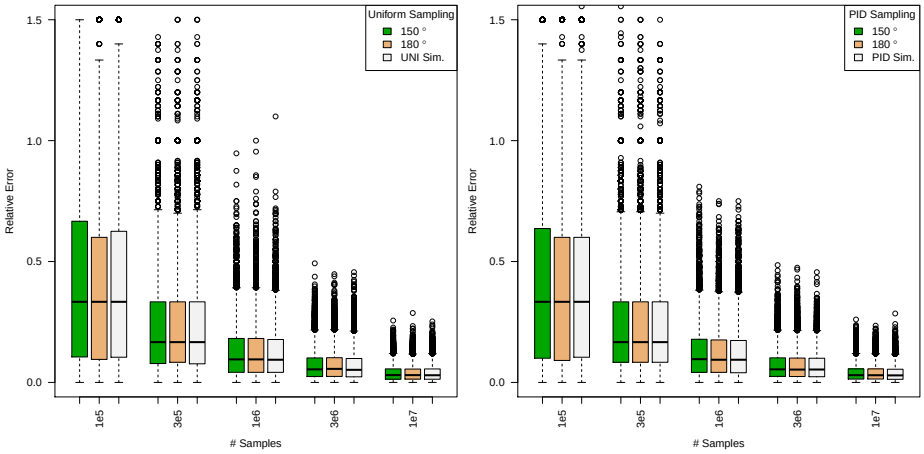

Figure 11: Relative error using different angles in a 100×100 concentric rings topology *without* uniform connectivity and using the AAP algorithm.

network, while samples with the exact probability distributions are obtained. Future work will explore sampling in dynamic networks using random centrifugal walks. Additionally, we will investigate more general algorithms that would also deal with (or employ) distributions that do not only depend on the distance from the source.

Acknowledgements

We would like to thank the anonymous referees for their useful comments. This work has been supported in part by the Regional Government of Madrid (CM) grant Cloud4BigData (S2013/ICE-2894, cofunded by FSE & FEDER), and the Spanish Ministerio de Economía y Competitividad grant TEC2014-55713-R.

References

[1] Asad Awan, Ronaldo A. Ferreira, Suresh Jagannathan, and Ananth Grama. Distributed uniform sampling in unstructured peer-to-peer networks. In *HICSS*. IEEE Computer Society, 2006.

[2] Marin Bertier, François Bonnet, Anne-Marie Kermarrec, Vincent Leroy, Sathya Peri, and Michel Raynal. D2HT: The best of both worlds, integrating RPS and DHT. In *EDCC*, pages 135–144. IEEE Computer Society, 2010.

[3] F. Bonnet, A.-M. Kermarrec, and M. Raynal. Small-world networks: From theoretical bounds to practical systems. In E. Tovar, Ph. Tsigas, and

H. Fouchal, editors, *Proc. OPODIS*, volume 4878 of *Lecture Notes in Computer Science*, pages 372–385. Springer, 2007.

[4] Marc Bui, Franck Butelle, and Christian Lavault. A distributed algorithm for constructing a minimum diameter spanning tree. *J. Parallel Distrib. Comput.*, 64(5):571–577, May 2004.

[5] Yann Busnel, Roberto Beraldi, and Roberto Baldoni. On the uniformity of peer sampling based on view shuffling. *Journal of Parallel and Distributed Computing*, 71(8):1165 – 1176, 2011.

[6] Michael Elkin. A faster distributed protocol for constructing a minimum spanning tree. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '04, pages 359–368, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[7] Pierre Fraigniaud and George Giakkoupis. On the searchability of small-world networks with arbitrary underlying structure. In Leonard J. Schulman, editor, *STOC*, pages 389–398. ACM, 2010.

[8] Beat Gfeller, Nicola Santoro, and Peter Widmayer. A distributed algorithm for finding all best swap edges of a minimum-diameter spanning tree. *IEEE Trans. Dependable Secur. Comput.*, 8(1):1–12, January 2011.

[9] Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *INFOCOM*, pages 2498–2506. IEEE, 2010.

[10] Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Practical recommendations on crawling online social networks. *Selected Areas in Communications, IEEE Journal on*, 29(9):1872 –1892, october 2011.

[11] Maxim Gurevich and Idit Keidar. Correctness of gossip-based membership under message loss. *SIAM J. Comput.*, 39(8):3830–3859, December 2010.

[12] P. Hall. On representatives of subsets. In Ira Gessel and Gian-Carlo Rota, editors, *Classic Papers in Combinatorics*, Modern Birkhäuser Classics, pages 58–62. Birkhäuser Boston, 1987.

[13] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Trans. Comput. Syst.*, 25(3), 2007.

[14] David Kempe, Jon M. Kleinberg, and Alan J. Demers. Spatial gossip and resource location protocols. *J. ACM*, 51(6):943–967, 2004.

