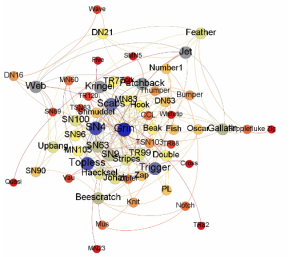
Network Centrality

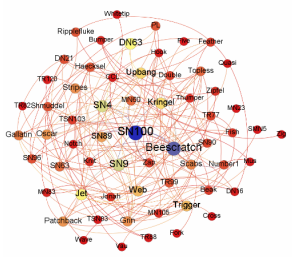
Degree Centrality

A



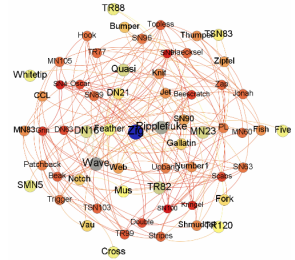
Degree Centrality

B



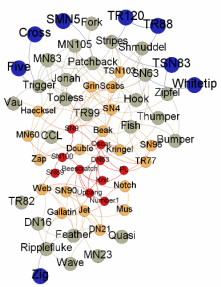
Betweenness Centrality

C



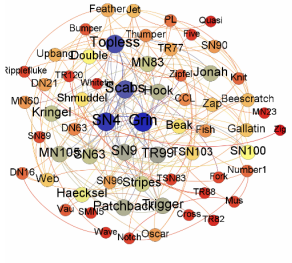
Closeness Centrality

D

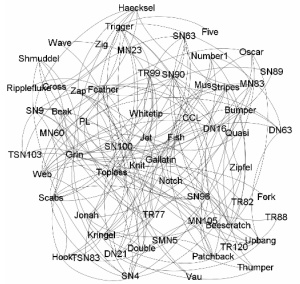


Eccentricity Centrality

E



Eigenvector Centrality



Dolphin's Social Network

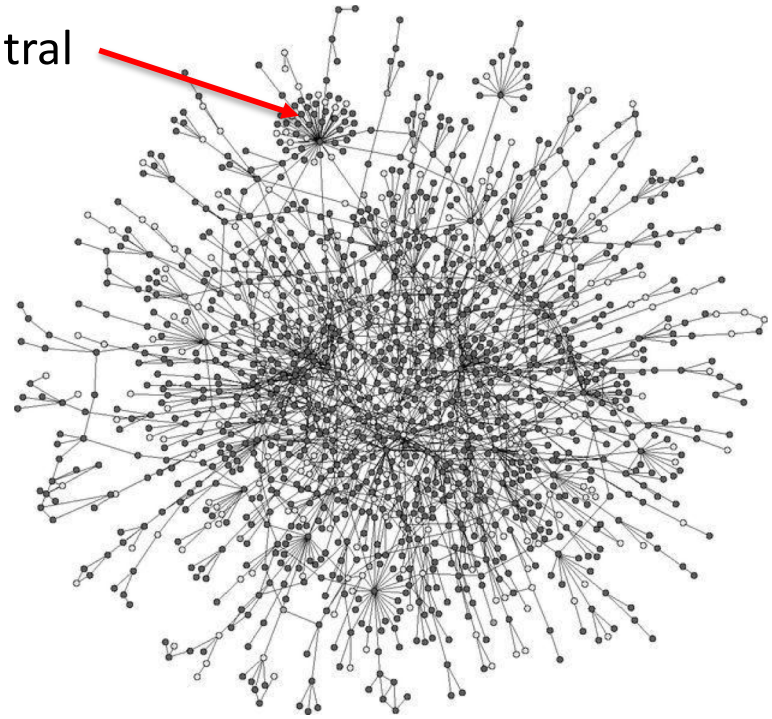
Degree Centrality

How can the nodes of a network be ranked according to their importance?

Because there is no right answer, several centrality measures have been developed and used. Google developed a good one, which is why google searches usually find the best web sites on the first page.

The simplest centrality measure just ranks the nodes according to their degree (or in-degree or out-degree for directed networks). This is degree centrality.

High degree, very central



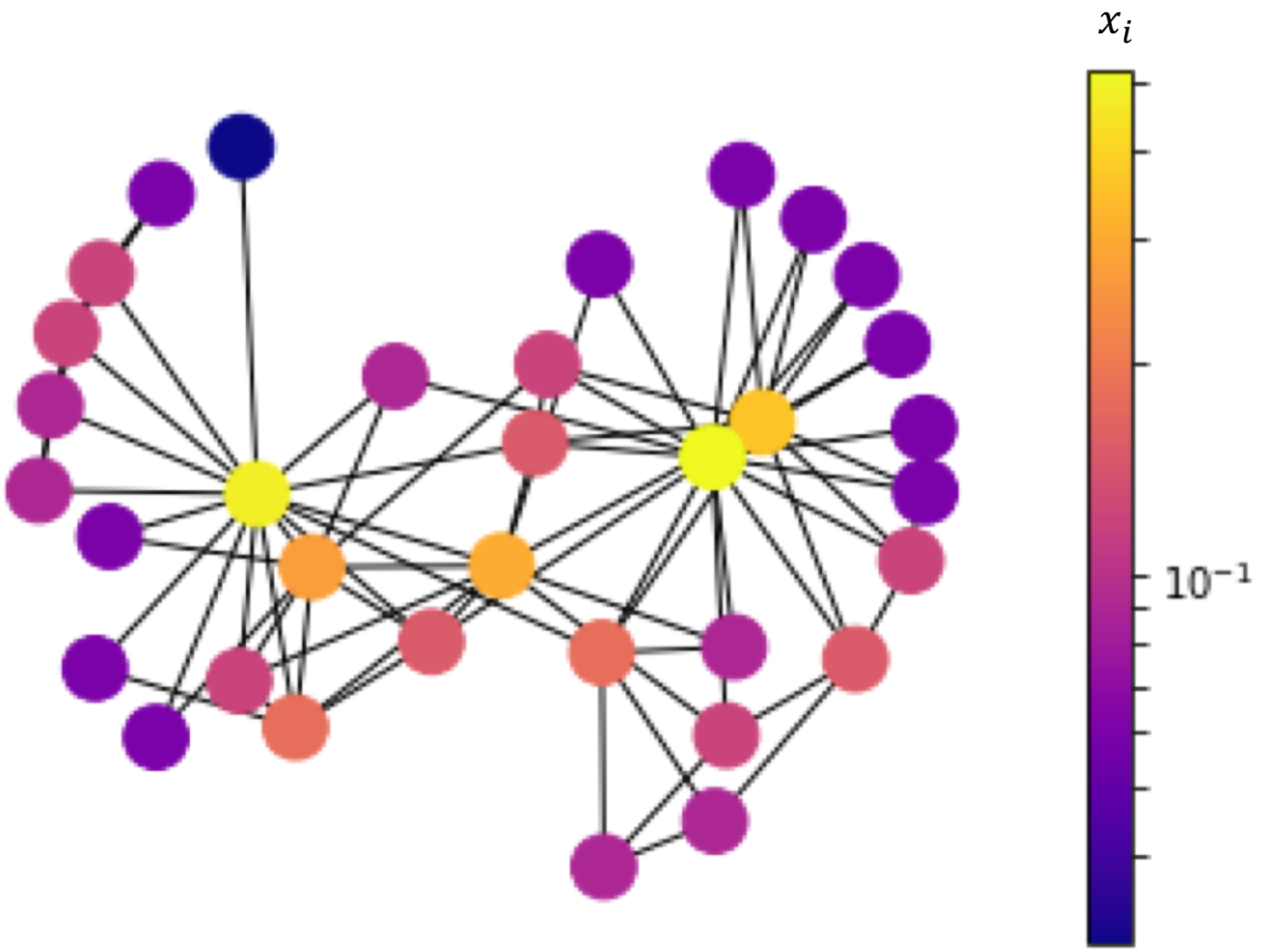
Degree Centrality

Let $\vec{x}$ be the n-dimensional degree centrality vector for an undirected network. Then

$$x_i = \sum_{j=1}^{n} A_{ij} \qquad \text{or} \qquad x_i = \frac{1}{n}\sum_{j=1}^{n} A_{ij}$$

