Centrality Measures and Link Analysis

Gonzalo Mateos
Dept. of Electrical and Computer Engineering
University of Rochester
gmateosb@ece.rochester.edu
http://www.ece.rochester.edu/~gmateosb/

February 26, 2015
Centrality measures

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
Quantifying vertex importance

- In network analysis many questions relate to vertex importance

Example

- **Q1**: Which actors in a social network hold the ‘reins of power’?
- **Q2**: How authoritative is a WWW page considered by peers?
- **Q3**: The ‘knock-out’ of which genes is likely to be lethal?
- **Q4**: How critical to the daily commute is a subway station?

- **Measures of vertex centrality** quantify such notions of importance
- Degrees are simplest centrality measures. Let’s study others
Closeness centrality

- **Rationale:** ‘central’ means a vertex is ‘close’ to many other vertices

- **Def:** Distance $d(u, v)$ between vertices $u$ and $v$ is the length of the shortest $u - v$ path. Oftentimes referred to as geodesic distance

- **Closeness centrality** of vertex $v$ is given by

$$c_{Cl}(v) = \frac{1}{\sum_{u \in V} d(u, v)}$$

- Interpret $v^* = \arg \max_v c_{Cl}(v)$ as the most approachable node in $G$
Normalization, computation and limitations

- To compare with other centrality measures, often normalize to $[0, 1]$
  
  $$c_{CI}(v) = \frac{N_v - 1}{\sum_{u \in V} d(u, v)}$$

- **Computation**: need all pairwise shortest path distances in $G$
- Can use Dijkstra’s algorithm in $O(N_v^2 \log N_v + N_v N_e)$ time

- **Limitation 1**: sensitivity, values tend to span a small dynamic range
- Difficult to discriminate between central and less central nodes

- **Limitation 2**: assumes connectivity, if not $c_{CI}(v) = 0$, for all $v \in V$
- Can compute centrality indices in different connected components
Betweenness centrality

- **Rationale**: ‘central’ node is (in the path) ‘between’ many vertex pairs

- Let $\sigma(s, t)$ be the total number of $s - t$ shortest paths

- Let $\sigma(s, t|v)$ be the number of $s - t$ shortest paths through $v \in V$

- **Betweenness centrality** of vertex $v$ is given by

$$c_{Be}(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma(s, t|v)}{\sigma(s, t)}$$

- Interpret $v^* = \arg \max_v c_{Be}(v)$ as the **controller of information flow**
Notice that a $s - t$ shortest path goes through $v$ if and only if

$$d(s, t) = d(s, v) + d(v, t)$$

Betweenness centralities can be naively computed for all $v \in V$ by:

1. **Step 1**: Modifying Dijkstra’s algorithm to compute tables with $d(s, t)$ and $\sigma(s, t)$ for all vertex pairs $s, t$
2. **Step 2**: Using the tables to identify $\sigma(s, t|v)$, and summing the fractions to obtain $c_{Be}(v)$ for all $v \in V$ in $O(N^3_v)$ time

Cubic complexity can be prohibitive for large networks

An $O(N_vN_e)$-time algorithm for unweighted graphs is given in:

Eigenvector centrality

- **Rationale**: ‘central’ vertex if ‘in-neighbors’ are themselves important
- Compare with ‘importance-agnostic’ degree centrality
- Eigenvector centrality of vertex $v$ is implicitly defined as

$$c_{Ei}(v) = \alpha \sum_{(u,v) \in E} c_{Ei}(u)$$

- No one points to 1
- Only 1 points to 2
- Only 2 points to 3, but 2 more important than 1
- 4 as high as 5 with less links
- Links to 5 have lower rank
- Same for 6
Eigenvalue problem

- Recall the adjacency matrix $A$ and
  \[
c_{Ei}(v) = \alpha \sum_{(u,v) \in E} c_{Ei}(u)
  \]

- Vector $c_{Ei} = [c_{Ei}(1), \ldots, c_{Ei}(N_v)]^\top$ solves the eigenvalue problem
  \[
  Ac_{Ei} = \alpha^{-1} c_{Ei}
  \]

