
# Lab 7: PageRank

(Ranking the Terrorists)

October 29, 2014

Sarah Schwettmann

# Problems with the Predecessor

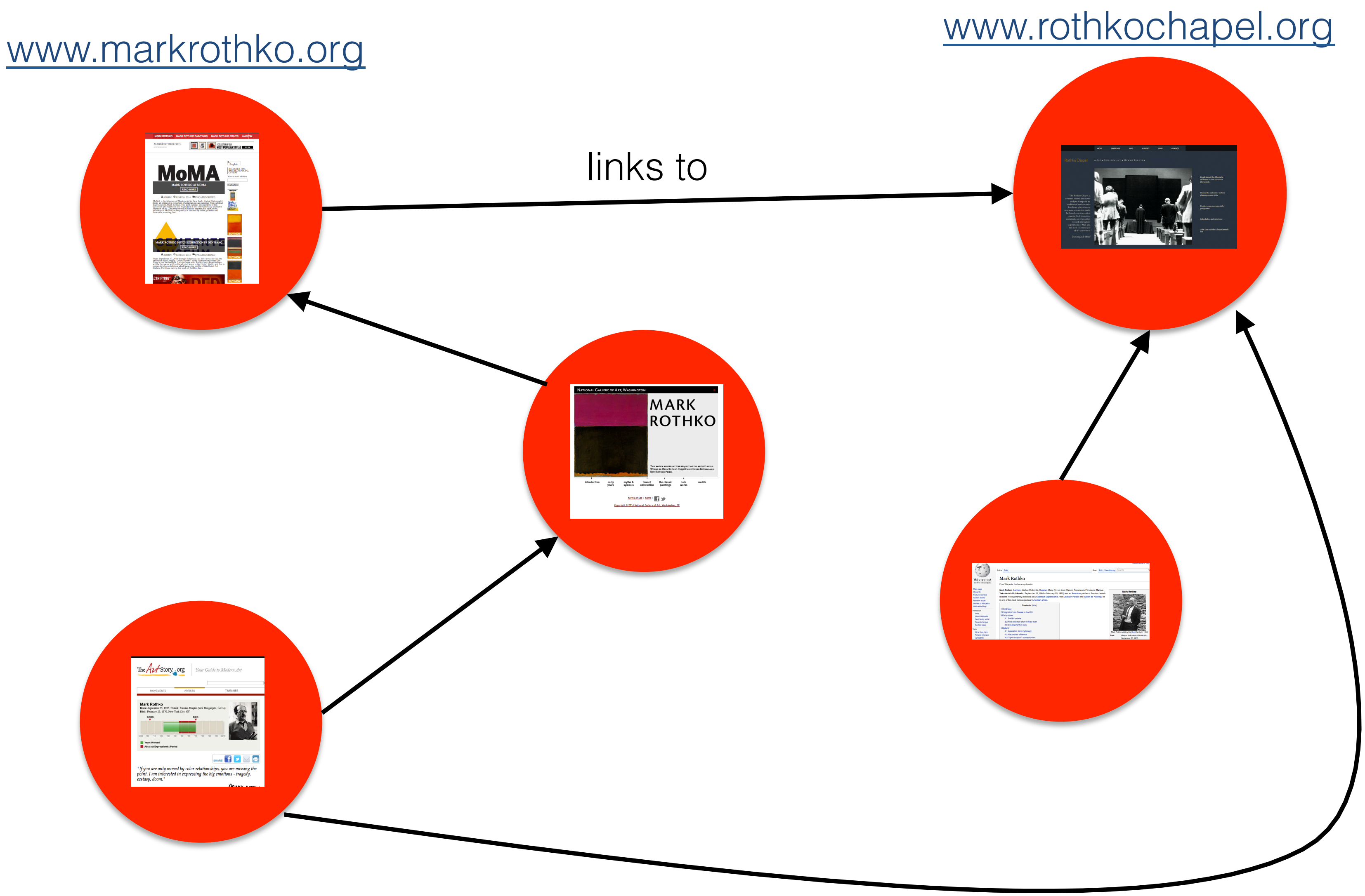
**Mid '90's**: "popularity"-based, search engines ranked results in terms of of how many times each link was clicked.