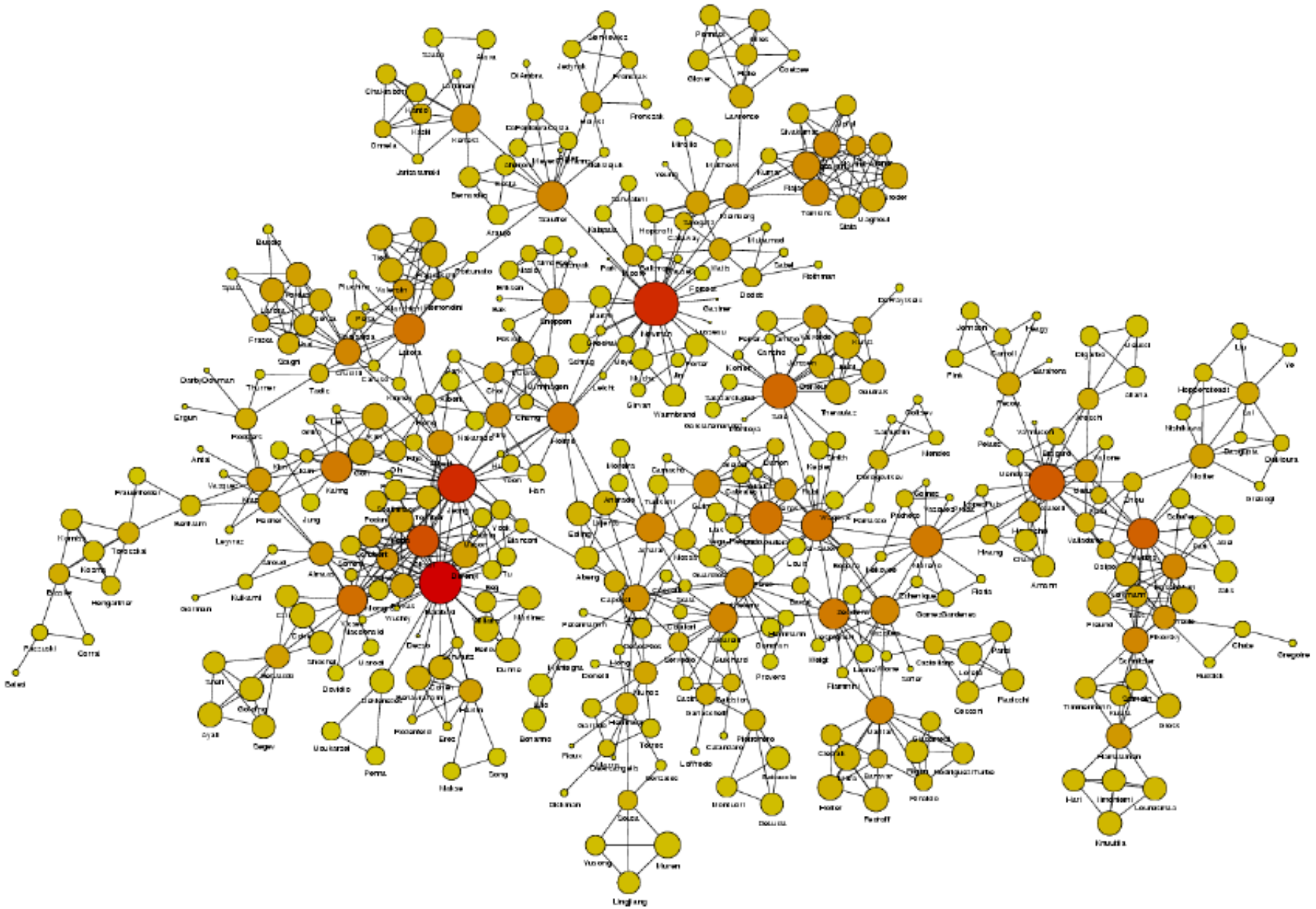
The actual values don't matter, only the ranking matters.

Degree Centrality



Color coded according to degree centrality

Eigenvector and Katz Centrality



It's Not How Many You Know, But Who You Know

In a social network, you may have more influence if you know a few influential people than if you know many not-so-influential people.

Idea: when computing centrality of a node, take into consideration the centrality of the nodes it is connected to.

$$x_i = \alpha \sum_{j=1}^{n} A_{ij} x_j$$

Eigenvector centrality

where α is a positive scaling factor.

In vector form: $\frac{1}{\alpha}\vec{x} = A\vec{x}$ so $\vec{x}$ is an eigenvector of *A*

Which Eigenvector?

There are *n* eigenvectors. Which is the right one?

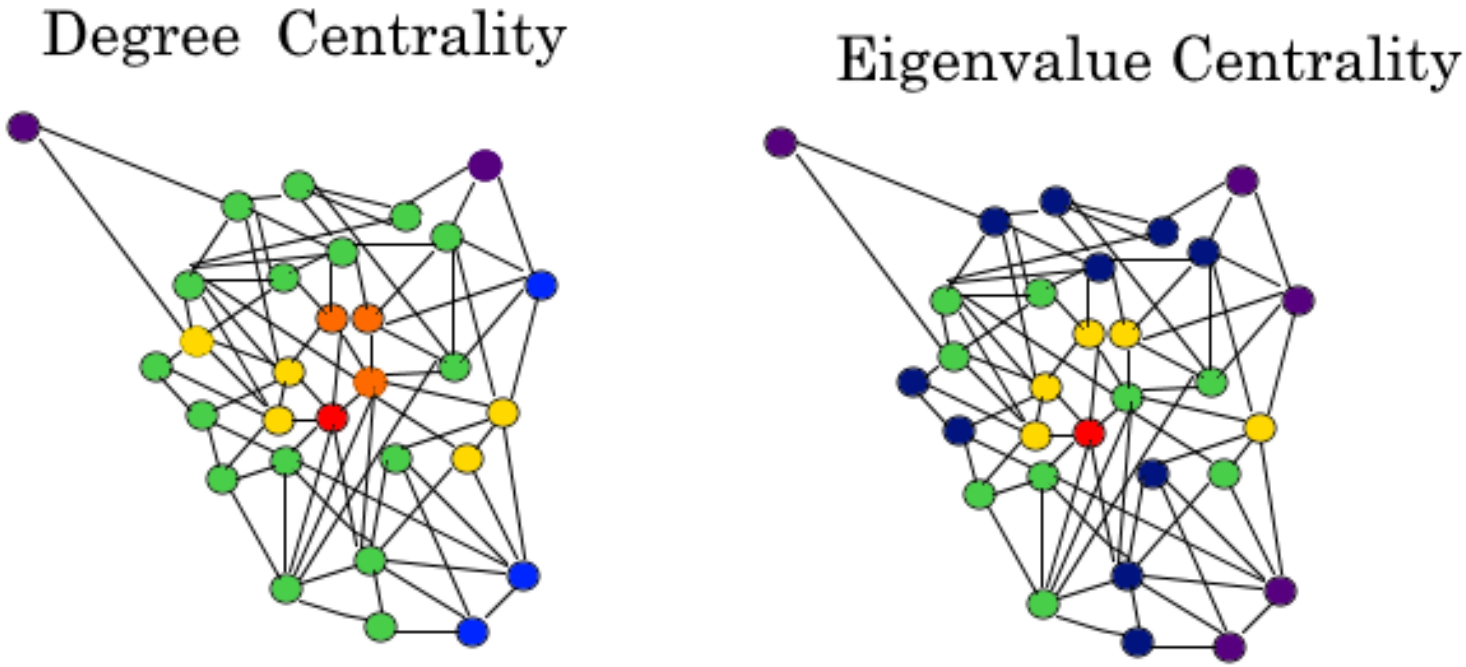
Pick one with all non-negative elements, since this is a ranking.

