# Social and Economic Networks

## Navigation in networks

Xiang Sun

2019 Fall

# Outline

1. Navigation in a random network
   - The environment without a network structure
   - Search in a network
   - Variations on navigation techniques
2. Tree-structure model and homophily
   - Navigating in a binary tree
   - Navigation with general homophily
   - Search efficiency/navigation speed
3. Social structure and decentralized search
   - One-dimensional model
   - Two-dimensional model

# Reference

- Easley and Kleinberg, Chapter 20
- Jackson, Chapter 7.3
- MIT Open Course Networks, Fall 2009, Lecture 7

# Navigation

- Problem: Find a node that has a certain attribute.
- In the classic Milgram experiment, subjects were faced with the task of getting a letter to a particular person.
- Other examples:
  - find a webpage with particular information on it;
  - find someone who knows how to perform a given task;
  - find a file-sharer that has a given file.
- Relationship with diffusion:
  - Navigation: targeted decentralized search;
  - * An algorithm for searching to find a directed path from node $s$ to node $t$ is decentralized if it only uses information about the identities of the neighbors of $s$ and their location, and the location of node $t$.
  - Diffusion: wide-ranging diffusion (flood the network).

# Milgrom's letter experiment

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 (表示双方互以名字相称，即彼此很熟).
- 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.

# Key finding

Milgram's experiment has two fundamentally surprising discoveries.

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

# Key finding (Cont.)

Milgram's experiment has two fundamentally surprising discoveries.

- First, such short paths exist in networks of acquaintances.
- 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.
  - * Example: In a large social-networking site, everyone was known only by 9-digit pseudonyms. Then it is not easy to forward a letter to user number 482285204, using only people you know on a first-name basis.

# Navigation

- One could just randomly navigate the network until one bumps into the target node.
- One could also take advantage of the structure of the network to better search.
  - going to nodes that have more neighbors might save time, if such nodes can also tell you something about their neighbors.
  - one might use information about the nodes themselves to help in cases where nodes tend to be connected to other nodes with similar attributes.
- Question to be answered:
  - How do different search methods perform? Speed? Effectively?
  - How does search depend on the network structure?

# Benchmark: Without a network structure

- There is a network of $n$ nodes.
- We need to find a single target node in the network.
- We can simply exhaustively visit the nodes one by one, picking the order uniformly at random.
- Here we do not use the network in any way.

# Benchmark: Without a network structure (Cont.)

- There is an equal chance that the desired node will be the first node we visit, or the second, ... or the last.
- The number of nodes we would have to visit under this method follows a uniform distribution, where the probability of it taking $k$ nodes is simply $\frac{1}{n}$.
- The expected number of nodes we would have to visit is

$$\frac{1}{n} + \frac{2}{n} + \cdots + \frac{k}{n} + \cdots + \frac{n}{n} = \frac{n+1}{2}.$$

- With large numbers of nodes, this could be very time consuming and inefficient—the expected speed is $O(n)$.

# With a network structure

- We use some aspects of the network structure as follows.
- We begin by randomly picking a node.
- If it is not the right node, then we randomly pick one of its neighbors, and so forth.
- We add a feature to the setting that makes search easier. When visiting a node, in addition to being able to discern whether it is the target node, we can also tell whether any of its neighbors is the target node. For instance,
  - looking for a person: we can just ask the person we are visiting whether he know the person we are looking for.
  - crawling the world-wide-web: when we visit a given page, its links could be labeled in such a way that we can tell whether any of those links point to our target page.

# With a network structure (Cont.)

- We use some aspects of the network structure as follows.
- We begin by randomly picking a node.
- If it is not the right node, then we randomly pick one of its neighbors, and so forth.
- We add a feature to the setting that makes search easier. When visiting a node, in addition to being able to discern whether it is the target node, we can also tell whether any of its neighbors are the target node.
- In this case, when none of the neighbors of the node we are currently visiting is our target node, select the next node to be visited uniformly at random from the list of neighbors that we have not yet visited.

# Illustration: Regular network without overlap

- Consider a regular network where each node has degree $d$.
- On the first step, we randomly pick a starting node and search its $d$ neighbors.
- On the second step, we randomly pick one neighbor and search its neighbors except the starting node ($d-1$ nodes), presuming that the new nodes found do not overlap with previously visited nodes.
- If overlap were never an issue, it would take us effectively $\approx \frac{n}{d-1}$ steps to visit the whole network, not counting back-tracking if we hit a dead end.

# Illustration: Regular network without overlap (Cont.)

- The probability that our target is the $k$-th node found is $\frac{1}{n}$.
- It takes approximately $\frac{k}{d-1}$ steps to find $k$ nodes (or $k$-th node) if there is no overlap.
- The expected number of steps it would take us if there were no overlap is approximated as

$$\sum_{k=1}^{n} \frac{1}{n} \frac{k}{d-1} = \frac{n+1}{2(d-1)}.$$

# Illustration: Regular network with overlap

- How overlap slows down searching?
- Suppose that for the first half of the nodes searched the rate is only $\frac{d-1}{2}$ new nodes found at each step, so that half of the nodes are ones already visited.
- Then for the next quarter of the nodes visited, the rate is $\frac{d-1}{4}$ and so forth.

# Illustration: Regular network with overlap (Cont.)

- If the node happens to be in the first half of the nodes searched, then the expected time to find the node is

$$\frac{\frac{n}{2}+1}{2^{\frac{d-1}{2}}} \approx \frac{n}{2(d-1)}.$$

- If the node happens to be in the next quarter of nodes visited, then the expected time is

$$\frac{\frac{n}{2}}{\frac{d-1}{2}} + \frac{\frac{n}{4}+1}{2^{\frac{d-1}{4}}} \approx \frac{n}{d-1} + \frac{n}{2(d-1)}.$$

- If we continue in this manner, the expected time conditional on the node being in the next eighth is

$$\frac{2n}{d-1} + \frac{n}{2(d-1)}$$

and so forth.

# Illustration: Regular network with overlap (Cont.)

- The overall expected time to finding the node is then approximately

$$\sum_{k=1}^{\infty} \frac{1}{2^k} \left[ \frac{(k-1)n}{d-1} + \frac{n}{2(d-1)} \right] = \frac{n}{2(d-1)} \sum_{k=1}^{\infty} \frac{k - \frac{1}{2}}{2^k} = \frac{3n}{2(d-1)}.$$

- This has tripled the expected time to finding the node (than the case without overlap).
- This is not a precise calculation, since it presumes that the fraction of new nodes found at a given step is roughly proportional to the proportion of unmet nodes in network, which might be an over- or under-estimate depending on the architecture of the network.
- But at least it gives us the idea that while this slows down the process, it changes it by a factor rather than by a power.

# Illustration: Network with degree distribution

- A network with degree distribution $P$.
- We randomly pick new nodes (neighbors of the starting point).
- The degree of the new node has a distribution described by $\tilde{P}(d) = \frac{dP(d)}{\mathbf{E}[d]}$ (degree distribution of neighbors).
- Then, ignoring overlap, each new node visited through this search process informs us about an expected number of additional nodes given by

$$\sum_d (d-1)\frac{dP(d)}{\mathbf{E}[d]} = \frac{\mathbf{E}[d^2]}{\mathbf{E}[d]} - 1.$$

# Illustration: Network with degree distribution (Cont.)

- Using the expression in place of the $d - 1$ from the analysis with a regular network, we have that the expected number of steps until we find our target is roughly

$$\frac{n+1}{2(\frac{\mathbf{E}[d^2]}{\mathbf{E}[d]} - 1)} \approx \frac{1}{2}\frac{n\mathbf{E}[d]}{\mathbf{E}[d^2]}.$$

- Ignoring overlap is a good approximation for many random networks below a threshold where fixed-sized loops become prevalent, but could lead to under-estimation above such thresholds. Providing fully accurate estimates for rich models of networks admitting nontrivial clustering is a difficult problem, and there is little work on that subject.