- Typically, $\alpha^{-1}$ chosen as the largest eigenvalue of $A$ [Bonacich'87]

- If $G$ is undirected and connected, by Perron’s Theorem then
  - The largest eigenvalue of $A$ is positive and simple
  - All the entries in the dominant eigenvector $c_{Ei}$ are positive

- Can compute $c_{Ei}$ and $\alpha^{-1}$ via $O(N_v^2)$ complexity power iterations
  \[
c_{Ei}(k + 1) = \frac{Ac_{Ei}(k)}{\|Ac_{Ei}(k)\|}, \quad k = 0, 1, \ldots
  \]
Example: Comparing centrality measures

- **Q:** Which vertices are more central? It depends on the context

- The small green vertices are arguably more peripheral, less central

- It is less clear how the yellow, dark blue and red vertices compare
Centrality measures

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
The problem of ranking websites

- Search engines rank pages by looking at the Web itself
- Enough information intrinsic to the Web and its structure

- Information retrieval is a historically difficult problem
- Keywords simplify complex information needs (synonymy, polysemy)
- Diversity of authoring styles, people issuing queries
- Beyond explosion in scale, unique issues arised with the Web
  - Dynamic and constantly changing content
  - Paradigm: from scarcity to abundance

- Finding and indexing documents that are relevant is ‘easy’
- Q: Which few of these should the engine recommend?
- Key is understanding the structure of Web pages, i.e., link analysis
Voting by in-links

▶ Ex: Suppose we issue the query ‘newspapers’
▶ First, use text-only information retrieval to identify relevant pages

▶ Idea: Links suggest implicit endorsements of other relevant pages
▶ Count in-links to assess the authority of a page on ‘newspapers’
A list-finding technique

- Query also returns pages that compile lists of relevant resources
- These hubs voted for many pages that received high endorsements

- **Idea:** Good lists have a better sense of where the good results are
- Page’s **hub** value is the *sum of votes received by its linked pages*
Repeated improvement

- Reasonable to weight more the votes of pages scoring well as lists
- Recompute votes summing linking page values as lists

- Why stop here? Use improved votes to refine the list scores as well
- Alternating refinements known as principle of repeated improvement
Relevant pages fall in two categories: hubs and authorities.

Authorities are pages with useful, relevant content:
- Newspaper home pages
- Course home pages
- Auto manufacturer home pages

Hubs are ‘expert’ lists pointing to multiple authorities:
- List of newspapers
- Course bulletin
- List of US auto manufacturers

Key idea: self-reinforcing recursive definition based on link structure:
- A good hub links to multiple good authorities
- A good authority is linked from multiple good hubs
Hubs and authorities ranking algorithm

- Hyperlink-Induced Topic Search (HITS) algorithm [Kleinberg’98]
- Each page \( v \in V \) has a hub score \( h_v \) and authority score \( a_v \)
- Network-wide vectors \( h = [h_1, \ldots, h_{N_v}]^\top \), \( a = [a_1, \ldots, a_{N_v}]^\top \)

Authority update rule:

\[
a_v(k) = \sum_{(u,v) \in E} h_u(k - 1), \text{ for all } v \in V \iff a(k) = A^\top h(k - 1)
\]

Hub update rule:

\[
h_v(k) = \sum_{(v,u) \in E} a_u(k), \text{ for all } v \in V \iff h(k) = Aa(k)
\]

- Initialize \( h(0) = 1/\sqrt{N_v} \), normalize \( a(k) \) and \( h(k) \) each iteration
Define the hub and authority rankings as

\[ a := \lim_{k \to \infty} a(k), \quad h := \lim_{k \to \infty} h(k) \]

From the HITS update rules one finds for \( k = 0, 1, \ldots \)

\[ a(k+1) = \frac{A^\top A a(k)}{\|A^\top A a(k)\|}, \quad h(k+1) = \frac{A A^\top h(k)}{\|A A^\top h(k)\|} \]