Perron-Frobenius Theorem: If a square matrix has non-negative elements, then only the leading eigenvector has elements with all elements of the same sign.

Pick this eigenvector, and multiply by -1 if necessary to make all elements positive.

Then $\lambda_1 = \frac{1}{\alpha}$ and $\vec{x}$ is the corresponding leading eigenvector of *A*.
This is the eigenvector centrality.

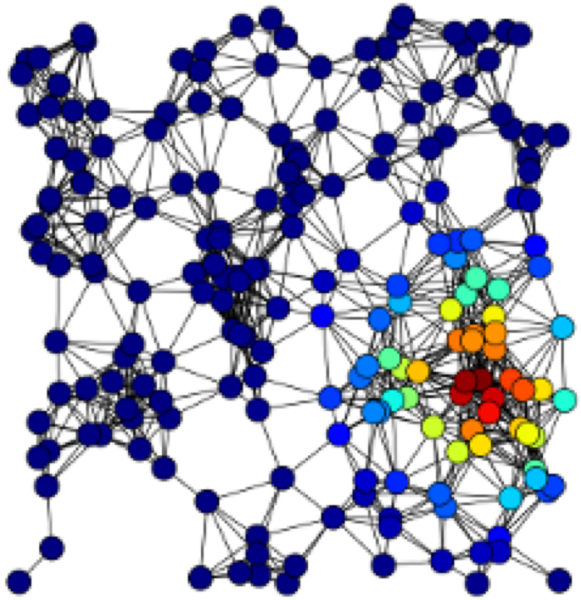
Degree vs. Eigenvector Centrality



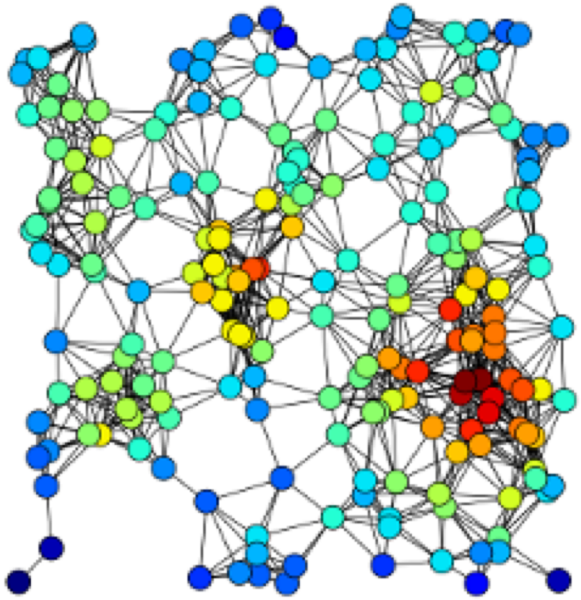
Some nodes ranked similarly (red is highest, purple lowest), but many differences. With eigenvector centrality, more of a distinction between nodes with high centrality and nodes with low centrality.

Degree vs. Eigenvector Centrality

Eigenvector centrality



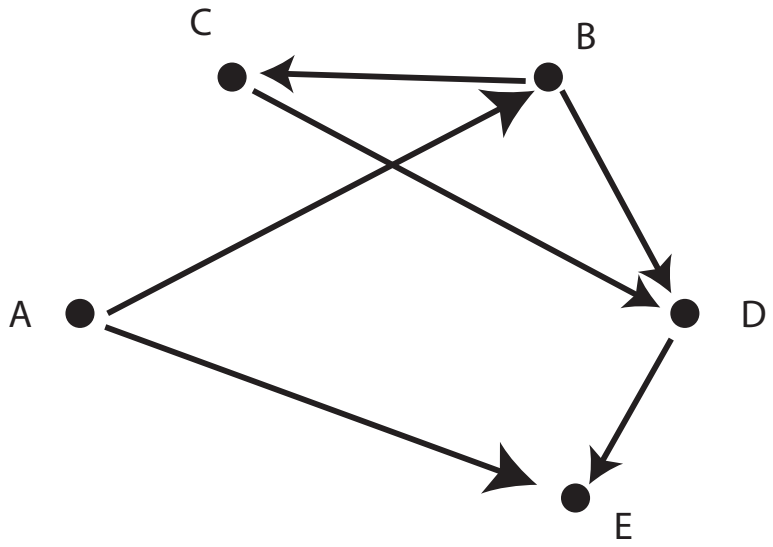
Degree centrality



Some nodes ranked similarly (red is highest, dark purple lowest), but many differences. With eigenvector centrality, more of a distinction between nodes with high centrality and nodes with low centrality.

Potential Problem with Directed Graphs

Calculate centrality here in terms of incoming edges



What is the eigenvector centrality for A? $x_A = 0$

What about B? $x_B = 0$

What about C, D, E? $x_{C,D,E} = 0$

Katz Centrality

Workaround: give every node some centrality

$$x_i = \alpha \sum_{j=1}^{n} A_{ji} x_j + \beta = \alpha \sum_{j=1}^{n} A_{ij}^T x_j + \beta$$

where $\alpha > 0$ is weight put on neighbors and $\beta > 0$ is the free centrality. Notice that the sum is down a column of *A* since it is using information about incoming edges. For convenience, define the matrix $K \equiv A^T$.

Can just set the free centrality to 1, then

$$x_i = \alpha \sum_{j=1}^{n} K_{ij} x_j + 1$$

In vector form, $\vec{x} = \alpha K \vec{x} + \vec{1}$

$\Longrightarrow$ $(I - \alpha K)\vec{x} = \vec{1}$

$\Longrightarrow$ $\boxed{\vec{x} = (I - \alpha K)^{-1}\vec{1}}$

Katz Centrality

Katz Centrality: Choosing α

How do we choose α?

Make it large to emphasize input from neighbors, but can't make it too large.

As $\alpha \to 0$, the centrality becomes dominated by the free centrality term β, so all centrality terms tend to equal values. As α is increased, the centrality of some nodes grow much larger than others, and will eventually diverge to ∞ as $\alpha \to \infty$. So there must be some upper bound on α. What is it?

Katz Centrality Convergence Condition

We must make sure the matrix is invertible. The matrix becomes singular when

$$\det(I - \alpha K) = 0$$

$$\Longrightarrow \qquad \det\left(K - \frac{1}{\alpha} I\right) = 0$$

This is just the characteristic equation for the eigenvalues of $K = A^T$. That is, eigenvalues satisfy

$$\det(K - \lambda I) = 0$$

So $\lambda = \frac{1}{\alpha}$ $\Longrightarrow$ $\alpha = \frac{1}{\lambda}$

To keep the matrix non-singular requires $\alpha < \frac{1}{\lambda}$ Which eigenvalue?

All of them! So pick the one that is most restrictive $\boxed{\alpha < \frac{1}{\lambda_1}}$

If α does not satisfy this condition the centrality will be meaningless

Iterative Calculation of Katz Centrality

Problem: Katz centrality requires inversion of an $n \times n$ matrix. This requires n^3 algebraic operations. For a 1000-node network, this is 1 billion operations.