- Positive feedback loop for top-ranked sites
- Waiting time for a significant number of clicks

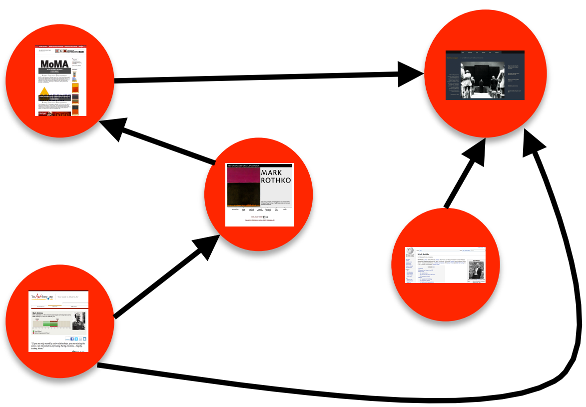
# Reformulation as a network problem

→ use the link structure of the internet itself to predict behavior

# Reformulation as a network problem



# Reformulation as a network problem

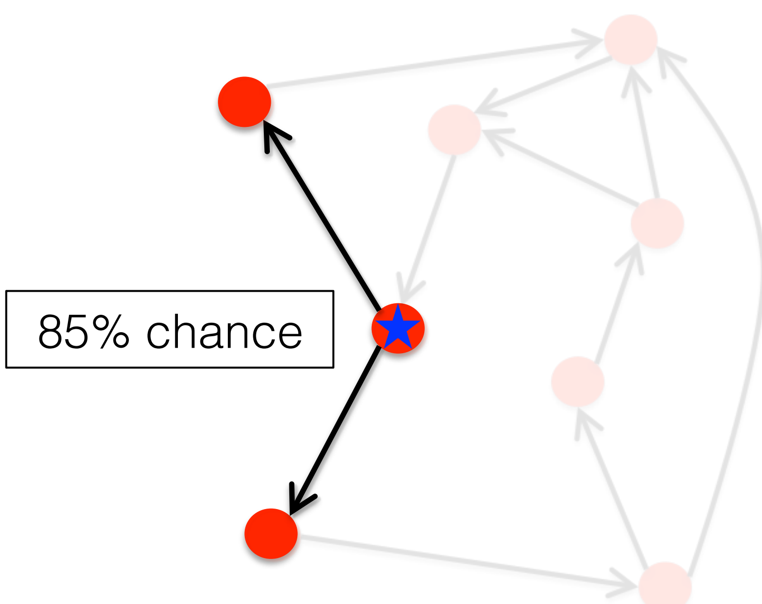



- Each webpage occupies a node on a graph
- Edges of graph represent links from one page to another
- The graph is ***directed***, where there is an arrow pointing from node *i* to node *j* if node *i* links to node *j*.
- A "small" site could be highly ranked if it has a lot of important sites linking to it

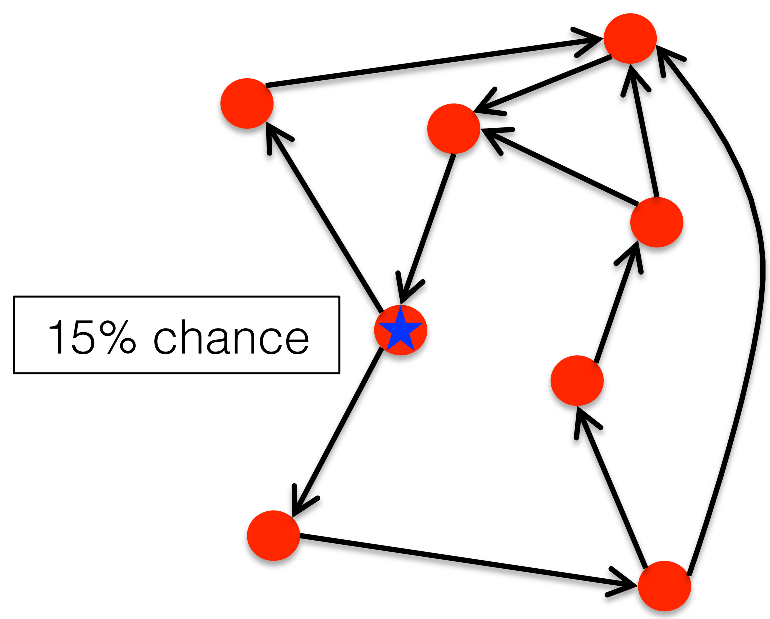
# PageRank Algorithm

Once the network is constructed, we can simulate the “walk” of a web user across the graph and determine how many visits each page will receive, *based on its position on this graph*. The number of visits gives us the page rank.