# Illustration: Poisson random network

- The expected number of steps until we find our target is roughly

$$\frac{n+1}{2(\frac{\mathbf{E}[d^2]}{\mathbf{E}[d]}-1)} \approx \frac{1}{2}\frac{n\mathbf{E}[d]}{\mathbf{E}[d^2]}.$$

- For a Poisson random network, it is

$$\frac{n\mathbf{E}[d]}{\mathbf{E}[d^2]} = \frac{n}{\mathbf{E}[d]} = \frac{n}{(n-1)p} = O(n).$$

- Thus, a searching through a Poisson network is quite similar to searching through a regular network.

# Illustration: Scale-free network

- Consider a network that has a degree distribution approximated by a power distribution, so that $P(d) = cd^{-\gamma}$ for some scalar $c$ and $\gamma < 3$, but such that the nodes' degrees are independent. (for example, generated by the configuration model)
- Assume that the maximal degree in the distribution is $M < n$ (truncation).
- $\mathbf{E}[d^2] = \sum_{d=1}^{M} cd^2 d^{-\gamma} \approx \int_1^M cd^{2-\gamma}\,\mathrm{d}d = \frac{c}{3-\gamma}(M^{3-\gamma} - 1)$, where the second relation follows using an integral approximation.
- Similarly, $\mathbf{E}[d] = \frac{c}{2-\gamma}(M^{2-\gamma} - 1)$.

# Illustration: Scale-free network (Cont.)

- Thus, the expected time to finding the desired node in a power distribution truncated at a maximum degree of $M$ is proportional to

$$\frac{n\mathbf{E}[d]}{\mathbf{E}[d^2]} \approx n\frac{(3-\gamma)M^{2-\gamma}-1}{(2-\gamma)M^{3-\gamma}-1}.$$

- For large $M$ and $2 < \gamma < 3$, this is proportional to

$$\frac{n}{M^{3-\gamma}}.$$

# Illustration: Scale-free network (Cont.)

- Since a change in the truncation or maximum degree $M$, can lead to dramatic changes in the calculation of $\mathbf{E}[d^2]$ and other moments, we find that it can lead to a significant change in the expected time to finding the desired node.
- There is no right or wrong way to do things here, as each truncation leads to a valid degree distribution that approaches a continuous power-distribution as the number of nodes expands.
- But many finite distributions that approach continuous power-distributions in the limit have different features.

# Illustration: Scale-free network (Cont.)

- Cohen et al. suggest the setting the maximum on a discrete finite approximate power-law distribution to be $M = n^{\frac{1}{\gamma-1}}$.
- It leads the expected number of steps to finding the desired node to be proportional to

$$\frac{n}{n^{\frac{3-\gamma}{\gamma-1}}} = n^{\frac{2(\gamma-2)}{\gamma-1}}.$$

- If $\gamma = 2.5$, then the expected time is $n^{\frac{2}{3}}$, which is much more efficient than the linear-in-$n$-time that we saw for the Poisson and regular networks.

# Summary

- For the regular and Poisson networks, the navigation speed is $O(n)$.
- For the power distribution, the navigation speed is $O(\text{lower power of } n)$.
- Networks that have larger tails in distribution lead to much more effective search.
- The degree distribution places more weight on higher degree nodes.

$\Rightarrow$ We are more likely to find larger degree nodes through following randomly chosen links.

$\Rightarrow$ We end up discovering more of the network by searching fewer nodes.

# Information of second neighborhoods

- Consider a situation where instead of just getting information about direct neighbors, a node can also report on second neighbors.
- At each step we learn about a new number of nodes which is proportional to the size of the second neighborhood of a node found by following a random link.
- Without any overlap, and with independence in neighboring nodes degrees, the size of the second neighborhood of a node found through such search is simply

$$\underbrace{\frac{\mathbf{E}[d^2]}{\mathbf{E}[d]} - 1}_{\text{direct neighbors}} + \underbrace{\left[\frac{\mathbf{E}[d^2]}{\mathbf{E}[d]} - 1\right]^2}_{\text{expected number of second neighbors}} = \frac{\mathbf{E}[d^2]}{\mathbf{E}[d]}\left[\frac{\mathbf{E}[d^2]}{\mathbf{E}[d]} - 1\right].$$

# Information of second neighborhoods (Cont.)

- This roughly squares the number of nodes found at each step.
- For the regular and Poisson networks, the expected times become proportional to $\frac{n}{\mathbf{E}[d]^2}$.
- For the power distribution, it becomes proportional to

$$\frac{n}{n^{2\frac{3-\gamma}{\gamma-1}}} = n^{\frac{3\gamma-7}{\gamma-1}}.$$

# Taking advantage of high-degree nodes

- When a given node is not the desired node, and neither are its neighbors, then instead of picking an unvisited neighbor uniformly at random to move to next, one chooses the unvisited neighbor with the highest degree.
- As higher degree nodes have more neighbors, this not only leads to observing more nodes on a given step, but also then leads to improved opportunities (through more draws) of finding even higher degree nodes.

# Taking advantage of high-degree nodes (Cont.)

- This process quickly results in searches in which most of the nodes being searched are at the high end of the distribution.
- * For instance, in the power network, after a few steps most of the nodes examined have degree near $M$, and so a rough approximation for the expected time of search is then $\frac{n}{M}$, which for $M = n^{\frac{1}{\gamma-1}}$ becomes $n^{\frac{\gamma-2}{\gamma-1}}$.
- This method is significantly quicker than simply following links chosen uniformly at random.

# Taking advantage of high-degree nodes (Cont.)

The size of improvement depends on $\gamma$:

- If $\gamma$ is close to 2, then higher-degree nodes have more weight in the distribution.

$\Rightarrow$ One naturally finds the very largest degree nodes simply by following random links.

- If $\gamma$ is close to 3, then higher-degree nodes are a bit rarer.

$\Rightarrow$ There is more of an improvement from following a degree-based search algorithm.

# Milgram's experiment

- Navigating a network should take a time that is proportional to $n$ in a regular or Poisson network, and some lower power of $n$ if the network's degree distribution follows a power law.
- This seems inconsistent with the Milgram small-world experiments.
  - In the experiment, people were able to get a letter to a target in a median number of 5 steps, of those that were successful.
  - This was in a population on the order of hundreds of millions of people, so that even the square root of $n$ is on the order of 10,000.
  - Sending the letter to very highly connected individuals is not enough to hit the median number of 5.

# Milgram's experiment (Cont.)

- It must be that the individuals in Milgram's experiment were taking advantage of additional structure of the network in order to choose to whom to forward the letter.
- The previous search algorithms that were based entirely on network primitives without reference to any other characteristics of the nodes.