Good news: There is a much better way, using an iterative algorithm. Recall that the Katz centrality vector satisfies

$$\vec{x} = \alpha K \vec{x} + \vec{1}$$

Convert this to the following recursion relation:

$$\vec{x}_{k+1} = \alpha K \vec{x}_k + \vec{1}$$

Iterate for k=1, 2, 3, … until a stopping criterian is met, like

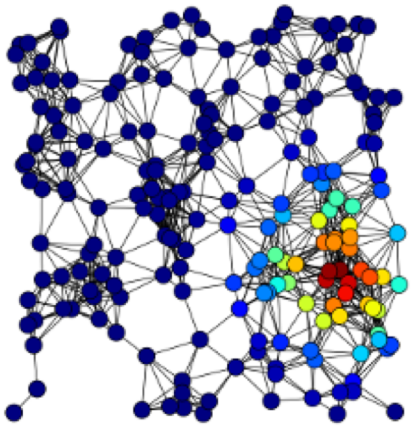
$$\|\vec{x}_{k+1} - \vec{x}_k\| < \epsilon \qquad \text{for some small } \epsilon > 0.$$

As long as α satisfies the convergence condition, the iterates will converge to the solution to the equation at the top. That is, iterates will converge to the Katz centrality.

Each iteration requires matrix-vector multiplication, cheaper than matrix inversion. Can often stop after just a few iterations.

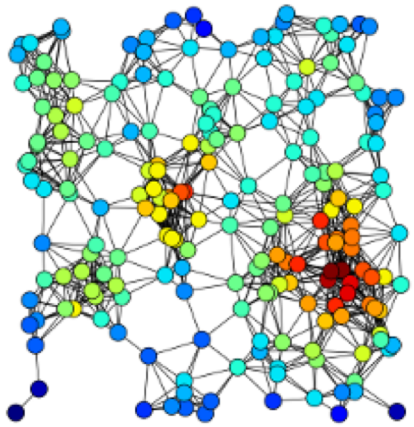
Degree, Eigenvector, and Katz Centrality

Eigenvector centrality



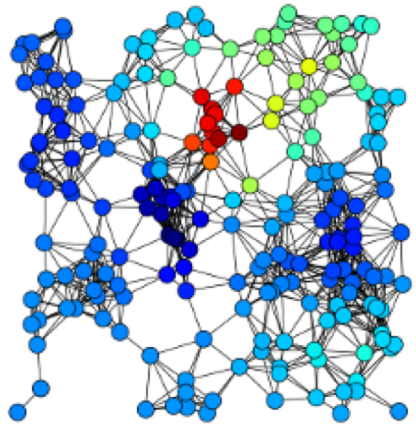
C

Degree centrality



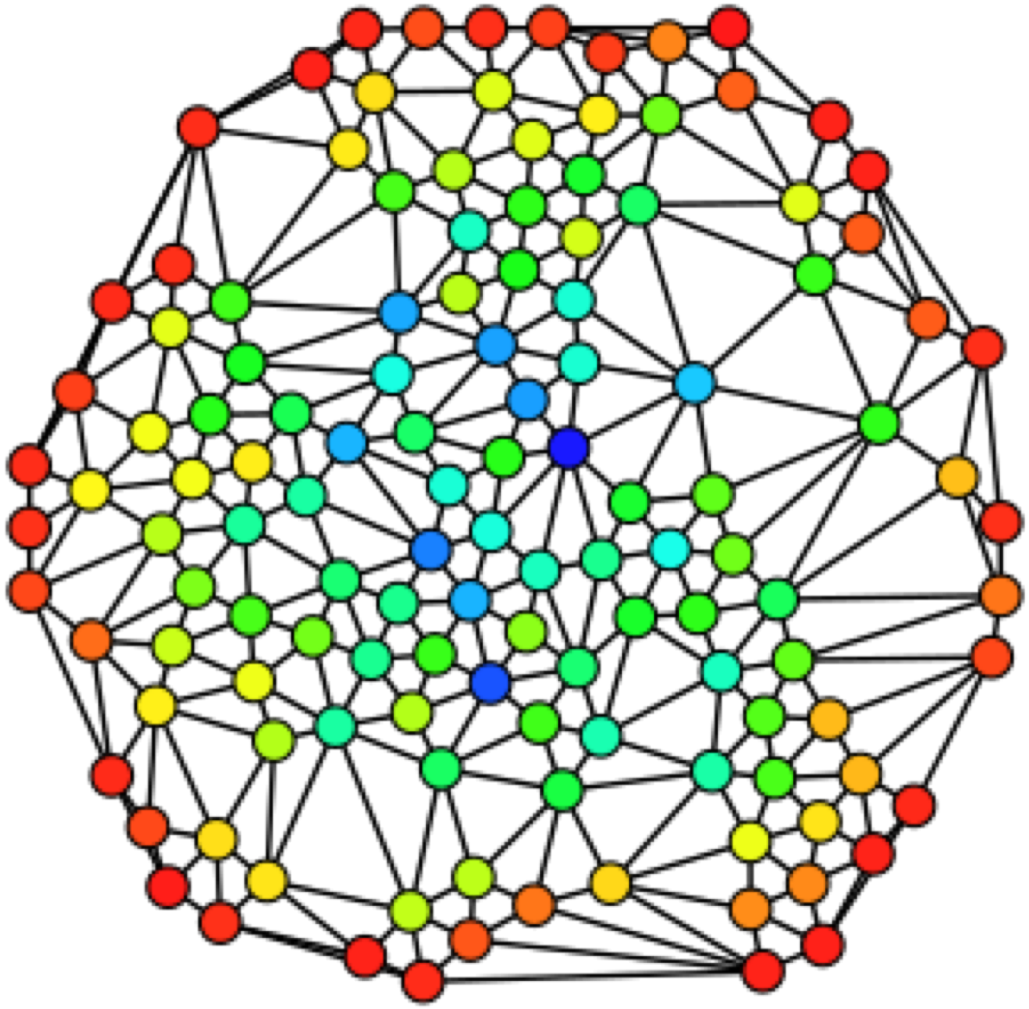
D

Katz centrality



F

Closeness, Harmonic, and Betweenness Centrality



Closeness Centrality

Basic idea: Nodes closest to all other nodes in the network (in terms of path length) are the most central.

Let p_{ij} be the length of a shortest (geodesic) path between nodes *i* and *j*. To compute the closeness centrality for node *i*, compute the mean geodesic distance between *i* and all other nodes in the network.

$$l_i = \frac{1}{n-1}\sum_{j=1}^{n} p_{ij}$$ mean geodesic path length

Now invert this, so that nodes with large l_i have small centrality.

$$\boxed{x_i = \frac{1}{l_i}}$$

Closeness centrality

Harmonic Centrality

In closeness centrality there is summation of geodesic path lengths, followed by inversion. In harmonic centrality these two operations are reversed.

Let p_{ij} be the length of a shortest (geodesic) path between nodes *i* and *j*. Then

$$x_i = \frac{1}{n-1}\sum_{j=1}^{n}\frac{1}{p_{ij}}$$

Harmonic centrality

In the summation, if $p_{ij} = 0$ this means that nodes *i* and *j* are not connected. Replace $\frac{1}{p_{ij}}$ with 0.

It gets its name from the harmonic sum, $\sum_{n=1}^{N}\frac{1}{n}$.

Problems with Closeness and Harmonic Centrality

They are hard to compute: Finding geodesic paths is not easy in a large network. It is very computationally expensive to apply the breadth first algorithm *n* times when *n* is large.

They do not effectively separate nodes: The span of centrality values over the network is typically small, so not much difference in value between hubs and non-hubs.

Betweenness Centrality

Basic idea: The most central nodes are those on geodesic paths of a lot of other nodes.

