Searching and Sampling

Take a Walk Through a Network!

Antonio Carzaniga

Mach 4, 2020

Say you have a peer-to-peer system. It may be a “structured” distributed hash table (DHT) like Chord or Kademlia, or may be an “unstructured” network like Gnutella.

Suppose that you want to search for information within the peer-to-peer system, and if you have a structured DHT suppose that you want to search outside the domain of the DHT (node id). How do you do that?

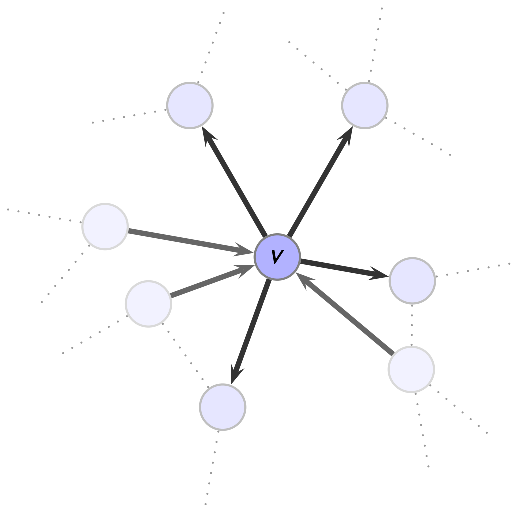
Suppose you want to *sample* the system. That is, suppose each node v has a numeric property x_v, and you want to measure the average value of x across the network. For example, you might want to measure the average number of files stored at each node.

How do you do that?

One simple way is to iterate through all the nodes. But that is most often not practical if at all possible. The problem is that the whole network is a large and decentralized system, which means that you do not have a global view of the network. Instead, you have only a *local view*.

1 A Local View of the Network

A local view means that each node v knows the nodes that v connects to, and perhaps those that connect to v. In other words, a none knows its immediate neighbors.

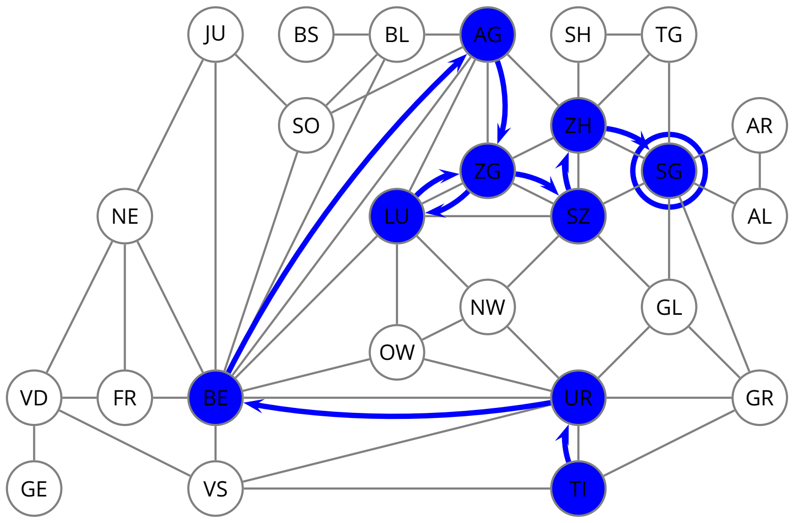


a very limited *local view* of the network

This local view is typical of peer-to-peer systems and other networks such as ad-hoc wireless networks, and can be used for several purposes and with several types of algorithms.

2 Random Walks

In particular, one way to use a local view for sampling is to use *random walks.* Random walks are also the essential components of other types of algorithms, such as “gossip” routing.



A random walk is just what it says it is: it is a path through the network induced by a very simple randomized algorithm. Notice that, as in the example above, a random walk is not necessarily a *simple* path. That is, it may go through the same node more than once.

The randominzed algorithm builds a random walk as follows: at each step, the walk visits node v, and the walk can either stop or proceed by moving to a node w adjacent to v. Typically, the walk stops after a pre-set number of hops, which defines the *length* of the walk. The choice of next hop from node v is made at random according to a given fixed distribution of probabilities (among v's neighbors) that depends *only* on the current node v, as shown in the graph below. The given and fixed probability of selecting neighbor w from node v is also called the transition probability from v to w.