Power iterations converge to dominant eigenvectors of \( A^\top A \) and \( A A^\top \)

\[ A^\top A a = \alpha_a^{-1} a, \quad A A^\top h = \alpha_h^{-1} h \]

The hub and authority ranks are eigenvector centrality measures
Link analysis beyond the web

- Link analysis of citations among US Supreme Court opinions

- Rise and fall of authority of key Fifth Amendment cases [Fowler-Jeon’08]
PageRank

- **Node rankings** to measure website relevance, social influence
- **Key idea**: in-links as votes, but ‘not all links are created equal’
  - How many links point to a node (outgoing links irrelevant)
  - How important are the links that point to a node

- **PageRank** key to Google’s original ranking algorithm [Page-Brin’98]
- **Intuition 1**: fluid that percolates through the network
  - Eventually accumulates at most relevant Web pages
- **Intuition 2**: random web surfer (more soon)
  - In the long-run, relevant Web pages visited more often

- PageRank and HITS success was quite different after 1998
Basic PageRank update rule

- Each page \( v \in V \) has PageRank \( r_v \), and let \( r = [r_1, \ldots, r_{N_v}]^T \)
- Define \( P := (D^\text{out})^{-1}A \), where \( D^\text{out} \) is the out-degree matrix
- PageRank update rule:
  \[
  r_v(k) = \sum_{(u,v) \in E} \frac{r_u(k-1)}{d^\text{out}_u}, \text{ for all } v \in V \iff r(k) = P^T r(k-1)
  \]
- Split current PageRank evenly among outgoing links and pass it on
- New PageRank is the total fluid collected in the incoming links
- Initialize \( r(0) = 1/N_v \), flow conserved so no normalization needed
- **Problem:** ‘Spider traps’
- Accumulate all PageRank
- Only when not strongly connected
Scaled PageRank update rule

- Apply the basic PageRank rule and scale the result by $s \in (0, 1)$
- Split the leftover $(1 - s)$ evenly among all nodes (evaporation-rain)

Scaled PageRank update rule:

$$r_v(k) = s \times \sum_{(u,v) \in E} \frac{r_u(k - 1)}{d_u^{out}} + \frac{1 - s}{N_v}, \text{ for all } v \in V$$

- Can view as basic update $r(k) = \bar{P}^T r(k - 1)$ with

$$\bar{P} := sP + (1 - s) \frac{11^\top}{N_v}$$

- Scaling factor $s$ typically chosen between 0.8 and 0.9
- Power iteration converges to the dominant eigenvector of $\bar{P}$
A primer on Markov chains

Centrality measures

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
Markov chains

- Consider discrete-time index \( n = 0, 1, 2, \ldots \)
- Time-dependent random state \( X_n \) takes values on a countable set
  - In general denote states as \( i = 0, 1, 2, \ldots \), i.e., here the state space is \( \mathbb{N} \)
  - If \( X_n = i \) we say “the process is in state \( i \) at time \( n \)"
- Denote random process as \( X_N \) and its history \( X_n = [X_n, X_{n-1}, \ldots, X_0]^T \)
- The process \( X_N \) is a Markov chain (MC) if for all \( n \geq 1, i, j, x \in \mathbb{N}^n \)

\[
P(X_{n+1} = j \mid X_n = i, X_{n-1} = x) = P(X_{n+1} = j \mid X_n = i) = P_{ij}
\]

- Future depends only on current state \( X_n \) (memoryless, Markov property)
- Future conditionally independent of the past, given the present
Group the $P_{ij}$ in a transition probability “matrix” $\mathbf{P}$

$$
\mathbf{P} = \begin{pmatrix}
P_{00} & P_{01} & P_{02} & \ldots & P_{0j} & \ldots \\
P_{10} & P_{11} & P_{12} & \ldots & P_{1j} & \ldots \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
P_{i0} & P_{i1} & P_{i2} & \ldots & P_{ij} & \ldots \\
\vdots & \vdots & \vdots & \ddots & \vdots 
\end{pmatrix}
$$