# Homophily

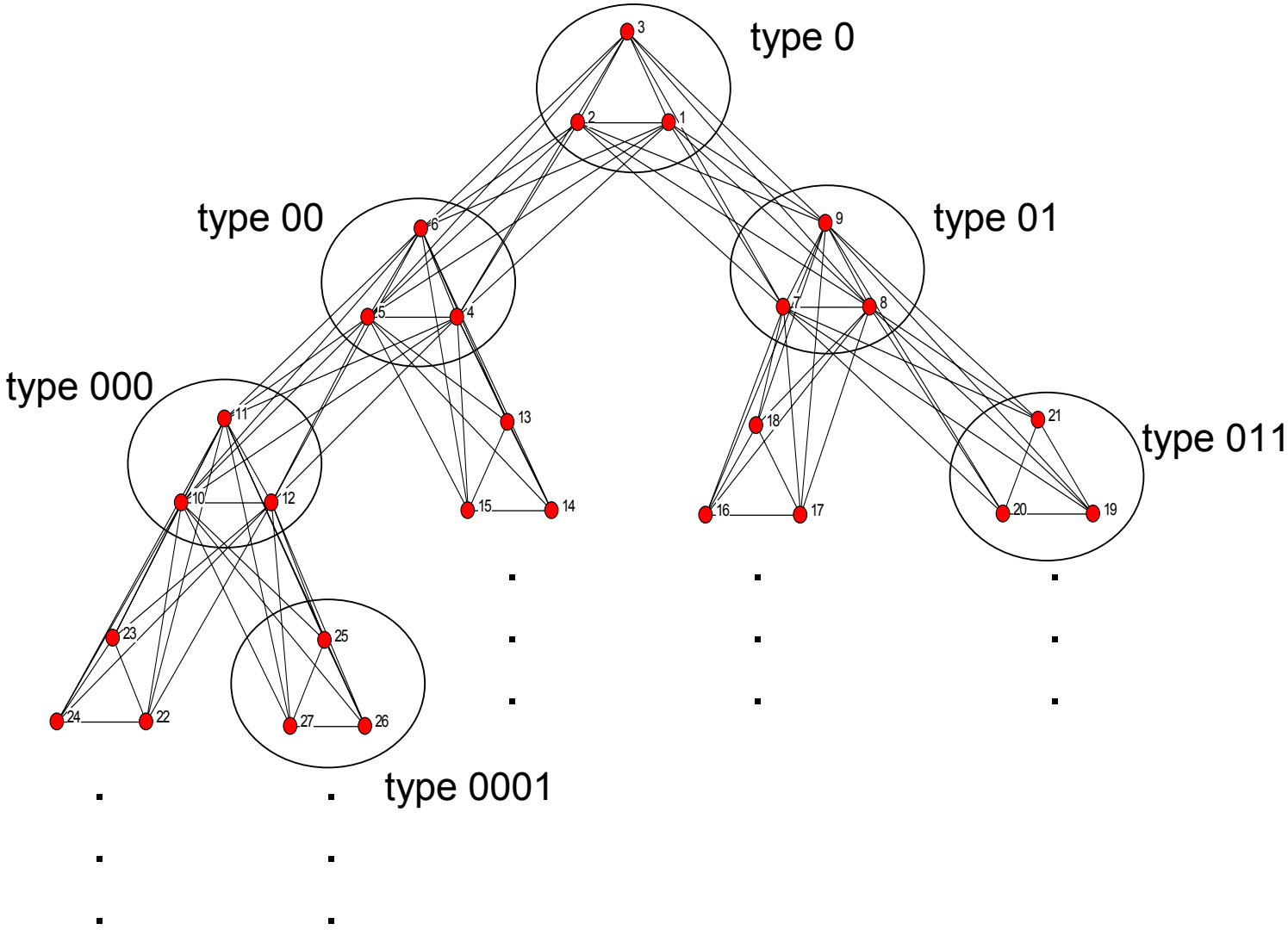
- Individuals would not just randomly choose a neighbor to send the letter to, but would instead try to send the letter to someone who has something in common with the target, or else to someone who they think might be closer to someone who has something in common with the target.
- Evidence shows that people in small-world letter experiments are primarily guided by occupation and/or geography in their choices of whom to forward the letter to.
- In situations where the formation of links is actually governed by some sort of underlying social structure, there can be much more efficient methods of navigation that use such social information.

# Binary tree

Consider a society of individuals who form a network that is described by a hierarchy in the form of a binary tree as follows.

- Each individual has a type/label.
- The first group are of type 0. This group forms the root of the tree.
- Next there are two groups, of types 00 and 01, which form the second level of the tree.
- Next there are four groups, of types 000, 001, 010 and 011, which form the third level, etc.

# Binary tree (Cont.)



# Binary tree (Cont.)

- A given individual is linked to all other individuals who are of the same type, as well as all of those who are of a type that that differs from the individual's own type by the addition or deletion of one terminal digit.
  - * So, someone of a type 0101 is connected to those with labels 010, 0101, 01010 and 01011.
- They are connected to the individuals of the same type, as well as the type immediately preceding them in the tree, and the two types that follow them in the tree (unless they are at the last level of the tree).

# Binary tree (Cont.)

- We can think of types as specifying individuals by a list of attributes, which we can think of as including all sorts of information such as their ethnicity, gender, profession, education, physical attributes, hobbies, geographic location, favorite music, etc.; which we code as vectors of 0 and 1's.
- The "tree" has $K$ levels, so the vectors of types have length at most $K$.
- If there are $m$ individuals of each type, then the society consists of

$$n = m \sum_{k=0}^{K} 2^k = m(2^{K+1} - 1)$$

individuals in total.

# Binary tree (Cont.)

- It takes at most $2(K-1)$ links to get from one individual to another.
- Since $n = m(2^{K+1} - 1)$, we have

$$K = \frac{\log(n+m) - \log m}{\log 2} - 1.$$

- Therefore, the maximum distance of $2(K-1)$ is

$$2\frac{\log(n+m) - \log m}{\log 2} - 3.$$

- Thus, the maximum distance is growing proportionally to $\log n$ for a fixed $m$.

# Binary tree (Cont.)

- If the society has hundreds of millions of people, then the maximum distance will be on the order of 10, and the median distance even less.
- This is much more in line with data from the Milgram experiments, in which observed distances had a median of 5 and maximum of 12 steps.

# Greedy algorithm

- We pick an individual at random and give him/her a letter and then ask him or her to get the letter to some other target agent in the society with a known type.
- For an individual with a type $\ell$ of length $k$:
  - If the target is a neighbor then send it directly.
  - If the target is not a neighbor, but has a type equal to $\ell$ plus some additional entries (so lies further "down" the tree), then send it to any acquaintance whose type has a $(k+1)$-st entry that matches that of the target.
  - If the target is not a neighbor and has a type that does not match $\ell$ in the first $k$ entries, then send it to a neighbor who is "higher" in the tree.

# Greedy algorithm: Remark

- Implementing the algorithm only requires an individual to have an idea of which neighbor lies closer to the target, rather than having a full appreciation of the network structure.
- The algorithm make use of the types of the individuals and the underlying social structure that indicates where different individuals appear in the tree.

# Homophily

- Let individuals be described by vectors of 0's and 1's, but such that each type is exactly $K$ entries long.
- * This is equivalent to only considering the individuals whose types lie at the leaves of the tree.
- The social distance $x_{ij}$ between two individuals $i$ and $j$ is defined as follows.
  - If two individuals are of the same type, let their distance be 1.
  - Two individuals who differ only in their last entry are at a distance of 2.
  - Individuals whose first point of difference is their second to last entry or later are at a distance of 4.
  - Individuals whose first point of difference is the third to last entry are at a distance of 6, and so forth.

# Homophily (Cont.)

- The social distance keeps track of how many links one would have to travel in the tree to get from one type at the bottom row to some other type at a bottom row.
- These distances are just "social" distances, which are some measures of similarity, but do not yet correspond to actual distances in the network.

# Random network

- The random network is then formed as follows.
- Uniformly at random pick a node $i$.
- Next pick a distance of $x \in \{1, 2, 4, 6, \ldots, 2(K-1)\}$, where $K$ is the depth of the social tree, with probabilities $ce^{-\alpha x}$, where $c$ normalizes the probabilities to sum to one and $\alpha$ is a parameter that adjusts how sensitive the link formation process is to similarity.
- Once $x$ is chosen, then uniformly at random select a node $j$ at that distance $x$ from $i$ and connect those nodes (provided there is not already a connection).
- Repeat this process until some average number of links per node, $d$, has been reached.

# Random network (Cont.)

- When $\alpha$ is high, then nodes will form most of their links to other nodes that are more similar to themselves.
- When $\alpha$ is low, then the links are formed more uniformly at random.

# Greedy algorithm

- Since this is a random network, it is possible that there will not exist a path between two nodes.
- Nevertheless, individuals can still follow a greedy algorithm of forwarding the letter to the neighbor who has a minimal social distance $x_{ij}$ to the target, although that might no longer be a fully optimal algorithm given that the network structure is now randomly determined.

# Simulation

- Watts, Dodds and Newman construct such random networks through simulation, and then examine the results of following the above described algorithm for randomly selected pairs of nodes.
- They set population size $n = 10^8$, average degree $d = 300$, the "homophily parameter" $\alpha = 1$, work with a tree with ten branches at each level, and 100 individuals in each group at the leaf of a hierarchy.
- They then posit a probability of .25 that a message is lost during any step, so that it is possible that some messages never reach their targets.
- Based on this, they measure the average distance of messages that eventually reach their targets and find it to be about 6.7, which is quite close to the 6.5 from the Milgram experiments.

# Efficiency of decentralized search

- Kleinberg proves that there exist parameter values for which as $n$ grows, two nodes picked uniformly at random will be connected at a distance of at most $O(\log n)$ with a probability of at least $1 - \varepsilon(n)$ for some function $\varepsilon(n) \to 0$.
- Kleinberg also shows that it is critical for to be exactly 1 in order for such a result to hold.
  - If becomes too small, then the network begins to resemble a uniformly random network, which has a longer navigation time.
  - If is too large, then the network connections are only formed to nearby nodes.