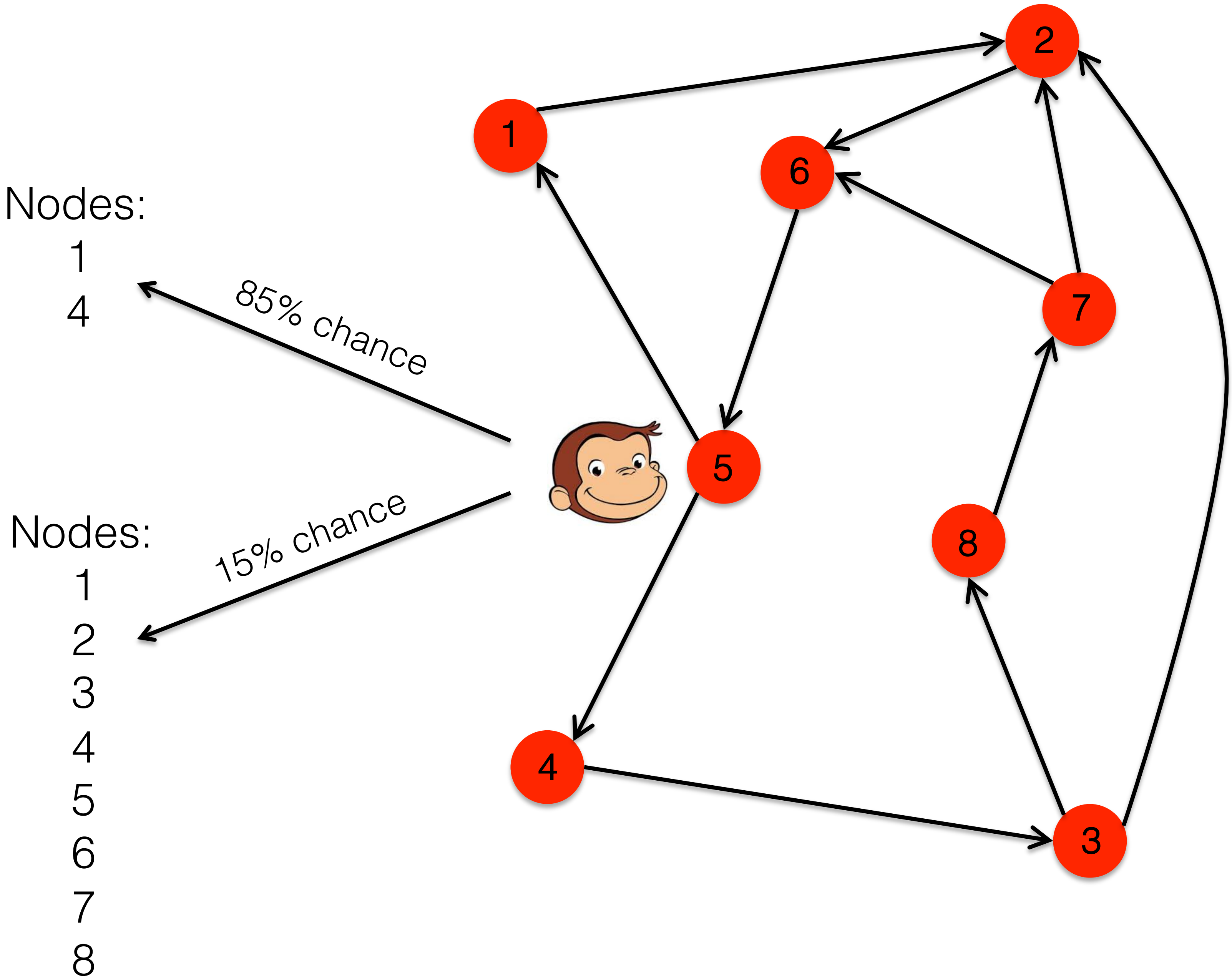
At any give node, there are two possibilities for behavior, each with a set probability of occurrence.




