6.207/14.15: Networks
Lecture 7: Search on Networks: Navigation and Web Search

Daron Acemoglu and Asu Ozdaglar
MIT

September 30, 2009

Outline

Navigation (or decentralized search) in networks

Web search

- Hubs and authorities: HITS algorithm
- PageRank algorithm

Reading:

EK, Chapter 20.3-20.6 (navigation)

EK, Chapter 14 (web search)

(*Optional reading:*) "The Small-World Phenomenon: An Algorithmic Perspective," by Jon Kleinberg, in Proc. of ACM Symposium on Theory of Computing, pp./ 163-170, 2000.

Introduction

We have studied random graph models of network structure.

- Static random graph models (Erdös-Renyi model, configuration model, small-world model)
- Dynamic random graph models (preferential attachment model)

In the next two lectures, we will study processes taking place on networks. In particular, we will focus on:

- Navigation or decentralized search in networks
- Web search: ranking web pages
- Spread of epidemics

Decentralized Search

Recall Milgram's small-world experiment, where the goal was to find short chains of acquaintances (short paths) linking arbitrary pairs of people in the US.

- A source person in Nebraska is asked to deliver a letter to a target person in Massachusetts.
- This will be done through a chain where each person forwards the letter to someone he knows on a first-name basis.
- Over many trials, the average number of intermediate steps in successful chains was found to lie between 5 and 6, leading to six degrees of separation principle.

Milgram's experiment has two fundamentally surprising discoveries.

First is that such short paths exist in networks of acquaintances.

- The small-world model proposed by Watts and Strogatz (WS) was aimed at capturing two fundamental properties of networks: short paths and high clustering.

Decentralized Search

Second, people are able to find the short paths to the designated target with only local information about the network.

- If everybody knows the global network structure or if we can "flood the network" (i.e., everyone will send the letter to all their friends), we would be able to find the short paths efficiently.
- With local information, even if the social network has short paths, it is not clear that such decentralized search will be able to find them efficiently.

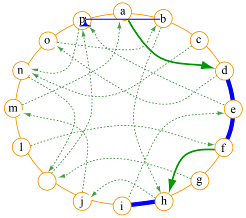


Figure: In myopic search, the current message-holder chooses the contact that is closest to the target and forwards the message to it.

A Model for Decentralized Search—1

Kleinberg introduces a simple framework that encapsulates the paradigm of WS – rich in local connections with a few long range links.

The starting point is an $n \times n$ two-dimensional grid with directed edges (instead of an undirected ring).

The nodes are identified with the lattice points, i.e., a node v is identified with the lattice point (i, j) with $i, j \in \{1, \dots, n\}$.

For any two nodes v and w, we define the distance between them $d(v, w)$ as the number of grid steps between them, $d((i,j),(k,l)) = |k - i| + |l - j|$.

Each node is connected to its 4 local neighbors directly – his local contacts.

Each node also has a random edge to another node – his long range contact.

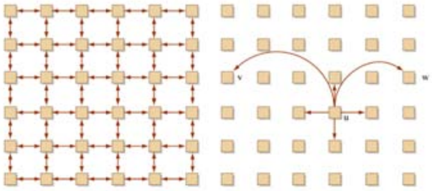


Figure: A grid network with n = 6 and the contacts of a node u.

A Model for Decentralized Search—2

The model has a parameter that controls the “scales spanned by the long-range weak ties.”

The random edge is generated in a way that decays with distance, controlled by a clustering exponent α: In generating a random edge out of v, we have this edge link to w with probability proportional to $d(v, w)^{-\alpha}$.

- When $\alpha = 0$, we have the uniform distribution over long-range contacts – the distribution used in the model of WS.
- As α increases, the long-range contact of a node becomes more clustered in its vicinity on the grid.

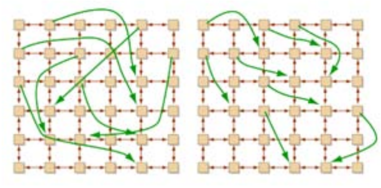


Figure: (left) Small clustering exponent, (right) large clustering exponent

Decentralized Search in this Model

We evaluate different search procedures according to their expected delivery time– the expected number of steps required to reach the target (where the expectation is over a randomly generated set of long-range contacts and randomly chosen starting and target nodes).

Given this set-up, we will prove that decentralized search in WS model will necessarily require a large number of steps to reach the target (much larger than the true length of the shortest path).

- As a model, WS network is effective in capturing clustering and existence of short paths, but not the ability of people to actually find those paths.
- The problem here is that the weak ties that make the world small are "too random" in this model.

The parameter α captures a tradeoff between how uniform the random links are.

Question: Is there an optimal α (or network structure) that allows for rapid decentralized search?