- Not really a matrix if number of states is infinite
- Row-wise sums should be equal to one, i.e., $\sum_{j=0}^{\infty} P_{ij} = 1$ for all $i$
A graph representation or **state transition diagram** is also used

Useful when number of states is infinite, skip arrows if $P_{ij} = 0$

Again, sum of per-state **outgoing** arrow weights should be one
Example: Bipolar mood

- I can be happy ($X_n = 0$) or sad ($X_n = 1$)
- My mood tomorrow is only affected by my mood today
- Model as Markov chain with transition probabilities

$$P = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}$$

- Inertia $\Rightarrow$ happy or sad today, likely to stay happy or sad tomorrow (higher $P_{00} = 0.8$, $P_{11} = 0.7$)
- But when sad, a little less likely so ($P_{00} > P_{11}$)
Example: Random (drunkard’s) walk

- Step to the right w.p. \( p \), to the left w.p. \( 1 - p \)
- Not that drunk to stay on the same place

\[
\begin{align*}
\text{States are } 0, \pm 1, \pm 2, \ldots \text{ (state space is } \mathbb{Z} \text{), infinite number of states} \\
\text{Transition probabilities are} \\
\mathbb{P}_{i,i+1} = p, \quad \mathbb{P}_{i,i-1} = 1 - p \\
\text{\( \mathbb{P}_{ij} = 0 \) for all other transitions}
\end{align*}
\]
Multiple-step transition probabilities

▶ What can be said about multiple transitions?
▶ Probabilities of $X_{m+n}$ given $X_m \Rightarrow n$-step transition probabilities

$$P_{ij}^n = P (X_{m+n} = j \mid X_m = i)$$

▶ Define the matrix $P^{(n)}$ with elements $P_{ij}^n$

**Theorem**

The matrix of $n$-step transition probabilities $P^{(n)}$ is given by the $n$-th power of the transition probability matrix $P$, i.e.,

$$P^{(n)} = P^n$$

Henceforth we write $P^n$
Unconditional probabilities

- All probabilities so far are conditional, i.e., $P_{ij}^n = P(X_n = j \mid X_0 = i)$
- Want unconditional probabilities $p_j(n) = P(X_n = j)$
- Requires specification of initial conditions $p_i(0) = P(X_0 = i)$
- Using law of total probability and definitions of $P_{ij}^n$ and $p_j(n)$

$$p_j(n) = P(X_n = j) = \sum_{i=0}^{\infty} P(X_n = j \mid X_0 = i) P(X_0 = i)$$

$$= \sum_{i=0}^{\infty} P_{ij}^n p_i(0)$$

- Or in matrix form (define vector $\mathbf{p}(n) = [p_1(n), p_2(n), \ldots]^T$)

$$\mathbf{p}(n) = (\mathbf{P}^n)^T \mathbf{p}(0)$$
Limiting distributions

- MCs have one-step memory. Eventually they forget initial state.
- What can we say about probabilities for large $n$?

$$
\pi_j := \lim_{n \to \infty} P(X_n = j \mid X_0 = i) = \lim_{n \to \infty} P_i^n
$$

- Implicitly assumed that limit is independent of initial state $X_0 = i$.
- We’ve seen that this problem is related to the matrix power $P^n$.

$$
P = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}, \quad P^2 = \begin{pmatrix} 0.7 & 0.3 \\ 0.45 & 0.55 \end{pmatrix}, \quad P^7 = \begin{pmatrix} 0.6031 & 0.3969 \\ 0.5953 & 0.4047 \end{pmatrix}, \quad P^{30} = \begin{pmatrix} 0.6000 & 0.4000 \\ 0.6000 & 0.4000 \end{pmatrix}
$$

- Matrix product converges $\Rightarrow$ probs. independent of time (large $n$).
- All rows are equal $\Rightarrow$ probs. independent of initial condition.
Theorem
For an ergodic (i.e., irreducible, aperiodic and positive recurrent) MC, \( \lim_{n \to \infty} P^n_{ij} \) exists and is independent of the initial state \( i \), i.e.,

\[
\pi_j = \lim_{n \to \infty} P^n_{ij}
\]