Let $n_i^{st} = \begin{cases} 1 & \text{If node } i \text{ is on a geodesic path between nodes } s \text{ and } t \\ 0 & \text{Otherwise} \end{cases}$

Then

$$x_i = \sum_{s,t=1}^{n} n_i^{st}$$

But what if there are two geodesic paths between nodes *s* and *t* and node *i* is only on one them? Then it should get half credit. More generally, let g^{st} be the number of geodesic paths between nodes *s* and *t*. Then

$$x_i = \sum_{s,t=1}^{n} \frac{n_i^{st}}{g^{st}}$$

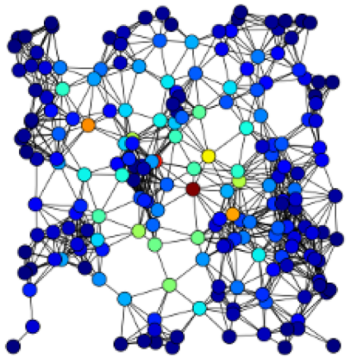
or normalized

$$x_i = \frac{1}{n^2} \sum_{s,t=1}^{n} \frac{n_i^{st}}{g^{st}}$$

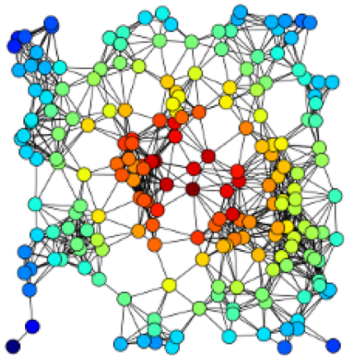
Betweenness centrality

Centrality Comparison

Betweenness



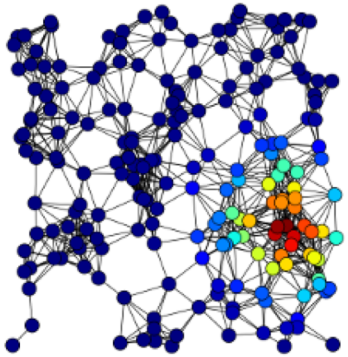
A



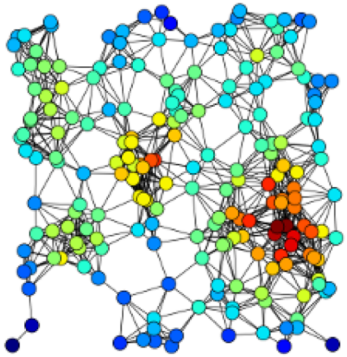
B

Closeness

Eigenvector



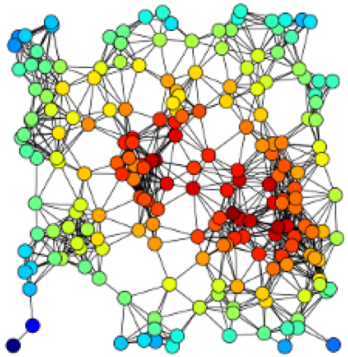
C



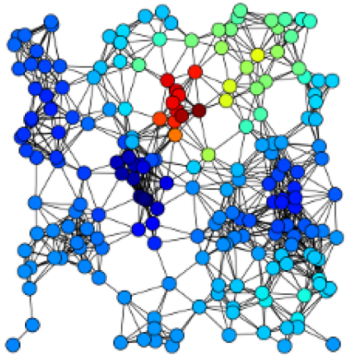
D

Degree

Harmonic



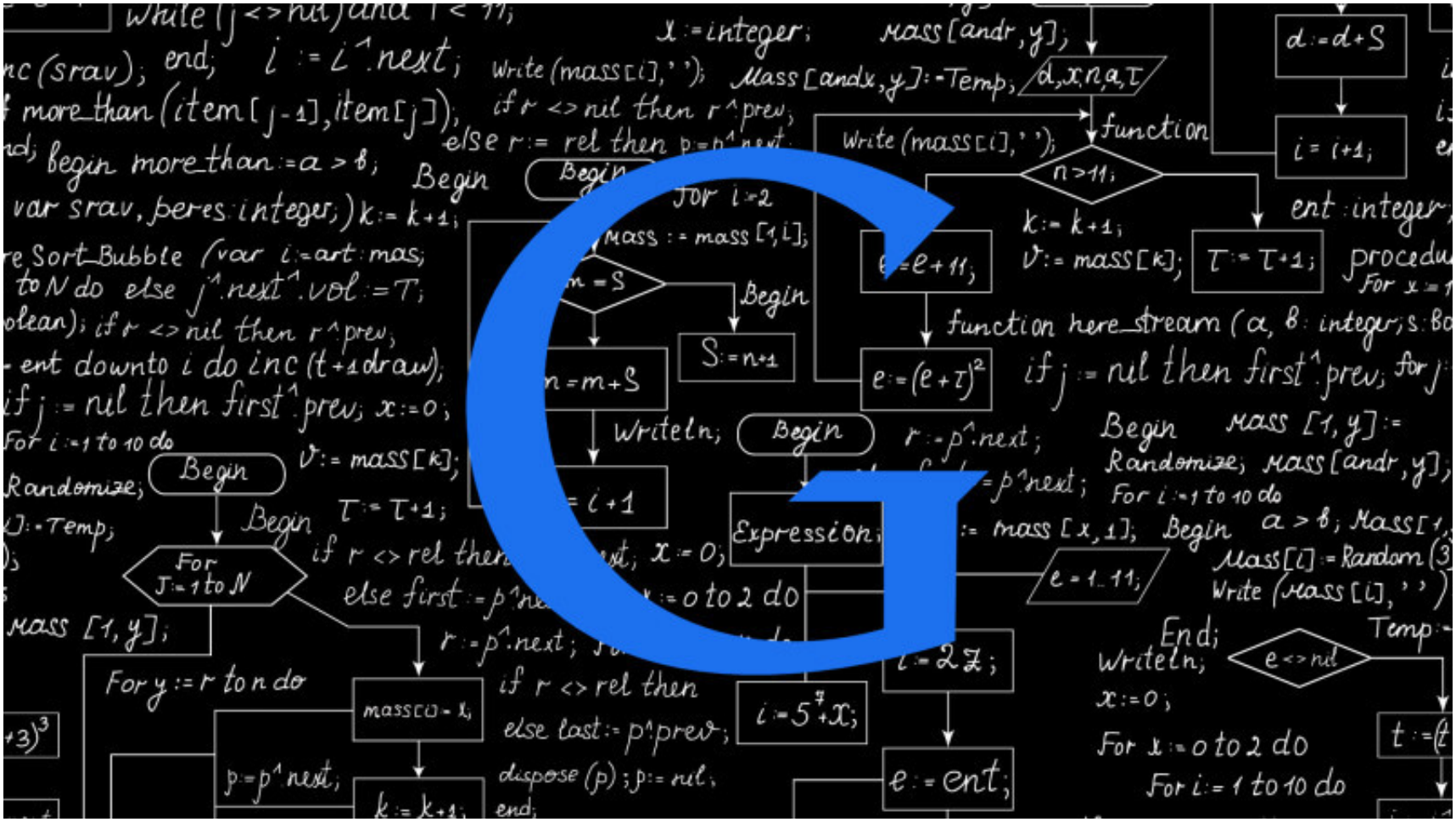
E



F

Katz

Hubs, Authorities, and PageRank



Being Google is Not Easy

Three major problems in information retrieval:

Synonymy: multiple ways of saying the same thing

Ex: a search for recipes involving green onions could miss recipes involving scallions