Move along directed edges to a node linked to current node




Move randomly to any node in network, including current node



Nodes:
1
4
85% chance
15% chance
Nodes:
1
2
3
4
5
6
7
8
1
2
3
4
5
6
7
8

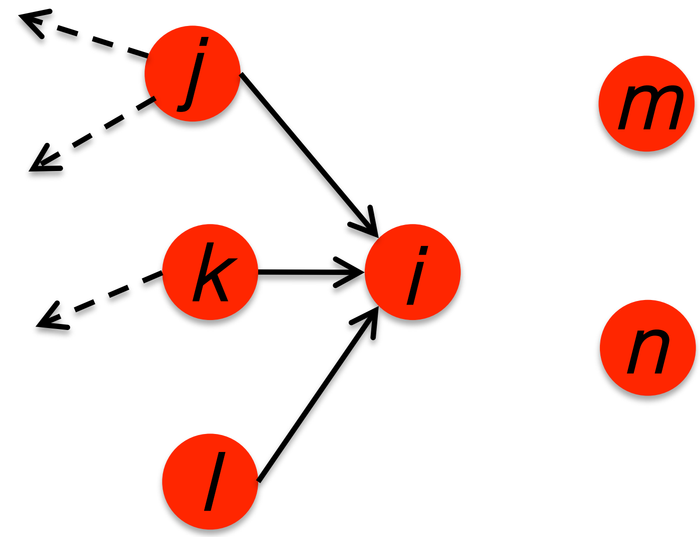
Running through the simulation of user navigation enough times for a significant ranking system would be very computationally intensive.

So…we are going to model the process using matrices to speed things up.

For each node, as its *rank*, we seek the proportion of visits to that node out of the network.

Let's start with the probability that we arrived at node *i* at a particular step.

We can either arrive at node *i* from a node that is connected to it (nodes j, k, l), or from any random node in the network.




Probabilities of each way to arrive at node i, given that you start at another node

j → i    $.85*(1/d_j) + .15/n$    ( $d_j$ = degree of node j = 3,  n = total nodes = 6 )

k → i    $.85*(1/d_k) + .15/n$

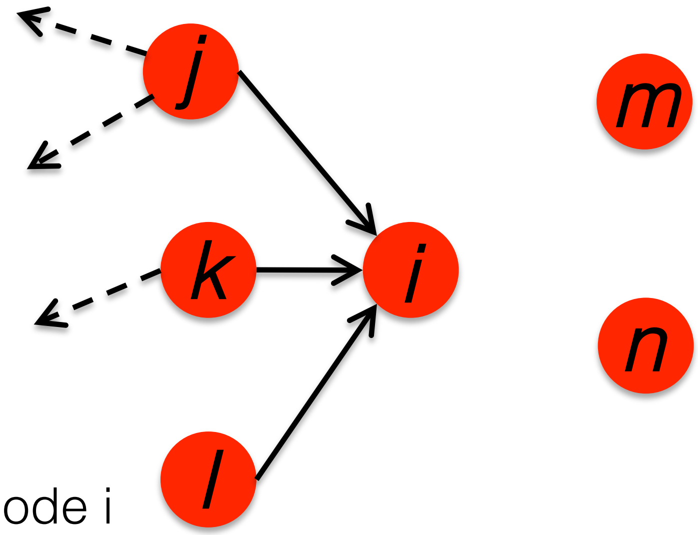
l → i    $.85*(1/d_l) + .15/n$

m → i    ?

n → i    ?

i → i    ?

We can either arrive at node *i* from a node that is connected to it (nodes j, k, l), or from any random node in the network.