[15] A.-M. Kermarrec, V. Leroy, and C. Thraves. Converging quickly to independent uniform random topologies. In *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, pages 159–166, Feb 2011.

[16] J. M. Kleinberg. Navigation in a small world. *Nature*, 406(6798), August 2000.

[17] Chul-Ho Lee, Xin Xu, and Do Young Eun. Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '12, pages 319–330, New York, NY, USA, 2012. ACM.

[18] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[19] D. Milić and and T. Braun. Netice9: A stable landmark-less network positioning system. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 96 –103, oct. 2010.

[20] Andrés Sevilla, Alberto Mozo, M. Lorenzo, Jose López-Presa, Pilar Manzano, and Antonio Fernández Anta. Biased selection for building small-world networks. In Chenyang Lu, Toshimitsu Masuzawa, and Mohamed Mosbah, editors, *Principles of Distributed Systems*, volume 6490 of *Lecture Notes in Computer Science*, pages 32–47. Springer Berlin / Heidelberg, 2010.

[21] W.K.Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[22] Ziqiu Yun, Xiaole Bai, Dong Xuan, Weijia Jia, and Wei Zhao. Pattern mutation in wireless sensor deployment. *IEEE/ACM Transactions on Networking (TON)*, 20(6):1964–1977, 2012.

[23] Ming Zhong and Kai Shen. Random walk based node sampling in self-organizing networks. *SIGOPS Oper. Syst. Rev.*, 40:49–55, July 2006.

Appendix A. Additional Figures

In this appendix we include additional figures presenting results obtained in the RCW simulations.

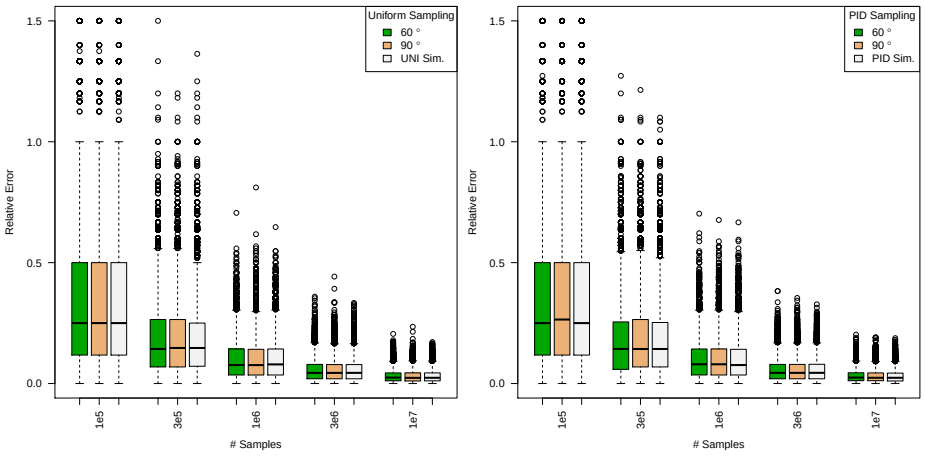

Figure A.12: Relative error using different angles in a $10 \times 1,000$ concentric rings topology *without* uniform connectivity and using the AAP algorithm.

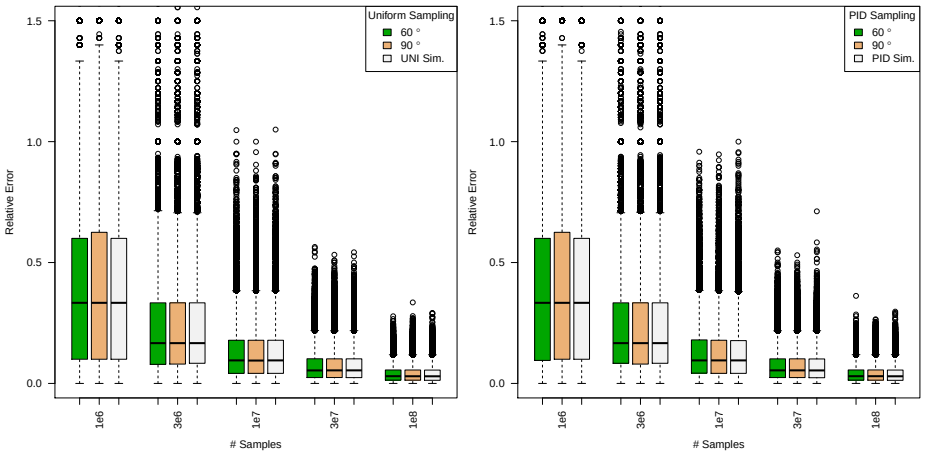

Figure A.13: Relative error using different angles in a $100 \times 1,000$ concentric rings topology *without* uniform connectivity and using the AAP algorithm.