Polysemy: multiple meanings for the same term

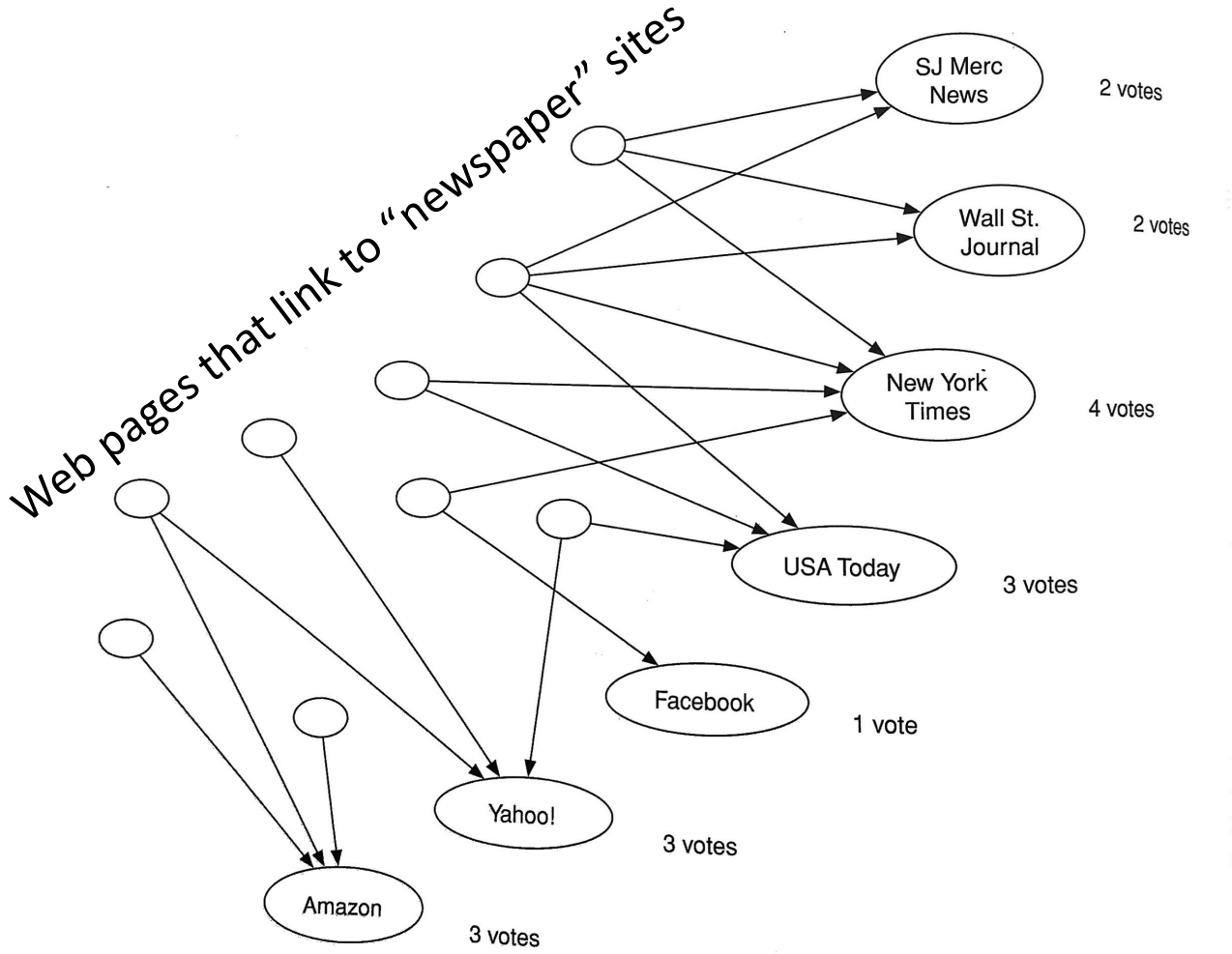
Ex: a search for jaguars, the animal, could turn up web pages on cars or football teams

Abundance: there are a lot of web pages out there, most are irrelevant to a user of the search engine. How does Google pick the ones the user might want?

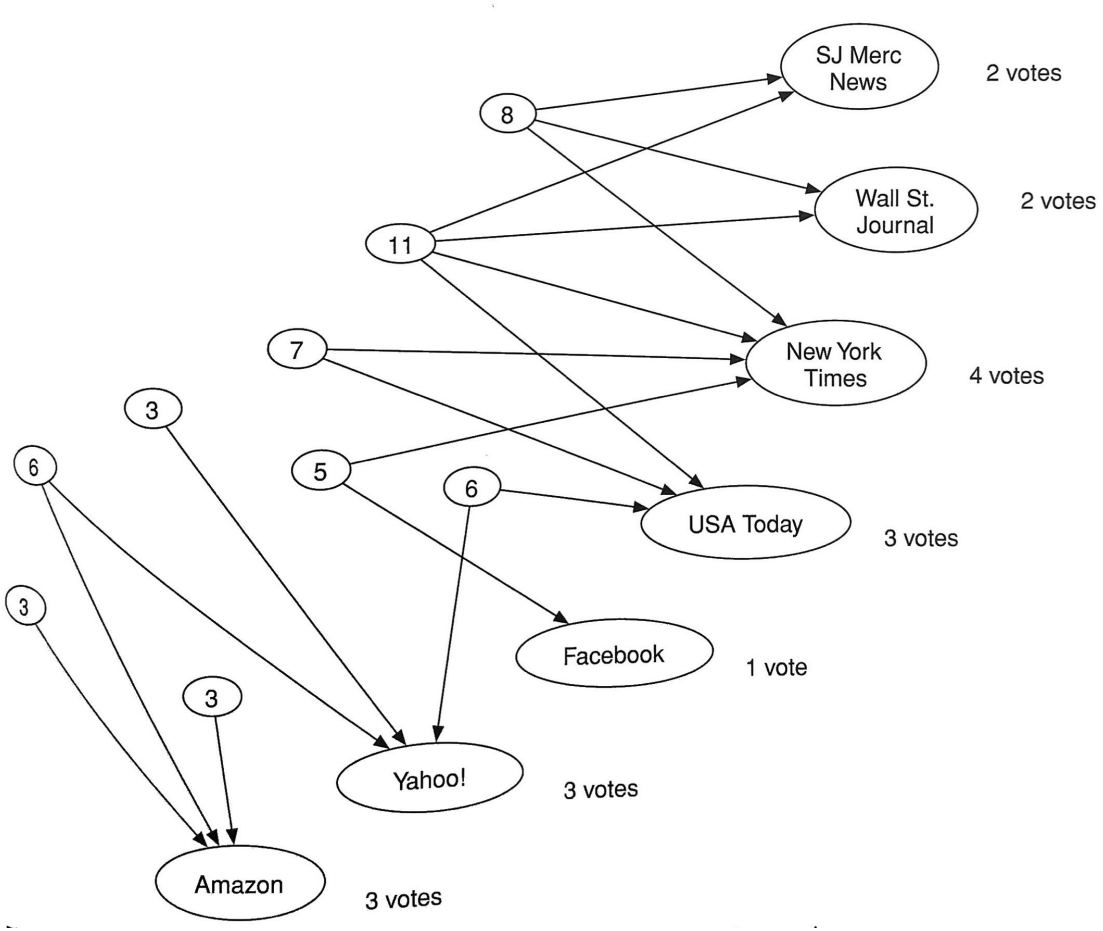
Example: Searching for a Good Newspaper

Consider the task of finding a good newspaper on the web. A search will turn up local newspapers and prominent newspapers (which is what you want), but also things links to things like Yahoo, Facebook, Amazon, etc. since these sites have a lot of other sites relating to newspaper pointing to them. That is, they have a high in-degree from relevant nodes. How does Google find the most relevant sites to list on the first page of the search?

