Ranking of nodes in graphs

Alejandro Ribeiro & Cameron Finucane
Dept. of Electrical and Systems Engineering
University of Pennsylvania
aribeiro@seas.upenn.edu
http://www.seas.upenn.edu/users/~aribeiro/

October 13, 2010
Ranking of nodes in graphs: Random walk

Ranking of nodes in graphs: Markov chain
Graphs

- **Graph** ⇒ A set of $J$ nodes $j = 1, \ldots, J$
  ⇒ Connected by a set of edges $E$ defined as ordered pairs $(i, j)$

- **In figure** ⇒ nodes are $j = 1, 2, 3, 4, 5$,
  ⇒ edges $E = \{(1, 2), (1, 5), (2, 3), (2, 5), (3, 4), \ldots, (3, 6), (4, 5), (4, 6), (5, 4)\}$

- **Websites and links** ⇒ “The” web.

- **People and friendship** ⇒ Social network
How well connected nodes are?

- Q: Which node is the most connected? A: Define most connected
- Can define “most connected” in different ways
- Node rankings to measure website quality, social influence
- There are two important connectivity indicators
  - How many nodes point to a link (outgoing links irrelevant)
  - How important are the links that point to a node
Connectivity ranking

- Insight ⇒ There is information in the structure of the network
- Knowledge is distributed through the network
  ⇒ The network (not the nodes) know the rankings
- Idea exploited by Google’s PageRank© to rank webpages
- ... by social scientists to study trust & reputation in social networks
- ... by ISI to rank scientific papers, transactions & magazines ...

- 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
Preliminary definitions

- Graph $\mathcal{G} = (V, E)$ ⇒ sets of vertices $V = \{1, 2, \ldots, J\}$ and edges $E$
- Edges (elements of $E$) are ordered pairs $(i, j)$
- We say there is a connection from $i$ to $j$
- 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 $i$:
  \[ n^{-1}(i) := \{ j : (j, i) \in E \} \]
- Connected graph
  ⇒ There is a path from any node to any other node
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 $n(i)$
  $\Rightarrow$ All neighbors chosen with equal probability
- If reach a dead end because node $i$ has no neighbors
  $\Rightarrow$ Chose next visit at random equiprobably among all nodes
- Redefine graph $G = (V, E)$ 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 $A$ to $i$
Equiprobable random walk

- Formally, let $A_n$ be the node visited at time $n$
- Define transition probability $P_{ij}$ from node $i$ into node $j$
  \[ P_{ij} := P \left[ A_{n+1} = j \mid A_n = i \right] \]
- Next visit equiprobable among neighbors
  \[ P_{ij} = \frac{1}{\#[n(i)]} = \frac{1}{N_i}, \quad \text{for all } j \in n(i) \]
- Defined number of neighbors $N_i = \#[n(i)]$

Still have a graph
- But also a MC
- Red (not blue) circles
Consider variable $\mathbb{I}\{A_m = i\}$ to indicate 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.
Algorithm

Output: Vector $r(i)$ with ranking of node $i$
Input: Vector $N(i)$ containing number of neighbors of $i$
Input: Matrix $N(i, k)$ containing indices $j$ of neighbors of $i$

$m = 1; \ r=\text{zeros}(J,1); \ \% \ \text{Initialize time and ranks}$
$A_0 = \text{random('unid',}J); \ \% \ \text{Draw first visit uniformly at random}$

while $m < n$ do
    jump = random('unid', $N_{A_{m-1}}$); \ \% \ Neighbor uniformly at random
    $A_m = N(A_{m-1}, \text{jump}); \ \% \ \text{Jump to selected neighbor}$
    $r(A_m) = r(A_m) + 1; \ \% \ \text{Update ranking for } A_m$
    $m = m + 1;$
end

$r = r/n; \ \% \ \text{Normalize by number of iterations } n$
Example: Social graph

- Asked students taking ESE303 about homework collaboration
- Created (crude) graph of the social network of students in this class
- Used ranking algorithm to understand connectedness

- E.g., If I want to know how well students are coping with the class it is best to ask people with higher connectivity ranking

- 2009 data
- Students in 2010 don’t show up in class in enough numbers
Ranked class graph

Stoch. Systems Analysis

Ranking of nodes in graphs
Convergence metrics

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

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

- If interest is only on largest 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 $n$

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

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

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

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

- Can confidently claim convergence not 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 faster algorithm
Ranking of nodes in graphs: Random walk

Ranking of nodes in graphs: Markov chain
Recall definition of rank \( r_i := \lim_{t \to \infty} \frac{1}{t} \sum_{u=1}^{t} \mathbb{I}\{A(u) = i\} \)

Rank is time average of number of state visits in a MC

- Can be equally obtained from limiting probabilities

Recall transition probabilities \( P_{ij} = \frac{1}{N_i} \), for all \( j \in n(i) \)

Stationary distribution \( \pi = [\pi_1, \pi_1, \ldots, \pi_J]^T \) solution of

\[
\pi_i = \sum_{j \in n^{-1}(i)} P_{ji} \pi_j = \sum_{j \in n^{-1}(i)} \frac{\pi_j}{N_j} \quad \text{for all } i
\]

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

- As per ergodicity \( \Rightarrow r = \pi \)
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

Can solve as system of linear equations or eigenvalue problem on $P^T$

Non-iterative method $\Rightarrow$ Convergence not an issue

But 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 costly to find even approximate solution
What are limit probabilities?

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

  \[ p_i(n) := P[A_n = i] \]

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

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

- 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 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

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

Can approximate rank by probability distribution

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

- Algorithm is just a recursive matrix product

Output : Vector \( r(i) \) with ranking of node \( i \)
Input : Matrix \( P \) containing transition probabilities

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

\[
\text{while } m < n \text{ do}
\]
\[
\begin{align*}
r &= P^T r; & \text{% Probability propagation} \\
m &= m + 1;
\end{align*}
\]
\[
\text{end}
\]
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 hand: 40 nodes, some 100 visits to each $\Rightarrow 4 \times 10^3$ iters.

- Idea: Unleash a large number of agents $K$

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

- Visits are now spread over time and space
  $\Rightarrow$ Converges “$K$ times faster” (depends agents’ initial distribution)
  $\Rightarrow$ But haven’t changed computational cost

Stoch. Systems Analysis
Ranking of nodes in graphs
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

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

Graph walk is a MC, then

$$\lim_{m \to \infty} \mathbb{P} \left[ A_m = i \right] = \lim_{m \to \infty} p_i(m) \text{ exists, and}$$

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

Probability propagation $\approx$ Unleashing infinite number of agents

Interpretation true for any MC
Distance to rank

- Initialize with uniform probability distribution \( p(0) = \frac{1}{J}1 \)
- Distance between \( p(n) \) and \( r \)

Distance is \( 10^{-2} \) in approximately 30 iterations, \( 10^{4} \) in 140 iterations

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**: Ten best ranks in 80 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

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

- Uses local information only

- 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 $p_i(n + 1) = \sum_{j \in n^{-1}(i)} p_j(n) / N_j$
  - Communicate rank estimate $p_i(n + 1)$ to outgoing neighbors $j \in n(i)$
  - Need only know 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 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 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
  - Non-iterative method, convergence not a problem
  - But computationally costly to obtain approximate solutions

- Probability propagation, matrix powers
  - Somewhat secure/robust. Info. shared with neighbors only
  - Implementation can be distributed
  - Convergence rate acceptable (orders of magnitude faster than RW)