# Network with homophily

- Consider a set of $n$ nodes.
- The primitive distances between nodes are described by a hierarchical structure of a tree. The tree $T$ has $b \geq 2$ branches at each level, and has $n$ leaves.

$\star$ $n = b^K$ where $K = \log_b n$ is the depth of the tree $T$.

- The distance between two nodes $i$ and $j$, denoted by $x_{ij}$, is half of the distance in the tree between two nodes $i$ and $j$.

$\Rightarrow$ $x_{ij}$ corresponds to the depth of the smallest subtree that contains both $i$ and $j$.

# Network with homophily (Cont.)

- $x_{ij}$ is not the distance in the random network that will be formed based on the tree, but just an auxiliary distance which might be thought of as some primitive measure of how dissimilar two nodes are.
- For each node $i$, form $d$ directed links, where the node at the other end of a given link is formed is chosen independently at random where the probability of choosing node $j$ is proportional to $b^{-\alpha x_{ij}}$.

# Efficiency of decentralized search

**Theorem (Kleinberg 2000)**

- If $\alpha = 1$ and $d \geq c(\log_b(n))^2$ for some $c > 0$, then there exists a decentralized algorithm for which search time is polylogarithmic (with exponent 1).
- If $\alpha \neq 1$, then there is no polylogarithmic degree for which there exists a decentralized algorithm with a search time that is polylogarithmic.

Let us say that the search time is polylogarithmic if there exists for which a starting node and target node picked uniformly at random are connected by a directed path of length at most $O((\log n)^\gamma)$ with a probability of at least $1 - \varepsilon(n)$ for some function $\varepsilon(n) \to 0$.

# Normalizing constant

- In the construction of the random network, the neighbors at different distances are assigned with different probabilities.
- The normalizing constant of the distribution of distances over nodes linked to by node $i$ is

$$Z = \sum_{j \neq i} b^{-x_{ij}} = \sum_{k=1}^{\log_b n} (b-1) b^{k-1} \cdot b^{-k} = \frac{b-1}{b} \log_b n \leq \log_b n.$$

# Subtree $T'$ and $T''$

- Uniformly at random select a starting node $i$ and a target node $j$.
- Let them be a "social" (not network) distance $x_{ij}$ apart, which is the depth of the smallest subtree $T'$ of the tree $T$ that contains them both.
- Consider the subtree $T''$ of depth $(x_{ij} - 1)$ that contains $j$.
- Since $i$ is at a distance of $x_{ij}$ from each leaf in $T''$ and there are $b^{x_{ij}-1}$ leaves in $T''$, the probability that $i$ is not directedly linked to any leaf in $T''$ is

$$\left(1 - b^{x_{ij}-1}\frac{b^{-x_{ij}}}{Z}\right)^d \leq \left(1 - \frac{1}{\log_b n}\right)^{c(\log_b n)^2} \to e^{-c\log_b n} = n^{-c/\log b}.$$

* The probability that $i$ fails to have a directed link to some node in $T''$ is at most $n^{-c/\log b}$.

# Subtree $T'$ and $T''$ (Cont.)

- The probability that $i$ fails to have a directed link to some node in $T''$ is at most $n^{-c/\log b}$.
- If there is a directed link to some node in $T''$, take one to a node (denoted by $i'$) in the smallest subtree possible that contains $j$.
- This new tree has depth no more than $x_{ij} - 1$.

# Iteration

- Repeat the argument starting from $i'$.
- We will establish the same upper bound on the probability of failure to find a new directed link to a further subtree.
- Given the maximal depth of the tree $T$, it takes at most $x_{ij} \leq \log_b n$ steps in this manner to reach the target $j$ from the starting node $i$ and thus the search time is polyalgorithmic with exponent 1.

# Total time

- The probability of a failure at any step is at most $n^{-c/\log b}$.
- So the overall probability of failing to find a directed path is at most $\log_b n \cdot n^{-c/\log b}$, which converges to 0 as $n$ grows.

# Interpretation

- The size of the subtree is balanced by the probability that a link goes to that subtree.
  - There are few nodes close-by in terms of social distance, but they have a proportionally higher probability of being linked to;
  - There are more nodes that are further away in terms of social distance and they have a proportionally lower probability of being linked to.
- When $\alpha = 1$ the balance is just right so that we end up with a sort of uniformity in the distribution over the "social distances" that different links span.

# $\alpha < 1$

- The normalizing constant $Z$ is such that for large $n$

$$Z = \sum_{j \neq i} b^{-\alpha x_{ij}} = \sum_{k=1}^{\log_b n} (b-1) b^{k-1} b^{-\alpha k} \geq \frac{n^{1-\alpha}}{b}.$$

- Consider a sub-tree $T'$ containing a target node $j$ and having between $n^\gamma$ and $bn^\gamma$ leaves, where $0 < \gamma < 1 - \alpha$.
- For any node $i'$ not in $T'$, the probability that $i'$ has any link into $T'$ is

$$1 - \left(1 - \frac{1}{Z} bn^\gamma\right)^d \leq d \frac{bn^\gamma}{n^{1-\alpha}/b} = dbn^{\gamma+\alpha-1}.$$

# $\alpha < 1$ (Cont.)

- For any polylogarithmic $d$, the expression $n^{\gamma+\alpha-1}$ dominates the expression.

$\Rightarrow$ With a high probability, it will still take more than a polylogarithmic number of draws of nodes before finding any one with a link into $T'$.

- In any decentralized algorithm, with high probability it will take more than a polylogarithmic number of steps from a starting node $i$ outside of $T'$ before any link into $T'$ is found.

# $\alpha > 1$

- $Z$ is larger than some constant $Z_0$.
- Consider a node $i$ such that the distance of $i$ to the target $j$ is $\log_b n$, so that the smallest subtree containing $i$ and $j$ is $T$.

  Such a starting node and target will be selected with a nonvanishing probability (in fact of just more than $\frac{b-1}{b}$).
- Let $T'$ be the tree of depth $\log_b n - 1$ that contains $j$.
- Each of $i$'s directed out-links go to any given node in $T'$ with a probability of no more than $b^{-\alpha \log_b n}/Z_0 = n^{-\alpha}/Z_0$.
- Thus, the probability that any of $i$'s directed links goes to a node in $T'$ is at most $dn\frac{n^{-\alpha}}{Z_0}$.
- In any decentralized algorithm, with high probability it will take more than a polylogarithmic number of steps from a starting node $i$ outside of $T'$ before any link into $T'$ is found.

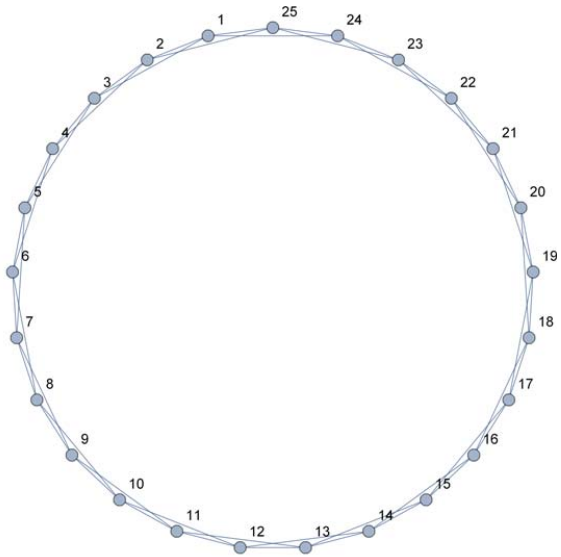
# Interpretation

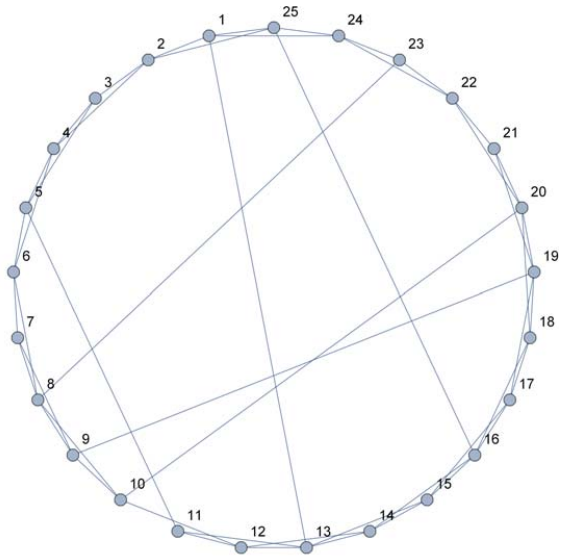
- When $\alpha$ differs from 1, then the proportionality is upset and one ends up with
  - ($\alpha < 1$) either a limiting probability that almost all links span socially dissimilar nodes, which makes it difficult to eventually approach a node;
  - ($\alpha > 1$) or else almost all links span socially similar nodes, which makes it difficult to reach between distant nodes.