Probabilities of each way to arrive at node i

$j \rightarrow i$ $\quad .85*(1/d_j) + .15/n$ $\quad$ ( $d_j$ = degree of node j = 3, n = total nodes = 6 )

$k \rightarrow i$ $\quad .85*(1/d_k) + .15/n$

$l \rightarrow i$ $\quad .85*(1/d_l) + .15/n$

$m \rightarrow i$ $\quad .15/n$

$n \rightarrow i$ $\quad .15/n$

$i \rightarrow i$ $\quad .15/n$

Importantly, we are doing this *recursively*, so the probability of going from $j \rightarrow i$ depends on the probability we were at j to begin with.

$$P(i) = P(j \rightarrow i)P(j) + P(k \rightarrow i)P(k) + P(l \rightarrow i)P(l) + P(m \rightarrow i)P(m) + P(n \rightarrow i)P(i) + P(i \rightarrow i)P(i)$$

# Constructing a Matrix Representation

$$P(j \rightarrow i)P(j) + P(k \rightarrow i)P(k) + P(l \rightarrow i)P(l) + P(m \rightarrow i)P(m) + P(n \rightarrow i)P(i) + P(i \rightarrow i)P(i) = P(i)$$

$$\begin{bmatrix} & & \cdots & & & \\ & & & & & \\ P(j \rightarrow i) & P(k \rightarrow i) & P(l \rightarrow i) & P(m \rightarrow i) & P(n \rightarrow i) & P(i \rightarrow i) \end{bmatrix} \begin{bmatrix} P(j) \\ P(k) \\ P(l) \\ P(m) \\ P(n) \\ P(i) \end{bmatrix} = \begin{bmatrix} P(j) \\ P(k) \\ P(l) \\ P(m) \\ P(n) \\ P(i) \end{bmatrix}$$

# Constructing a Matrix Representation

$$\mathbf{R}$$

$$\begin{array}{cc} & \begin{bmatrix} & & & \cdots & & \\ & & & & & \\ P(j\to i) & P(k\to i) & P(l\to i) & P(m\to i) & P(n\to i) & P(i\to i) \end{bmatrix} \\ \text{row } i & \\ & \text{col. } j \end{array}$$

Construct a matrix R s.t. $\mathbf{R}_{ij} = \begin{cases} \frac{0.85}{d_j} + \frac{0.15}{n} & \text{if } j \to i \\ \frac{0.15}{n} & \text{otherwise} \end{cases}$

# Solving for "rank" p of each node

$$P(j \rightarrow i)P(j) + P(k \rightarrow i)P(k) + P(l \rightarrow i)P(l) + P(m \rightarrow i)P(m) + P(n \rightarrow i)P(i) + P(i \rightarrow i)P(i) = P(i)$$

$$\underbrace{\begin{bmatrix} & & \cdots & & & \\ P(j \rightarrow i) & P(k \rightarrow i) & P(l \rightarrow i) & P(m \rightarrow i) & P(n \rightarrow i) & P(i \rightarrow i) \end{bmatrix}}_{\mathbf{R}} \underbrace{\begin{bmatrix} P(j) \\ P(k) \\ P(l) \\ P(m) \\ P(n) \\ P(i) \end{bmatrix}}_{\mathbf{p}} = \underbrace{\begin{bmatrix} P(j) \\ P(k) \\ P(l) \\ P(m) \\ P(n) \\ P(i) \end{bmatrix}}_{\mathbf{p}}$$

$$\mathbf{Rp} = \mathbf{p}$$

$$\mathbf{Rp} - \mathbf{p} = 0$$

$$\mathbf{Rp} - \mathbf{Ip} = 0 \rightarrow \boxed{(\mathbf{R} - \mathbf{I})\mathbf{p} = 0}$$

We know R and I, so we can solve for p using Gaussian elimination!

# Solving for p using gauss

$$\boxed{(\mathbf{R} - \mathbf{I})\mathbf{p} = \mathbf{0}}$$

Problem: there is no unique solution (e.g. we could have **p** = 0).
Solution: Augment (R-I) with a row of 1's at the bottom.

$$\underbrace{\left[\begin{array}{ccccccccc} & & & & \mathbf{R\text{-}I} & & & & \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & & \dots & 1 \end{array}\right]}_{\mathbf{A}} \; \mathbf{p} \;=\; \underbrace{\left[\begin{array}{c} \mathbf{0} \\ \hline 1 \end{array}\right]}_{\mathbf{z}}$$

Then run p = gauss(A,z) to return the vector of ranks, p

This presentation is available at [www.cogconfluence.com](http://www.cogconfluence.com) (under the CAAM 210, Fall '14 tab)

# Project

*Four functions:*

pagerankdriver

→ reads the .txt terrorist file into MATLAB

→ calls pagerankmatrix on the edgelist E to create R

→ calls gauss on "doctored" R to solve for p

→ uses pagerank p to create a biograph with nodes colored/shaped based on rank

[p] = gauss(A,z)