Efficiency of Decentralized Search

Theorem (Kleinberg 2000)

Assume that each node only knows his set of local contacts, the location of his long-range contact, and the location of the target (crucially, he does not know the long-range contacts of the subsequent nodes). Then:

(a) *For $0 \leq \alpha < 2$, the expected delivery time of any decentralized algorithm is at least $\beta_\alpha n^{(2-\alpha)/3}$ for some constant β_α.*

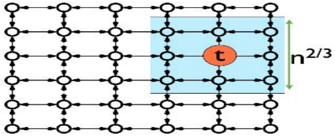
(b) *For $\alpha = 2$, there is a decentralized algorithm so that the expected delivery time is at most $\beta_2(\log(n))^2$ for some constant β_2.*

(c) *For $2 < \alpha < 3$, the expected delivery time of any decentralized algorithm is at least $\beta_\alpha n^{(\alpha-2)}$ for some constant β_α.*

Decentralized search with $\alpha = 2$ requires time that grows like a polynomial in $\log(n)$, while any other exponent requires time that grows like a polynomial in n – exponentially worse.

Proof Idea for $\alpha = 0$

The basic idea is depicted in the figure below: We randomly pick a target node in the grid and consider the (square) box around t of side length $n^{2/3}$.



With high probability, the source node lies outside the box.

Because long range contacts are created uniformly at random (since $\alpha = 0$), the probability that any one node has a long range contact inside the box is given by the size of the box divided by n^2, i.e., $\frac{n^{4/3}}{n^2} = n^{-2/3}$.

Therefore, any decentralized search algorithm will need at least $n^{2/3}$ steps in expectation to find a node with a long-range contact in the box.

Moreover, as long as it doesn't find a long range link leading into the box, it cannot reach the target in less than $n^{2/3}$ steps (using only local contacts).

This shows that the expected time for any decentralized search algorithm must take at least $n^{2/3}$ steps to reach the target node.

Proof Idea for $0 \leq \alpha < 2$

We use a similar construction for this range: we consider a (square) box around the target with side length n^{γ}, where $\gamma \equiv \frac{(2-\alpha)}{3}$.

Recall that a node v forms a long-range link with node w with probability proportional to $d(v,w)^{-\alpha}$. Here, the constant of proportionality is $1/Z$ with $Z = \sum_w d(v,w)^{-\alpha}$.

Note that in the 2-dimensional grid, a node has $\approx d$ neighbors at distance d from itself. This implies that

$$Z = \sum_{d=1}^{n} d.d^{-\alpha} \approx n^{2-\alpha} = n^{3\gamma},$$

where the last relation follows using an integral approximation.

Hence, the probability that any one node has a long range contact inside the box satisfies $\leq \frac{n^{2\gamma}}{n^{3\gamma}} = n^{-\gamma}$ (since there are $n^{2\gamma}$ nodes inside the box)

This shows that any decentralized search algorithm will need at least $n^{\gamma} = n^{\frac{(2-\alpha)}{3}}$ steps in expectation to find a node with a long-range contact in the box.

Proof Idea for $\alpha = 2$

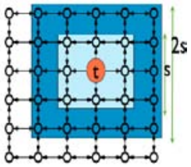
We again first compute the proportionality constant Z:

$$Z = \sum_{d=1}^{n} d \frac{1}{d^2} = \sum_{d=1}^{n} \frac{1}{d} \approx \log(n).$$

(where the last relation follows by an integral approximation).

Instead of picking an "impenetrable box" around the target, we show that it is easy to enter smaller and smaller sets centered around the target node.

We consider a node at a distance $2s$ from the target and compute the time it takes to halve this distance (i.e., get into a box of size s).



Proof Idea for $\alpha = 2$ (Continued)

The probability of having a long range contact in the box of size s is $\geq s^2 \frac{1}{\log(n)} \frac{1}{(2s)^2} \approx \frac{1}{\log(n)}$.

Hence, it takes in expectation $\log(n)$ time steps to halve the distance. Since we can halve the distance at most $\log(n)$ times, this leads to an algorithm that takes $(\log(n))^2$ steps to reach the target in expectation.

Proof Idea for $2 < \alpha < 3$

Computing the proportionality constant Z in this case yields $Z =$ constant (independent of n) for large n.

The expected length of a typical long-range contact is given by

$$\mathbb{E}[\text{length of a typical long range contact}] = \frac{1}{Z} \sum_{d=1}^{n} d.d.\frac{1}{d^{\alpha}} \approx n^{3-\alpha}.$$

Hence, the expected time is at least $n/n^{3-\alpha} = n^{\alpha-2}$.

Web Search – Link Analysis

When you go to Google and type "MIT", the first result it displays is web.mit.edu, the home page of MIT University.

- How does Google know that this was the best answer?

This is a problem of *information retrieval*: since 1960's, automated information retrieval systems were designed to search data repositories in response to keyword queries.

- Classical approach has been based on "textual analysis", i.e., look at each page separately without regard to the link structure.

In the example of MIT home page, what makes it stand out is the number of links that "point to it", which can be used to assess the authority of a page on the topic:

- First, collect a large sample of pages that are relevant to the query as determined by a classical, text-only, information retrieval approach.
- Pick the webpage that receives the greatest number of in-links (score) from these pages.

Link Structure–1

For a different query, say "newspaper", the "most important page" seems less obvious, i.e., you get a high score for a mix of prominent newspapers, as well as for pages that are going to receive many links no matter what the query is (such as Yahoo. Facebook. Amazon).