Furthermore, steady-state probabilities \( \pi_j \geq 0 \) are the unique nonnegative solution of the system of linear equations

\[
\pi_j = \sum_{i=0}^{\infty} \pi_i P_{ij}, \quad \sum_{j=0}^{\infty} \pi_j = 1
\]

- Limit probs. independent of initial condition exist for ergodic MC
- Simple algebraic equations can be solved to find \( \pi_j \)
Markov chains meet eigenvalue problems

- Define vector steady-state distribution \( \pi := [\pi_0, \pi_1, \ldots, \pi_J]^T \)
- Limit distribution is unique solution of
  \[
  \pi = P^T \pi, \quad \pi^T 1 = 1
  \]
- Eigenvector \( \pi \) associated with eigenvalue 1 of \( P^T \)
  - Eigenvectors are defined up to a scaling factor
  - Normalize to sum 1
- All other eigenvalues of \( P^T \) have modulus smaller than 1
  - If not, \( P^n \) diverges, but we know \( P^n \) contains \( n \)-step transition probs.
  - Eigenvector \( \pi \) associated with largest eigenvalue of \( P^T \)
- Computing \( \pi \) as eigenvector is computationally efficient and robust
Ergodicity

- Define $T_i^{(n)}$ as fraction of time spent in $i$-th state up to time $n$

$$T_i^{(n)} := \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}\{X_m = i\}$$

- Compute expected value of $T_i^{(n)}$

$$\mathbb{E}\left[T_i^{(n)}\right] = \frac{1}{n} \sum_{m=1}^{n} \mathbb{E}[\mathbb{I}\{X_m = i\}] = \frac{1}{n} \sum_{m=1}^{n} P(X_m = i) \to \pi_i$$

- As time $n \to \infty$, probabilities $P(X_m = i)$ approach $\pi_i$. Then

$$\lim_{n \to \infty} \mathbb{E}\left[T_i^{(n)}\right] = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} P(X_m = i) = \pi_i$$

- For ergodic MCs same is true without expected value $\Rightarrow$ ergodicity

$$\lim_{n \to \infty} T_i^{(n)} = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}\{X_m = i\} = \pi_i, \text{ a.s.}$$
Consider an ergodic Markov chain with transition probability matrix

\[ P := \begin{pmatrix} 0 & 0.3 & 0.7 \\ 0.1 & 0.5 & 0.4 \\ 0.1 & 0.2 & 0.7 \end{pmatrix} \]

Visits to states, \( nT_i^{(n)} \)

Ergodic averages, \( T_i^{(n)} \)

Ergodic averages slowly converge to \( \pi = [0.09, 0.29, 0.61]^T \)
Centrality measures

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
Graph $G = (V, E)$ ⇒ vertices $V = \{1, 2, \ldots, J\}$ and edges $E$

- Outgoing neighborhood of $i$ is the set of nodes $j$ to which $i$ points:
  \[ n(i) := \{ j : (i, j) \in E \} \]

- Incoming neighborhood, $n^{-1}(i)$ is the set of nodes that point to $i$:
  \[ n^{-1}(i) := \{ j : (j, i) \in E \} \]

- Strongly connected $G$ ⇒ directed path joining any pair of nodes
Definition of rank

- Agent $A$ chooses node $i$, e.g., web page, at random for initial visit
- Next visit randomly chosen between links in the neighborhood $n(i)$
  - All neighbors chosen with equal probability
- If reach a dead end because node $i$ has no neighbors
  - Chose next visit at random equiprobably among all nodes
- Redefine $G = (V, E)$ adding edges from dead ends to all nodes
- Restrict attention to connected (modified) graphs

- Rank of node $i$ is the average number of visits of agent $A$ to $i$
Formally, let \( A_n \) be the node visited at time \( n \)