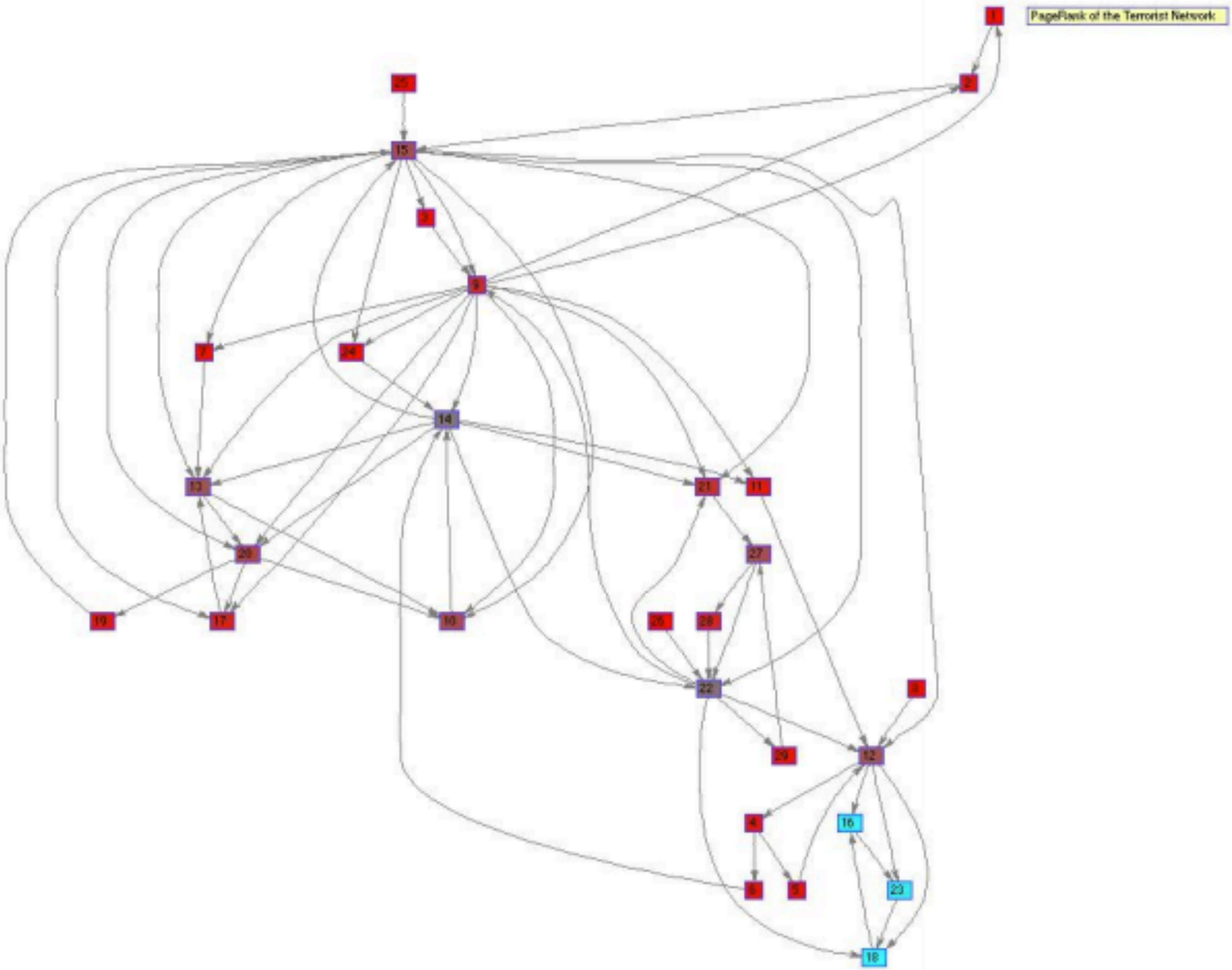
→ your own gaussian elimination code

[p] = trisolve(A,z)

→ back-substitution to complete gauss, provided in notes

[R] = pagerankmatrix(E)

→ your own code for creating R based on probabilities


PageRank of the Terrorist Network