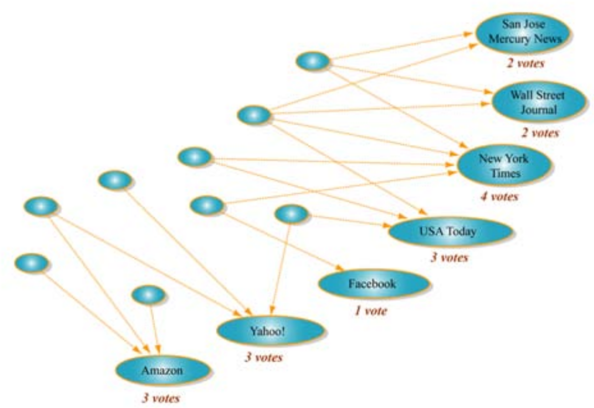


Image by MIT OpenCourseWare. Adapted from Easley, David, and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. New York, NY: Cambridge University Press, 2010. ISBN: 9780521195331.

Figure: Counting in-links to pages for the query "newspapers".

Link Structure–2

We can get more from the link structure by identifying nodes or pages that compile lists of resources relevant to the topic.

- Among the relevant pages, a few of them in fact point to *many* of the pages that received a high score.
- Concretely, we can say that a page's value as a list is equal to the sum of the scores received by all the pages that it is pointing to.
- This could then be used to update the scores of the webpages.

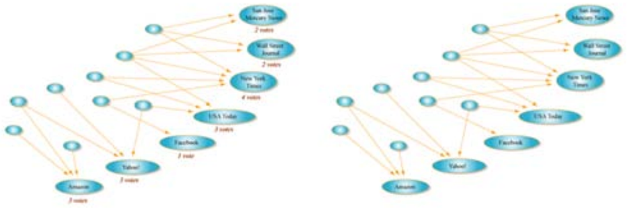


Figure: (left) Finding good lists for the query "newspapers". (right) Re-weighting votes for the query "newspapers".

Hubs and Authorities–1

This suggests a ranking procedure which is defined in terms of two kinds of nodes:

- Authorities: The prominent highly endorsed answers to the queries (nodes that are pointed to by highly ranked nodes)
- Hubs: High-value lists (nodes that point to highly ranked nodes)

For each page p, we are trying to estimate its value as a potential authority and as a potential hub, so we assign it to numerical values $b(p)$ (for authority weight) and $h(p)$ (for hub weight).

Let A denote the $n \times n$ adjacency matrix, i.e., $A_{ij} = 1$ if there is a link from node i to node j.

The authority and hub weights then satisfy:

$$h(i) = \sum_j A_{ij} b(j) \qquad \text{for all } i,$$

$$b(j) = \sum_i A_{ij} h(i) \qquad \text{for all } j.$$

Hubs and Authorities–2

This can be written in matrix-vector notation as:

$$h = Ab, \qquad b^T = h^T A \quad \text{or} \quad b = A^T h,$$

where b^T (or A^T) denotes the transpose of vector b (or matrix A).

This yields an iterative procedure in which at each time $k \geq 0$, the authority and hub weights are updated according to:

$$b_{k+1} = (A^T A) b_k,$$

$$h_{k+1} = (A A^T) h_k.$$

Using an eigenvector decomposition, one can show that this iteration converges to a limit point related to the eigenvectors of the matrices $A^T A$ and AA^T.

Implementation of this algorithm requires "global knowledge," therefore it is implemented in a "query-dependent manner".

Page Rank–1

Multi-billion query-independent idea of Google.

Each node (or page) is important if it is cited by other important pages.

Each node j has a single weight (PageRank value) $w(j)$ which is a function of the weights of his (incoming) neighbors:

$$w(j) = \sum_i \frac{w(i)}{d_{out}(i)} A_{ij},$$

where A is the adjacency matrix, $d_{out}(i)$ is the out-degree of node i (used to dilute the importance of node i if he is linked to many nodes).

We can express this in vector-matrix notation as:

$$w^T = w^T P,$$

where $P_{ij} = \frac{A_{ij}}{d_{out}(i)}$ (note that $\sum_j P_{ij} = 1$).

Page Rank–2

This defines a random walk on the nodes of the network:

- A walker chooses a starting node uniformly at random. Then, in each step, the walker follows an outgoing link selected uniformly at random from its current node, and it moves to the node that this link points to.

The PageRank of a node i is the limiting probability that a random walk on the network will end up at i as we run the walk for larger number of steps.

Difficulty with PageRank: Dangling ends which may cause the random walk to get trapped.

To fix this, we allow the random walk at each step with probability $(1 - s)$ to jump to a node chosen uniformly at random (with probability s, as before, it follows a random outgoing link).

This yields the following iteration for the PageRank algorithm: at step k,

$$w_{k+1}^T = s w_k^T P + \frac{(1-s)}{n} e^T, \qquad \text{where } e^T = [1, \ldots, 1].$$

This is the version of PageRank used in practice (with many other tricks) with s usually between 0.8 and 0.9.