Define transition probability \( P_{ij} \) from node \( i \) into node \( j \)

\[
P_{ij} := P(A_{n+1} = j \mid A_n = i)
\]

Next visit equiprobable among neighbors

\[
P_{ij} = \frac{1}{|n(i)|} = \frac{1}{d_i^{\text{out}}}, \quad \text{for all} \ j \in n(i)
\]

Still have a graph

But also a MC

Red (not blue) circles
Formal definition of rank

- Let $\mathbb{I}\{A_m = i\}$ indicate the visit to state $i$ at time $m$
- Rank $r_i$ of $i$-th node defined as time average of number of visits, i.e.,

$$r_i := \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}\{A_m = i\}$$

- Define vector of ranks $\mathbf{r} := [r_1, r_2, \ldots, r_J]^T$
- Rank $r_i$ can be approximated by average $r_{ni}$ at time $n$

$$r_{ni} := \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}\{A_m = i\}$$

- Since $\lim_{n \to \infty} r_{ni} = r_i$, it holds $r_{ni} \approx r_i$ for $n$ sufficiently large
- Define vector of approximate ranks $\mathbf{r}_n := [r_{n1}, r_{n2}, \ldots, r_{nJ}]^T$
- If modified graph is connected, rank independent of initial visit
Output : Vector \( r(i) \) with ranking of node \( i \)
Input : Scalar \( n \) indicating maximum number of iterations
Input : Vector \( N(i) \) containing number of neighbors of \( i \)
Input : Matrix \( N(i,j) \) containing indices \( j \) of neighbors of \( i \)

\[
m = 1; \quad r = \text{zeros}(J,1); \quad \% \text{Initialize time and ranks}
A_0 = \text{random('unid',}J); \quad \% \text{Draw first visit uniformly at random}
\text{while } m < n \text{ do}
 \quad \text{jump} = \text{random('unid',}d_{A_{m-1}}^{\text{out}}); \quad \% \text{Neighbor uniformly at random}
 \quad A_m = N(A_{m-1}, \text{jump}); \quad \% \text{Jump to selected neighbor}
 \quad r(A_m) = r(A_m) + 1; \quad \% \text{Update ranking for } A_m
 \quad m = m + 1;
\text{end}
\]
\[
r = r/n; \quad \% \text{Normalize by number of iterations } n
\]
Social graph example

- Asked probability students about homework collaboration
- Created (crude) graph of the social network of students in the class
- Used ranking algorithm to understand connectedness

**Ex:** To know how well students are coping with the class it is best to ask people with higher connectivity ranking

- 2009 data from “UPenn’s Introduction to Random Processes”
Convergence metrics

- Recall $r$ is vector of ranks and $r_n$ of rank iterates
- By definition $\lim_{n \to \infty} r_n = r$. How fast $r_n$ converges to $r$ (r given)?
- Can measure by distance between $r$ and $r_n$

$$\zeta_n := \|r - r_n\|_2 = \left( \sum_{i=1}^{J} (r_{ni} - r_i)^2 \right)^{1/2}$$

- If interest is only on highest ranked nodes, e.g., a web search
- Denote $r^{(i)}$ as the index of the $i$-th highest ranked node
- Similarly, $r_n^{(i)}$ is the index of the $i$-th highest ranked node at time $n$
- First element wrongly ranked at time $n$

$$\xi_n := \arg\min_i r^{(i)} \neq r_n^{(i)}$$
Evaluation of convergence metrics

**Distance**

- Distance gets close to $10^{-2}$ in approx. $5 \times 10^3$ iterations

- **Bad:** Two highest ranks in $4 \times 10^3$ iterations

- **Awful:** Six best ranks in $8 \times 10^3$ iterations

- Convergence appears (very) slow
When does this algorithm converge?