# Watt-Strogatz model

- Watt-Strogatz model:
  - lattice structure,
  - randomly rewiring (adding) links.

$\Rightarrow$ Short average path lengths and high clustering.



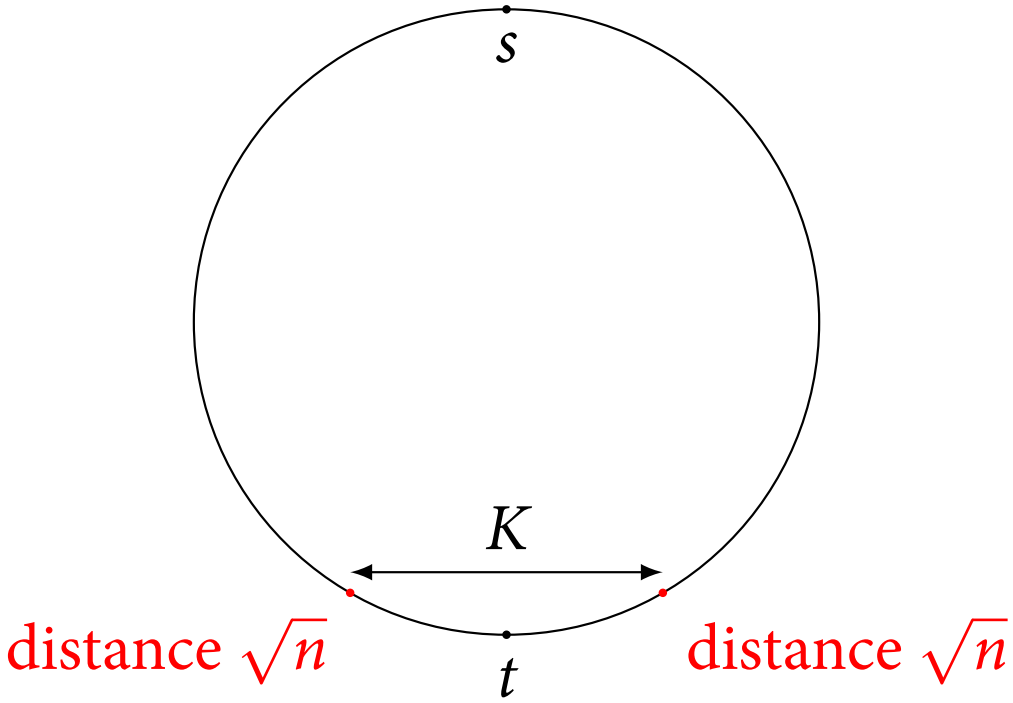



# Decentralized search in Watt-Strogatz model

- There are $n$ nodes.
- We suppose that a starting node $s$ is given a message that it must forward to a target node $t$, passing it along edges of the network (generated by Watt-Strogatz model).
- Initially $s$ only knows the location of $t$, but it does not know the random edges out of any node other than itself.
- Each intermediate node along the path has this partial information as well, and it must choose which of its neighbors to send the message to next.
- These choices amount to a collective procedure for finding a path from $s$ to $t$.

# Fail to search effectively

- Let $K$ be the set of all nodes within distance less than $\sqrt{n}$ of the target $t$.
- With high probability, the starting point $s$ of the search lies outside $K$.



# Fail to search effectively (Cont.)

- Because long-range contacts are created uniformly at random, the probability that any one node has a long-range contact inside $K$ is equal to the size of $\frac{|K|}{n} \leq \frac{2\sqrt{n}}{n} = \frac{2}{\sqrt{n}}$.
- Therefore, any decentralized search strategy will need at least $\frac{\sqrt{n}}{2}$ steps in expectation to find a node with a long-range contact in $K$.
- On the other hand, as long as it doesn't find a long-range link leading into $K$, it can't reach the target in less than $\sqrt{n}$ steps, since it would take this long to "walk" step-by-step through $K$ using only the connections among local contacts.

# Fail to search effectively (Cont.)

- From those, one can show that the expected time for any decentralized search strategy to reach $t$ must be at least proportional to $\sqrt{n}$.
- The decentralized search in the Watts-Strogatz model will necessarily require a large number of steps to reach a target—much larger than the true length of the shortest path.
- Key: The long contacts that make the world small are "too random" in this model.
- The long contacts are completely unrelated to the similarity among nodes that produces the homophily-based links, so they're hard for people to use reliably.

# Variant of WS model

- We introduce one extra quantity that controls the "scales" spanned by the long-range links.
- We have nodes on a ring, and each node still has edges to each other node within 1 steps.
- Each node has 1 random edge that is generated in a way that decays with distance, controlled by a clustering exponent $\alpha$ as follows.
  - For two nodes $v$ and $w$, let $d(v, w)$ denote the number of steps between them. (This is their distance if one had to walk along adjacent nodes on the ring.)
  - In generating a random edge out of $v$, we have this edge link to $w$ with probability proportional to $d(v, w)^{-\alpha}$.

# Clustering exponent

- The original model corresponds to $\alpha = 0$, since then the links are chosen uniformly at random.
- When $\alpha$ is very small, the long-range links are "too random," and can't be used effectively for decentralized search (as we saw specifically for the case $\alpha = 0$ above);
- When $\alpha$ is large, the long-range links are "not random enough," since they simply don't provide enough of the long-distance jumps that are needed to create a small world.

# Efficiency of decentralized search

**Theorem (Kleinberg 2000)**

1. For $\alpha = 1$, there is a decentralized algorithm so that the expected delivery time is at most $\beta_1 (\log n)^2$ for some constant $\beta_1$.
2. For $\alpha \in [0, 1)$, the expected delivery time of any decentralized algorithm is at least $\beta_\alpha n^{\frac{1-\alpha}{2}}$ for some constant $\beta_\alpha$.
3. For $\alpha \in (1, 2)$, the expected delivery time of any decentralized algorithm is at least $\beta_\alpha n^{\alpha-1}$ for some constant $\beta_\alpha$.