Other Applications

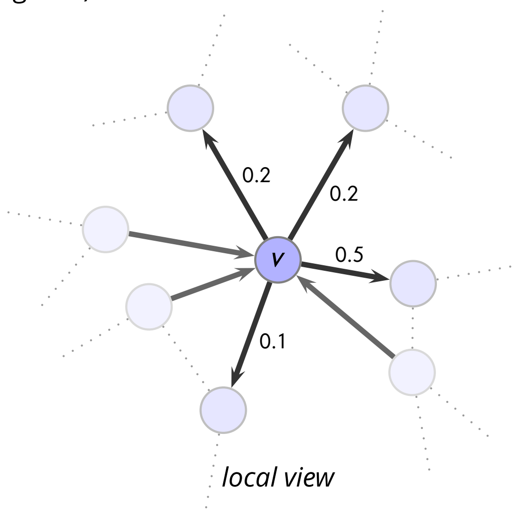
- Relevance score for hyper-linked documents (PageRank)
  - **Input:** a large collection of linked documents such as Web pages
  - **Output:** a ranking of the pages by reputation
  - a page that is linked by reputable pages acquires more reputation
  - equivalent to a random walk over the Web

Problem: given a directed graph $G = (V, A)$, compute the probability p_u that a sufficiently long random walk would end at node $u \in V$ for all nodes u.

Approaches:

1. Simulation

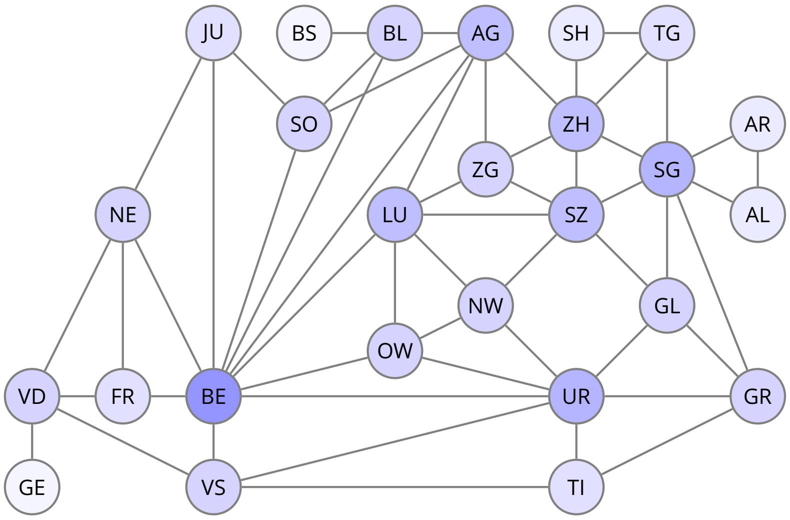
2. *Math!* (linear algebra)



local view

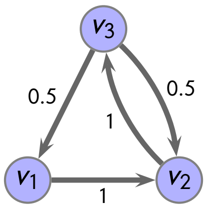
Under some very general conditions, namely if the network is *ergodic*, the visitation probabilities converge to a certain probability distribution that does not depend on the starting node. More specifically, let u_0 be the starting node and let u_k be the node reached after a random walk of k hops; then the probability $x_v(k) = \Pr[u_k = v]$ converges, with the length of the walk $k \longrightarrow \infty$, to a certain value π_v that depends on the network and the fixed transition probabilities, but does not depend on the starting node u_0.