- Cannot confidently claim convergence until $10^5$ iterations
- True for particular case. Slow convergence inherent to algorithm
- Example has 40 nodes, want to use in network with $10^9$ nodes

Use fact that this process is a MC to obtain a faster algorithm
Centrality measures

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
Limit probabilities

- Recall definition of rank $r_i := \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} I \{A_m = i\}$

- Rank is time average of number of state visits in a MC
  $\Rightarrow$ Can be equally obtained from limiting probabilities

- Recall transition probabilities $P_{ij} = \frac{1}{d_{i}^{\text{out}}}$, for all $j \in n(i)$

- Stationary distribution $\pi = [\pi_1, \pi_1, \ldots, \pi_J]^T$ solution of
  $$\pi_i = \sum_{j \in n^{-1}(i)} P_{ji} \pi_j = \sum_{j \in n^{-1}(i)} \frac{\pi_j}{d_{j}^{\text{out}}}$$
  for all $i$

- Plus normalization equation $\sum_{i=1}^{J} \pi_i = 1$

- As per ergodicity (strong connectivity of $G$) $\Rightarrow r = \pi$
Matrix notation, eigenvalue problem

- As always, can define matrix $P$ with elements $P_{ij}$

$$\pi_i = \sum_{j \in n^{-1}(i)} P_{ji} \pi_j = \sum_{j=1}^{J} P_{ji} \pi_j \quad \text{for all } i$$

- Right hand side is just definition of a matrix product leading to

$$\pi = P^T \pi, \quad \pi^T 1 = 1$$

- Also added normalization equation

- **Idea:** solve system of linear equations or eigenvalue problem on $P^T$

- Requires matrix $P$ available at a central location

- **Computationally costly** (matrix $P$ with $10^9$ rows and columns)
  - All methods are costly to compute exact solution
  - This one is even costly to find an approximate solution
What are the limit probabilities?

- Let $p_i(n)$ denote probability of agent $A$ visiting node $i$ at time $n$

  $p_i(n) := P(A_n = i)$

- Probabilities at time $n+1$ and $n$ can be related

  $P(A_{n+1} = i) = \sum_{j \in n^{-1}(i)} P(A_{n+1} = i \mid A_n = j) P(A_n = j)$

- Which is, of course, probability propagation in a MC

  $p_i(n+1) = \sum_{j \in n^{-1}(i)} P_{ji} p_j(n)$

- By definition limit probabilities are (let $p(n) = [p_1(n), \ldots, p_J(n)]^T$)

  $\lim_{n \to \infty} p(n) = \pi = r$

- Compute ranks from limit of propagated probabilities
Probability propagation

- Can also write probability propagation in matrix form

\[
p_i(n + 1) = \sum_{j \in n^{-1}(i)} P_{ji} p_j(n) = \sum_{j=1}^{J} P_{ji} p_j(n) \quad \text{for all } i
\]

- Right hand side is just definition of a matrix product leading to

\[
p(n + 1) = P^T p(n)
\]

- Idea: can approximate rank by large \( n \) probability distribution

\[ r = \lim_{n \to \infty} p(n) \approx p(n) \quad \text{for } n \text{ sufficiently large} \]
Ranking algorithm

- Algorithm is just a recursive matrix product, a power iteration

Output: Vector \( r(i) \) with ranking of node \( i \)
Input: Scalar \( n \) indicating maximum number of iterations
Input: Matrix \( P \) containing transition probabilities

\[ m = 1; \quad \% \text{Initialize time} \]
\[ r = (1/J)\text{ones}(J,1); \quad \% \text{Initial distribution uniform across all nodes} \]

\textbf{while} \( m < n \) \textbf{do}
\[
\begin{align*}
    r &= P^T r; \quad \% \text{Probability propagation} \\
    m &= m + 1;
\end{align*}
\]
\textbf{end}
Interpretation of probability propagation

- Why does the random walk converge so slow?
- What does it take to obtain a time average $r_{ni}$ close to $r_i$?
- Need to register a large number of agent visits to every state
- Back of envelope: 40 nodes, say 100 visits to each $\Rightarrow 4 \times 10^3$ iters.
- Smart idea: Unleash a large number of agents $K$

$$r_i = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \frac{1}{K} \sum_{k=1}^{K} I\{A_{km} = i\}$$