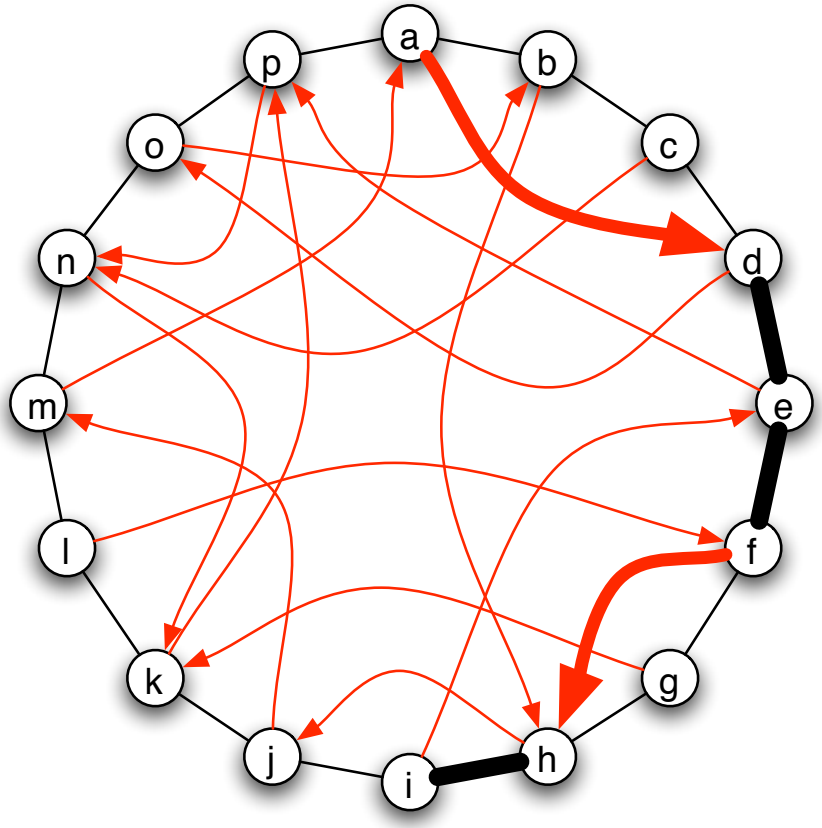
# Interpretation

- One can show that for every exponent $\alpha \neq 1$, there is a constant $\beta_\alpha$ (depending on $\alpha$), so that it takes at least proportional to $n^c$ steps in expectation for any decentralized search strategy to reach the target in a network generated with exponent $\alpha$.
- In the limit, as $n$ becomes large, decentralized search with exponent $\alpha = 1$ requires time that grows like a polynomial in $\log_2 n$, while decentralized search at any other exponent requires a time that grows like a polynomial in $n$—exponentially worse.
- The exponent $\alpha = 1$ on the ring is optimally balanced between producing networks that are "too random" for search, and those that are not random enough.

# Myopic search

- Myopic search: When a node $v$ is holding the message, it passes it to the contact that lies as close to $t$ on the ring as possible.
- Myopic search can clearly be performed even by nodes that know nothing about the network other than the locations of their friends and the location of $t$, and it is a reasonable approximation to the strategies used by most people in Milgram-style experiments.
- Result: Myopic search finds paths that are surprisingly short.

# Myopic search: Illustration



# Myopic search: Illustration (Cont.)

- The myopic path that would be constructed if we chose $a$ as the start node and $i$ as the target node in the network.
  - Node $a$ first sends the message to node $d$, since among $a$'s contacts $p$, $b$, and $d$, node $d$ lies closest to $i$ on the ring.
  - Then $d$ passes the message to its local contact $e$, and $e$ likewise passes the message to its local contact $f$, since the long-range contacts of both $d$ and $e$ lead away from $i$ on the ring, not closer to it.
  - Node $f$ has a long-range contact $h$ that proves useful, so it passes it to $h$. Node $h$ actually has the target as a local contact, so it hands it directly to $i$, completing the path in five steps.
- Notice that this myopic path is not the shortest path from $a$ to $i$. If $a$ had known that its friend $b$ in fact had $h$ as a contact, it could have handed the message to $b$, thereby taking the first step in the three-step $a - b - h - i$ path.

# Phase $j$

- The number of steps required by myopic search is a random variable $X$, and we want to show that $\mathbf{E}[X]$ is relatively small.
- As the message moves from $s$ to $t$, we'll say that it's in phase $j$ of the search if its distance from the target is between $2^j$ and $2^{j+1}$.
- The number of different phases is at most $\log_2 n$.
- We can write $X$ as the total time taken by the search is simply the sum of the times taken in each phase

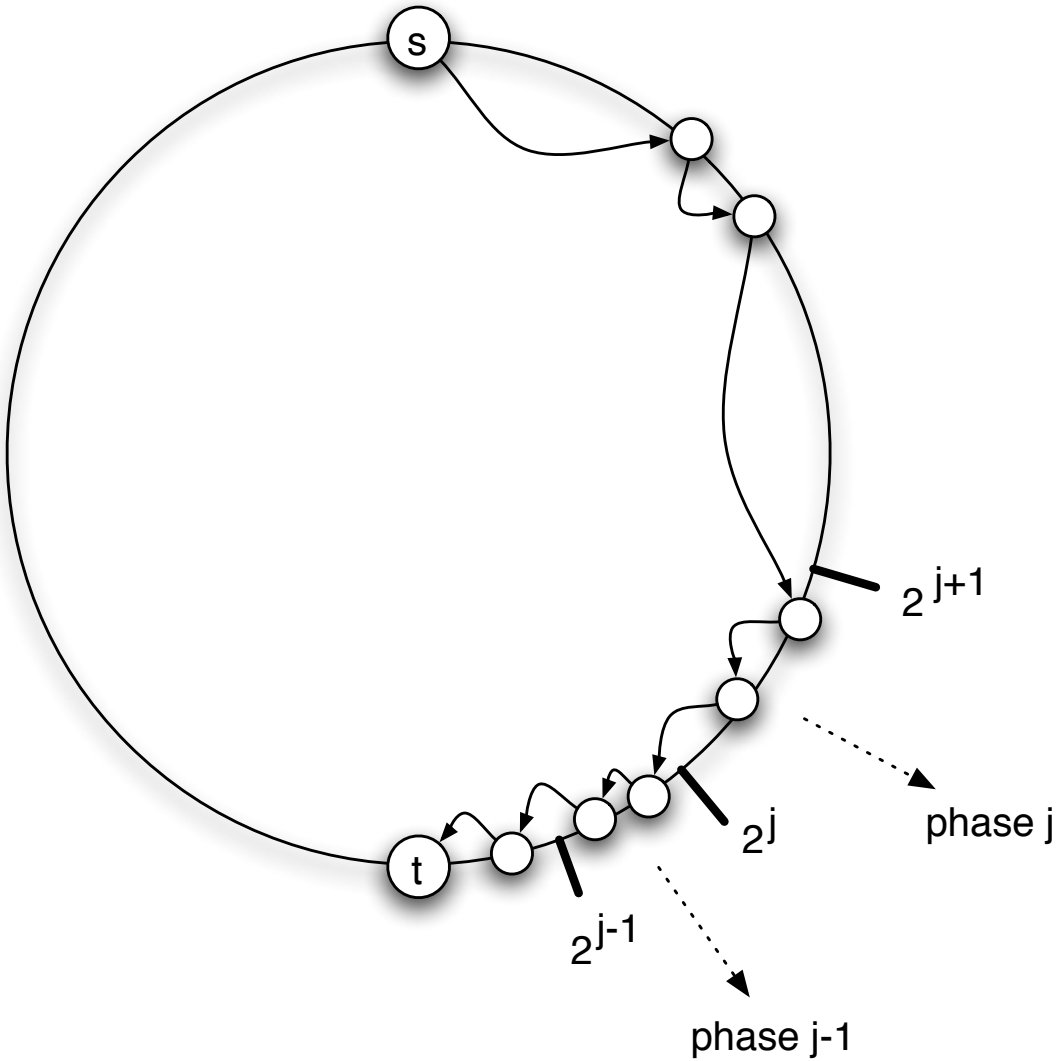
$$X = X_1 + X_2 + \cdots + X_{\log_2 n}.$$

- Thus,

$$\mathbf{E}[X] = \mathbf{E}[X_1] + \mathbf{E}[X_2] + \cdots + \mathbf{E}[X_{\log_2 n}].$$

- We want to show that $\mathbf{E}[X_j]$ is at most proportional to $\log_2 n$.