The probability distribution π is also called the *stationary distribution*. In the graph below, the intensity of the color of each node indicates the stationary distribution when the transition probabilities are uniform. That is, from each node v, the random walk transitions to each one of v's neighbors with equal probability.



stationary distribution (*hops* $\longrightarrow \infty$)

Controlling or simply knowing the stationary distribution is very important if one wants to measure network properties by sampling the network. Also, it is important to know how long one needs to walk in order to approximate the asyptotic limit of the stationary distribution. This is where we can apply basic notions of linear algebra.



let $p_i(t) = \Pr[\text{walk is at node } v_i \text{ at time } t]$
$p(0) = [0\ 1\ 0]^{\mathsf{T}}$ means the walk starts at v_2

$$
\begin{aligned}
p_1(t+1) &= 0.5 \cdot p_3(t) \\
p_2(t+1) &= p_1(t) + 0.5 \cdot p_3(t) \\
p_3(t+1) &= p_2(t)
\end{aligned}
$$

Consider the simple network shown above, with the annotated transition probabilities, which happen to be uniform. The main idea is to represent the state of all possible random walks after t hops as a vector of probabilities $p(t)$, where the i-th element $p_i(t)$ represents the probability that a walk visits node v_i after t hops.

Then one hop in the random walk is simply a *linear transformation*—that is, a multiplication—of the vector $p(t)$ by the matrix A of the transition probabilities. For example, consider the second equation above: $p_2(t+1) = p_1(t) + 0.5 \cdot p_3(t)$ says that the probability of visiting node v_2 at hop-count $t+1$ is the probability of visiting v_1 at hop t multiplied by the probability of transitioning from v_1 to v_2, plus the probability of visiting v_3 multiplied by the probability of transitioning from v_3 to v_2.

Therefore, we can express one hop in the random walk with the simple equation in matrix form: $p(t+1) = Ap(t)$, and for a sequence of t hopes starting from the initial distribution $p(0)$, $p(t) = A^t p(0)$.

$$p(t+1) = Ap(t) \qquad p(t) = A^t p(0)$$

We can then analyze a random walk by analizing $A^t p(0)$ for increasing walk lengths t. To do that, it is very useful to look at the *eigenvalues and eigenvectors* of A, namely the vectors $x_1, x_2, \ldots, x_n$ and scalars $\lambda_1, \lambda_2, \ldots, \lambda_n$ such that $Ax_i = \lambda_i x_i$. And in general, with t repreated transformations, $A^t x_i = \lambda_i^t x_i$.

Expressing $p(0)$ as a linear combination of A's eigenvectors (for some scalar coefficients $c_1, c_2, \ldots$):

$$p(0) = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$

$$p(t) = A^t p(0) = \lambda_1^t \underbrace{c_1 x_1}_{\pi} + \underbrace{\lambda_2^t c_2 x_2 + \cdots + \lambda_n^t c_n x_n}_{\epsilon_t \approx |\lambda_2|^t \longrightarrow 0}$$

Stationary distribution

A is stochastic: $1 = |\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots$

Mixing Time: $\tau \approx \log_{|\lambda_2|} \epsilon \quad$ s.t. $\quad \epsilon_t < \epsilon$ for $t > \tau$

A is a *stochastic* matrix, meaning that its columns sum to 1, since they represent probability distributions. A very useful fact about stochastic matrices is that its largest eigenvalue λ_1 is 1, and all other eigenvalues are less than one (in modulus).

Therefore, as t grows, the component of the initial probability vector $p(0)$ along the first eigenvector, $\lambda_1^t c_1 x_1$, remains constant and equals $c_1 x_1$, while all other components $\lambda_2^t c_2 x_2, \ldots, \lambda_n^t c_n x_n$ vanish exponentially.

Therefore, the stationary distribution is simply the first component $c_1 x_1$, which is simply the first eigenvalue normalized to add up to 1.

And the other components can be seen as an error that goes to zero. Of all these components, the slowest one to go to zero is the one associated with the *second* largest eigenvalue (modulus).

Therefore, the *mixing time* of a network—the minimal number of hops in a random walk that would get the visitation probabilities close enough to the stationary distribution (up to a given error ϵ)—can be computed from the second largest eigenvalue of A.

Notice that, if the network is ergodic, then we know for sure that there is a stationary distribution π that satisfies the equation $\pi = A\pi$.

So, we can compute the stationary distribution directly by solving this system of equations:

$$A\pi = \pi$$
$$\sum_{i=1}^{n} \pi_i = 1$$

p