- Visits are now spread over time and space
  $\Rightarrow$ Converges “$K$ times faster”
  $\Rightarrow$ But haven’t changed computational cost
What happens if we unleash infinite number of agents $K$?

$$r_i = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \lim_{K \to \infty} \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}\{A_{km} = i\}$$

Using law of large numbers and expected value of indicator function

$$r_i = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{E}[\mathbb{I}\{A_m = i\}] = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} P(A_m = i)$$

Graph walk is a MC, then \( \lim_{m \to \infty} P(A_m = i) = \lim_{m \to \infty} p_i(m) \) exists, and

$$r_i = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} p_i(m) = \lim_{n \to \infty} p_i(n)$$

Probability propagation $\approx$ Unleashing infinite number of agents
Distance to rank

- Initialize with uniform probability distribution $\Rightarrow p(0) = (1/J)1$
- Distance between $p(n)$ and $r$

- Distance is $10^{-2}$ in approximately 30 iters., $10^{-4}$ in 140 iters.
- Convergence is two orders of magnitude faster than random walk
Number of nodes correctly ranked

- Rank of highest ranked node that is wrongly ranked by time \( n \)

- Not bad: All nodes correctly ranked in 120 iterations
- Good: Six best ranks in 70 iterations
- Great: Four best ranks in 20 iterations
Distributed algorithm to compute ranks

- Nodes want to compute their rank \( r_i \)
  - Can communicate with neighbors only (incoming + outgoing)
  - Access to neighborhood information only

- Recall probability update

\[
p_{i}(n + 1) = \sum_{j \in n^{-1}(i)} P_{ji} p_{j}(n) = \sum_{j \in n^{-1}(i)} \frac{1}{d_{j}^{\text{out}}} p_{j}(n)
\]

- Uses local information only

- Distributed algorithm. Nodes keep local rank estimates \( p_{i}(n) \)
  - Receive rank (probability) estimates \( p_{j}(n) \) from neighbors \( j \in n^{-1}(i) \)
  - Update local rank estimate \( p_{i}(n + 1) = \sum_{j \in n^{-1}(i)} p_{j}(n) / d_{j}^{\text{out}} \)
  - Communicate rank estimate \( p_{i}(n + 1) \) to outgoing neighbors \( j \in n(i) \)

- Only need to know the number of neighbors of my neighbors
Distributed implementation of random walk

- Can communicate with neighbors only (incoming + outgoing)
- But cannot access neighborhood information
- Pass agent (‘hot potato’) around
- Local rank estimates $r_i(n)$ and counter with number of visits $V_i$
- Algorithm run by node $i$ at time $n$

```plaintext
if Agent received from neighbor then
  $V_i = V_i + 1$
  Choose random neighbor
  Send agent to chosen neighbor
end

$n = n + 1; r_i(n) = V_i/n$;
```

- Speed up convergence by generating many agents to pass around
Comparison of different algorithms

- Random walk (RW) implementation
  - Most secure & robust. No information shared with other nodes
  - Implementation can be distributed
  - Convergence exceedingly slow

- System of linear equations
  - Least security and robustness. Graph in central server
  - Distributed implementation not clear
  - Convergence not an issue
  - But computationally costly to obtain approximate solutions

- Probability propagation
  - Somewhat secure & robust. Information shared with neighbors only
  - Implementation can be distributed
  - Convergence rate acceptable (orders of magnitude faster than RW)
Glossary

- Centrality measure
- Closeness centrality
- Dijkstra’s algorithm
- Betweenness centrality
- Information controller
- Eigenvector centrality
- Perron’s Theorem
- Power method
- Information retrieval
- Link analysis
- Repeated improvement
- Hubs and authorities
- HITS algorithm
- PageRank
- Spider traps
- Scaled PageRank updates
- Ergodic Markov chain
- Limiting probabilities
- Random walk on a graph
- Long-run fraction of state visits
- Probability propagation
- Distributed algorithm