# Phase $j$ (Cont.)



# The normalizing constant

- We've been saying all along that $v$ forms its long-range link to $w$ with probability proportional to $d(v,w)^{-\alpha}$, but what is the constant of proportionality (denoted by $Z$)?
- There are two nodes at distance 1 from $v$, two at distance 2, and more generally two at each distance $d$ up to $\frac{n}{2}$.
- Therefore,

$$Z \leq 2\left(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n/2}\right).$$

- Since $1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{k} \leq 1+\int_1^k \frac{1}{x}\,\mathrm{d}x = 1+\ln k$, we have

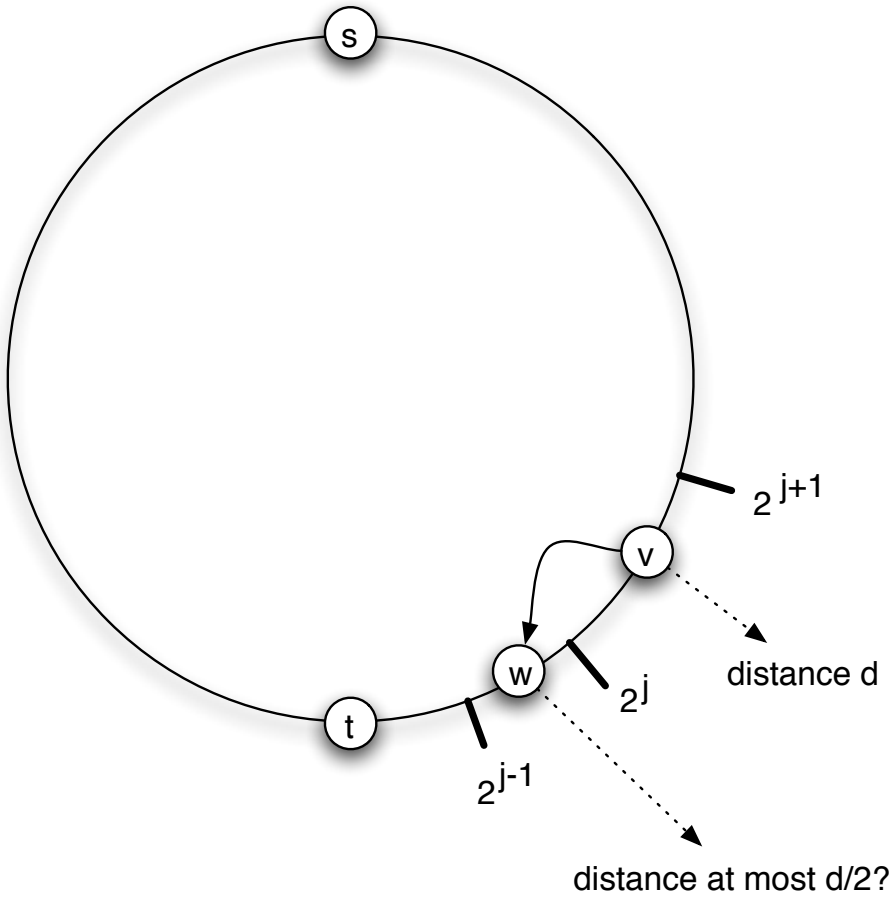
$$Z \leq 2(1+\ln(\tfrac{n}{2})) \leq 2+2\log_2(\tfrac{n}{2}) = 2\log_2 n.$$

- Thus, the probability $v$ links to $w$ is

$$\frac{1}{Z}d(v,w)^{-\alpha} \geq \frac{1}{2\log_2 n}d(v,w)^{-\alpha}.$$

# Time spent in phase $j$

Consider a particular phase $j$ of the search, when the message is at a node $v$ whose distance to the target $t$ is some number $d$ between $2^j$ and $2^{j+1}$.

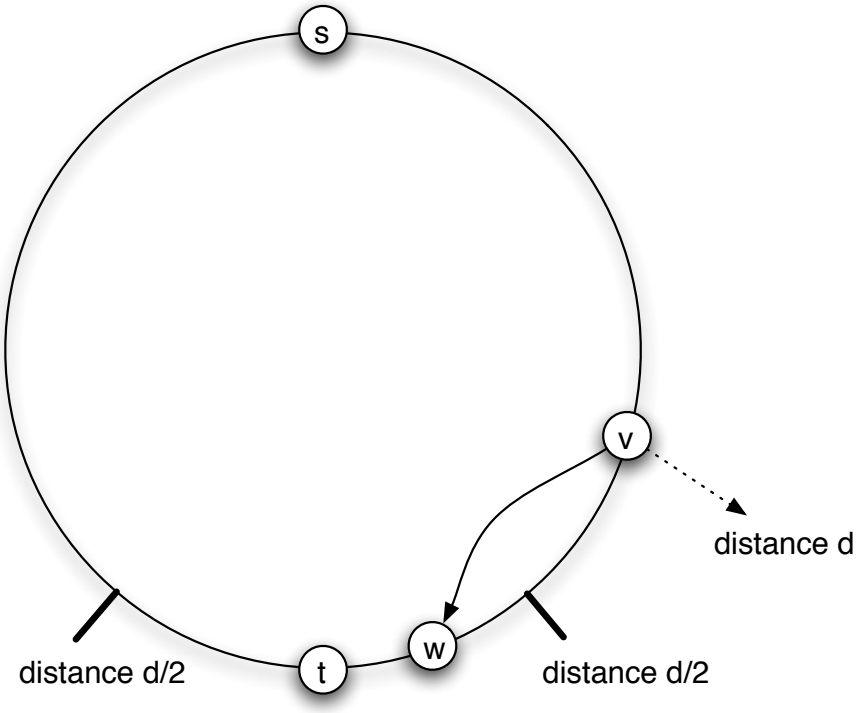


# Time spent in phase $j$ (Cont.)

- One way for phase $j$ to come to an end immediately would be for $v$'s long-range contact $w$ to be at distance $\leq \frac{d}{2}$ from $t$.
- In this case, $v$ would necessarily be the last node to belong to phase $j$.
- We want to show that this immediate halving of the distance in fact happens with reasonably large probability.

# Time spent in phase $j$ (Cont.)

Let $I$ be the set of nodes at distance $\leq \frac{d}{2}$ from $t$.



# Time spent in phase $j$ (Cont.)

- There are $d+1$ nodes in $I$: this includes node $t$ itself, and $\frac{d}{2}$ nodes consecutively on each side of it.
- Each node $w$ in $I$ has distance at most $\frac{3}{2}d$ from $v$: the farthest one is on the "far side" of $t$ from $v$, at distance $d+\frac{d}{2}$.
- Therefore, each node $w$ in $I$ has probability at least

$$\frac{1}{\log_2 n} d(v,w)^{-\alpha} \geq \frac{1}{\log_2 n} \frac{1}{3d/2} = \frac{1}{3d \log_2 n}$$

of being the long-range contact of $v$.

# Time spent in phase $j$ (Cont.)

- Since there are more than $d$ nodes in $I$, the probability that one of them is the long-range contact of $v$ is at least

$$d \cdot \frac{1}{3d \log_2 n} = \frac{1}{3 \log_2 n}.$$

- If one of these nodes is the long-range contact of $v$, then phase $j$ ends immediately in this step.
- Therefore, in each step that it proceeds, phase $j$ has a probability of at least $\frac{1}{3 \log_2 n}$ of coming to an end, independently of what has happened so far.
- To run for at least $i$ steps, phase $j$ has to fail to come to an end $(i-1)$ times in a row, and so the probability that phase $j$ runs for at least $i$ steps is at most

$$\left(1 - \frac{1}{3 \log_2 n}\right)^{i-1}.$$

# Time spent in phase $j$ (Cont.)

- We have

$$\mathbf{E}[X_j] = 1 \cdot \text{Prob}(X_j = 1) + 2 \cdot \text{Prob}(X_j = 2) + 3 \cdot \text{Prob}(X_j = 3) + \cdots .$$

- It is equal to

$$\text{Prob}(X_j \geq 1) + \text{Prob}(X_j \geq 2) + \text{Prob}(X_j \geq 3) + \cdots .$$

- Thus,

$$\begin{aligned}\mathbf{E}[X_j] &\leq 1 + \left(1 - \frac{1}{3\log_2 n}\right)^1 + \left(1 - \frac{1}{3\log_2 n}\right)^2 + \cdots \\ &= \frac{1}{1 - (1 - \frac{1}{3\log_2 n})} = 3\log_2 n.\end{aligned}$$

# Total time spent in myopic search

$$\mathbf{E}[X] = \mathbf{E}[X_1] + \mathbf{E}[X_2] + \cdots + \mathbf{E}[X_{\log_2 n}] \leq 3(\log_2 n)^2.$$

# $0 \leq \alpha < 1$

- Let $K$ be the set of all nodes within distance less than $n^\gamma$ of the target $t$, where $\gamma = \frac{1-\alpha}{2}$.
- 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 $\frac{1}{Z}$ with $Z = \sum d(v,w)^{-\alpha}$.
- A node has $\approx 2$ neighbors at distance $d$ from itself. This implies that

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

where the second relation follows using an integral approximation.