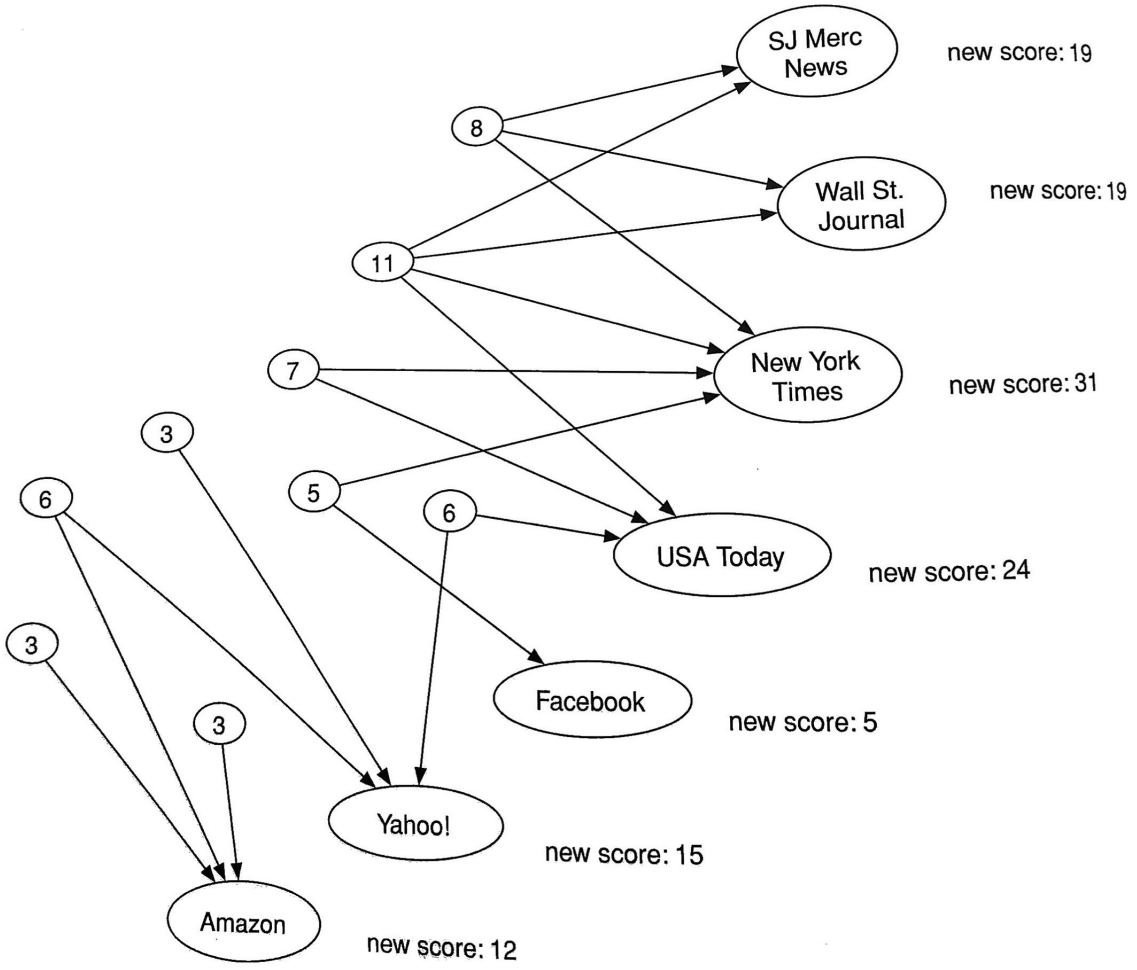
Iteration ½: Calculate "Votes" for Newspaper sites



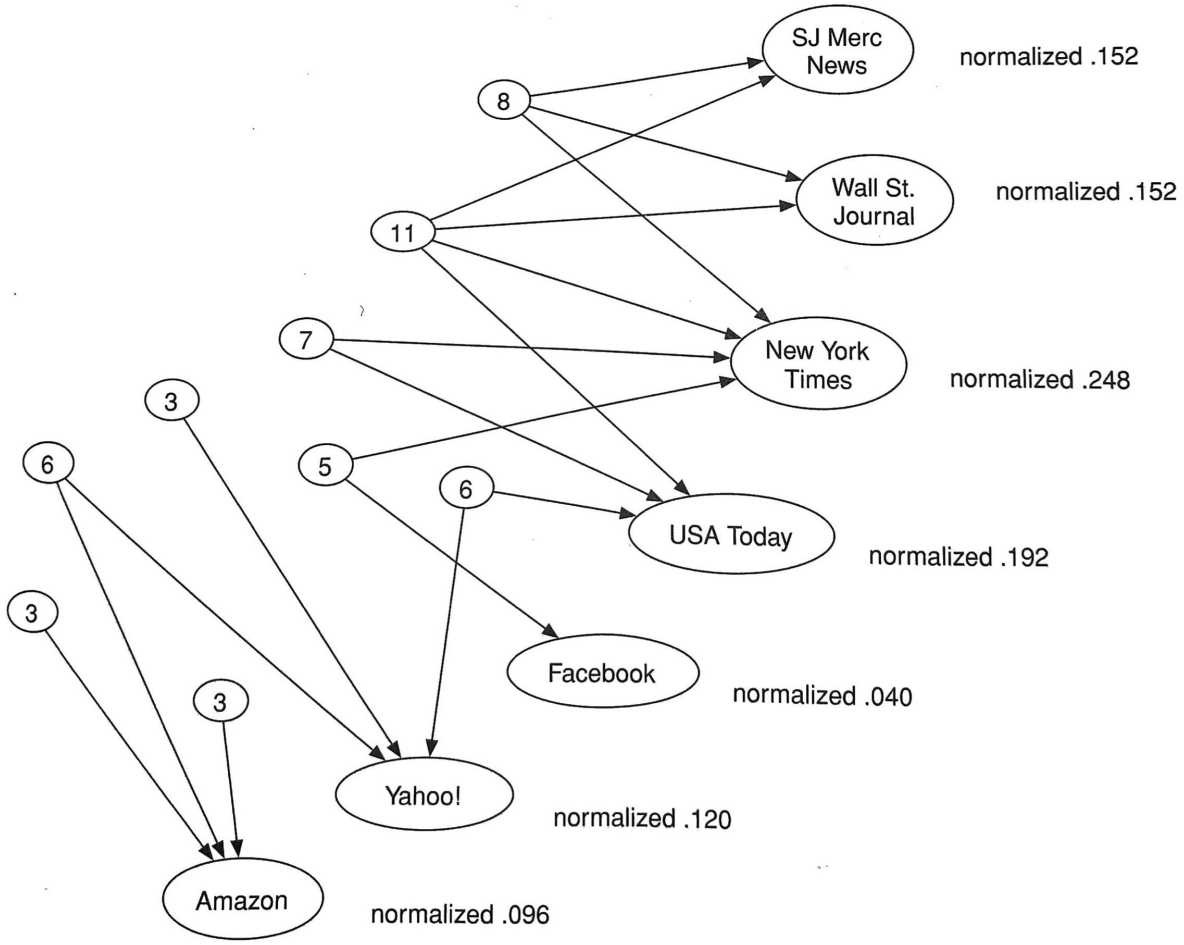
Iteration 1: Score the Sites That Casted Votes According to How Good Their Votes Were