# $0 \leq \alpha < 1$ (Cont.)

- Hence, the probability that any one node has a long range contact inside the arc satisfies $\leq \frac{n^\gamma}{n^{2\gamma}} = n^{-\gamma}$ (since there are $n^\gamma$ nodes inside the arc).
- This shows that any decentralized search algorithm will need at least $n^\gamma = n^{\frac{1-\alpha}{2}}$ steps in expectation to find a node with a long-range contact in $K$.
- On the other hand, as long as it doesn't find a long-range link leading into $K$, it can't reach the target in less than $n^\gamma$ steps, since it would take this long to "walk" step-by-step through $K$ using only the connections among local contacts.
- From those, one can show that the expected time for any decentralized search strategy to reach $t$ must be at least proportional to $n^{\frac{1-\alpha}{2}}$.

# $\alpha > 1$

- The proportionality constant $Z =$ constant (independent of $n$) for large $n$.
- The expected length of a typical long-range contact is given by

$$\mathbf{E}[\text{length of a long-range contact}] = \frac{1}{Z}\sum_{d=1}^{n/2} d \cdot d^{-\alpha} \approx n^{2-\alpha},$$

  where the last relation follows by an integral approximation.
- Hence, the expected time is at least $\frac{n/2}{n^{2-\alpha}} \approx n^{\alpha-1}$.

# Interpretation

- When $\alpha < 1$, excessive search.

$\Rightarrow$ A set of nodes centered at $t$ is somehow "impenetrable"—very hard for the search to enter.

- When $\alpha > 1$, since even the long-range links are relatively short, it takes a long time for decentralized search to find links that span sufficiently long distances.

$\Rightarrow$ This makes it hard to quickly traverse the distance from the starting node to the target

# Two-dimensional model

- 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 $i, j \in \{1, \ldots, n\}$.
- For any two nodes $v$ and $w$, we define the distance between them $d(v, w)$ as the number of grid steps between them,

$$d\big((i, j), (k, \ell)\big) = |k - i| + |\ell - j|.$$

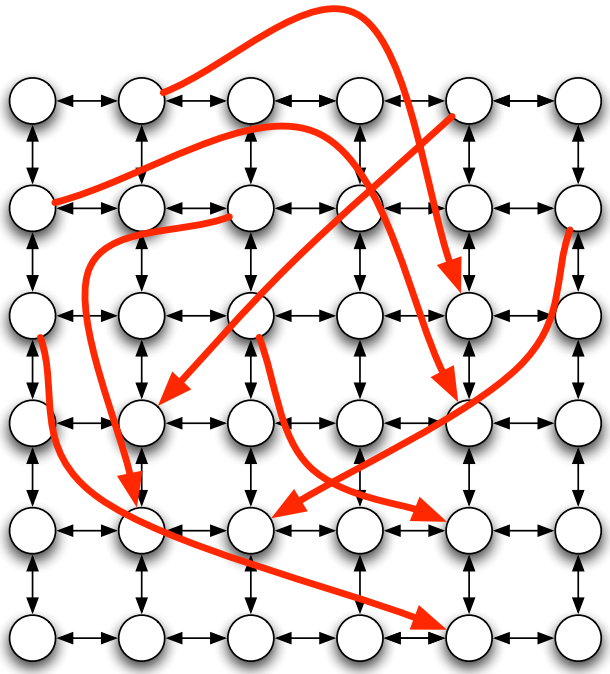
# Two-dimensional model (Cont.)

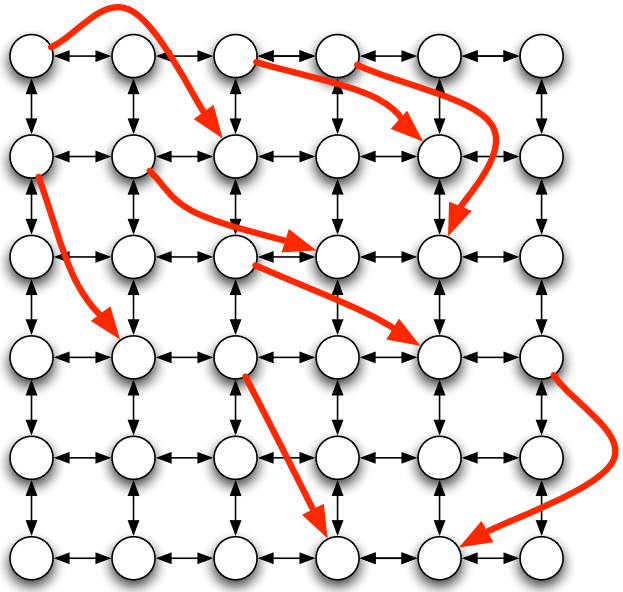
- Each node is connected to its 4 local neighbors directly—his local contacts.
- Each node also has a random edge to another node—his long range contact.

# Two-dimensional model (Cont.)

- The model has a parameter that controls the "scales spanned by the long-range links."
- The random edge is generated in a way that decays with distance, controlled by a clustering exponent $\alpha$: 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 $\alpha$ increases, the long-range contact of a node becomes more clustered in its vicinity on the grid.

# Two-dimensional model (Cont.)





- Left: A small clustering exponent
- Right: A large clustering exponent

# Efficiency of decentralized search

**Theorem (Kleinberg 2000)**

1. 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 $\beta_2$.
2. For $\alpha \in [0, 2)$, the expected delivery time of any decentralized algorithm is at least $\beta_\alpha n^{\frac{2-\alpha}{3}}$ for some constant $\beta_\alpha$.
3. For $\alpha \in (2, 3)$, the expected delivery time of any decentralized algorithm is at least $\beta_\alpha n^{\alpha-2}$ for some constant $\beta_\alpha$.

# From one-dimensional model

- First, we use the structure when we determine the normalizing constant $Z$.
- Second, we use the structure to argue that there are at least $d$ nodes within distance $\frac{d}{2}$ of the target $t$.

$\Rightarrow$ This factor of $d$ cancels the $d^{-1}$ in the link probability.

$\Rightarrow$ The probability of halving the distance to the target in any given step is at least proportional to $\frac{1}{\log_2 n}$, regardless of the value of $d$.

$^\star$ With link probability $d^{-1}$ on the ring, the probability of linking to any one node exactly offsets the number of nodes close to $t$, and so myopic search makes progress at every possible distance away from the target.

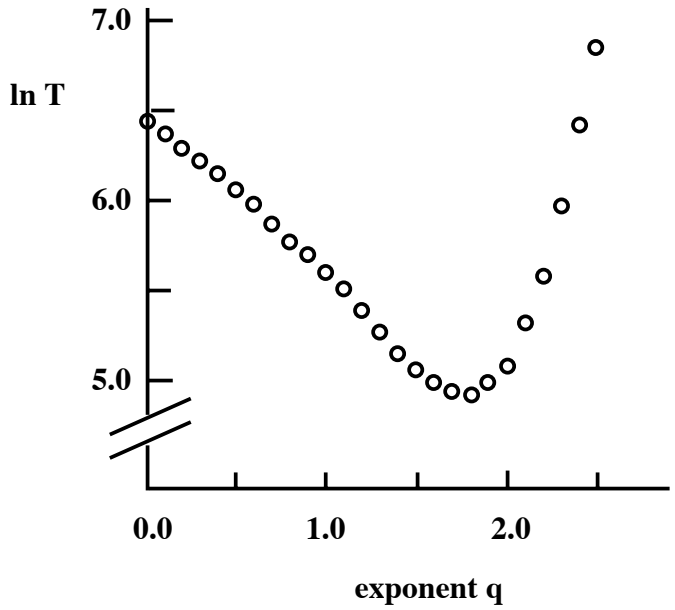
# From one-dimensional model (Cont.)

- In a two-dimensional model, there are at least $d^2$ nodes within distance $\frac{d}{2}$ of the target.

$\Rightarrow$ To get the same nice cancellation property, we should have $v$ link to each node $w$ with probability proportional to $d(v, w)^{-2}$, and this exponent $-2$ is what we will use.

- A similarly direct adaptation of the analysis shows that decentralized search is efficient for networks built by adding long-range contacts to grids in $D > 2$ dimensions, when the exponent $\alpha$ is equal to $D$.

# Simulation




- Each point is the average of 1000 runs on (a slight variant of) a grid with 400 million nodes.
- The delivery time is best in the vicinity of exponent $\alpha = 2$, as expected; but even with this number of nodes, the delivery time is comparable over the range between 1.5 and 2.