Repeat: Update the Newspaper Scores, Using the New Scores of the Voting Sites

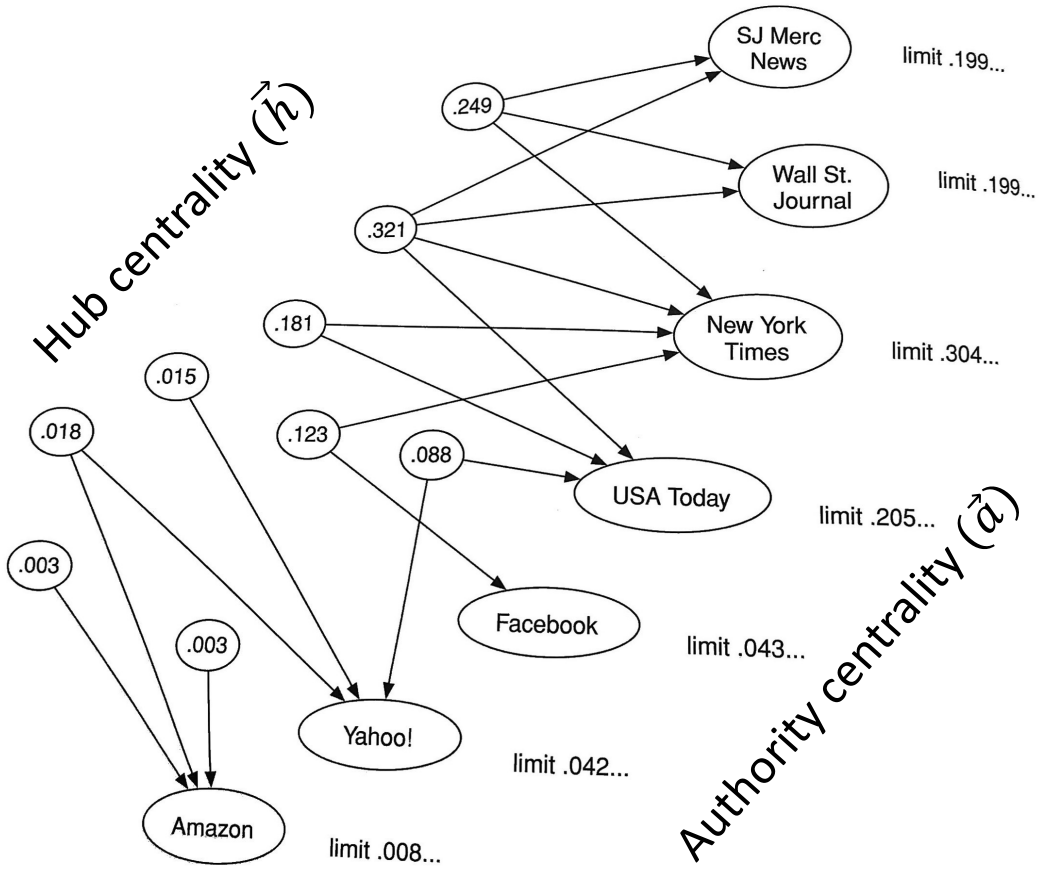


Normalize: So That Scores Don't Grow Without Bound, Normalize Newspaper Scores By Their Sum



Keep Iterating Until Satisfied

Note that all scores have been normalized (hubs and authorities separately).



This shows the limiting ranking or scores. Hubs are web sites that have high ranking for selecting good newspaper sites. Authorities are the web sites that have high ranking as determined by good endorsements.

Hubs and Authorities as Matrix Iteration

To update authority node *j*, sum all the hub nodes that project to *j*. This is a weighted sum down column *j* of *A*, or row *j* of its transpose.

Authority update: $\vec{a}_k = A^T \vec{h}_{k-1}$

To update hub node *j*, sum all the authority nodes that node *j* projects to. This is a weighted sum along row *j* of *A*.

Hub update: $\vec{h}_k = A\vec{a}_k$

or $\vec{a}_k = (A^T A)\vec{a}_{k-1}$

$\vec{h}_k = (AA^T)\vec{h}_{k-1}$

After *k* updates,

$\vec{a}_k = (A^T A)^k \vec{a}_0$

$\vec{h}_k = (AA^T)^k \vec{h}_0$

Spectral Solution for Hub/Authority Centrality

Either solution can be written in terms of eigenvalues/eigenvectors. For authorities,

Spectral Solution: $\vec{a}_k = c_1\lambda_1^k\vec{v}_1 + \cdots + c_n\lambda_n^k\vec{v}_n$

where λ_j, $\vec{v}_j$ are eigenvalues/eigenvectors of the matrix A^TA.

Since this matrix has all non-negative elements, the Perron-Frobenius Theorem tells us that there is exactly one eigenvector with all positive elements, and this is the eigenvector associated with the leading eigenvalue, λ_1. Since the limiting hub vector has all non-negative elements,

Equilibrium authority centrality: $\boxed{A^TA\vec{a} = \lambda_1\vec{a}}$

Spectral Solution for Hub/Authority Centrality

Similar arguments hold for the hub centrality. But what relationship is there between the leading eigenvector of AA^T and that of A^TA?

Multiply the equilibrium authority centrality by *A*:

$$AA^T(A\vec{a}) = \lambda_1(A\vec{a})$$

So the leading eigenvalue (or any eigenvalue) of AA^T is the same as for A^TA. Also, the associated eigenvectors are just *A* times those for A^TA.

Equilibrium hub centrality: $\boxed{\vec{h} = A\vec{a}}$

PageRank is Based on This Concept of Quality Endorsement

Guiding principle: A page (web site) is important if it is cited by other important pages.

Intuitive idea: We can think of PageRank as a kind of fluid that passes from node to node through the edges. Pooling of fluid into some nodes indicates the most important nodes.

PageRank is Based on This Concept of Quality Endorsement

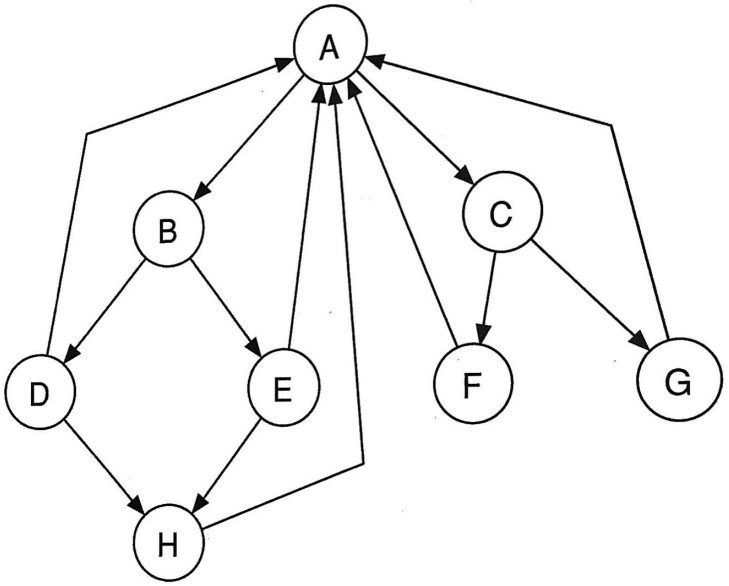
Algorithm:

- Assign all nodes an initial PageRank, $1/n$
- Choose a number of steps, K
- Perform K updates to the PageRank values using the following rule:

Basic PageRank update rule: Each page divides its current PageRank equally across its outgoing edges and passes these to pages it points to. (If there are no outgoing edges, it keeps the PageRank.) Each page updates its new PageRank to be the sum of the shares it receives.

Note that PageRank is conserved over the network, with a total of 1.

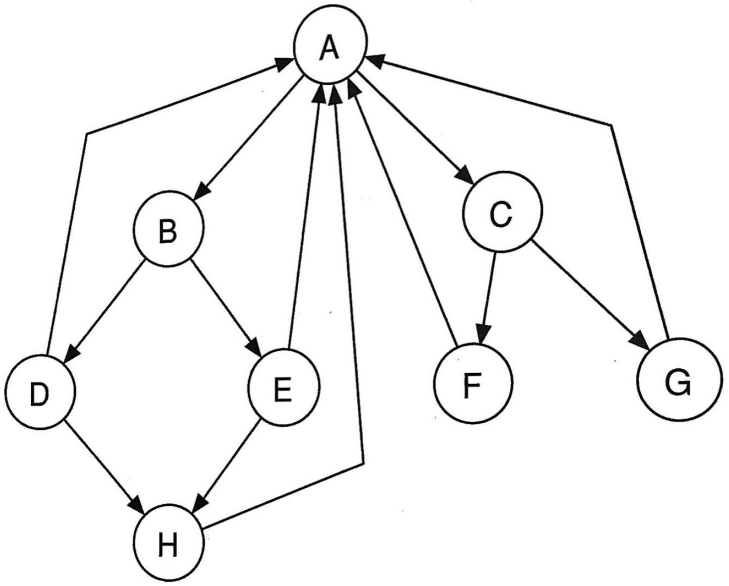
PageRank Iteration



Iteration 0: All nodes start with PageRank of 1/8

Step	A	B	C	D	E	F	G	H
1	$\frac{1}{2}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{8}$

PageRank Iteration

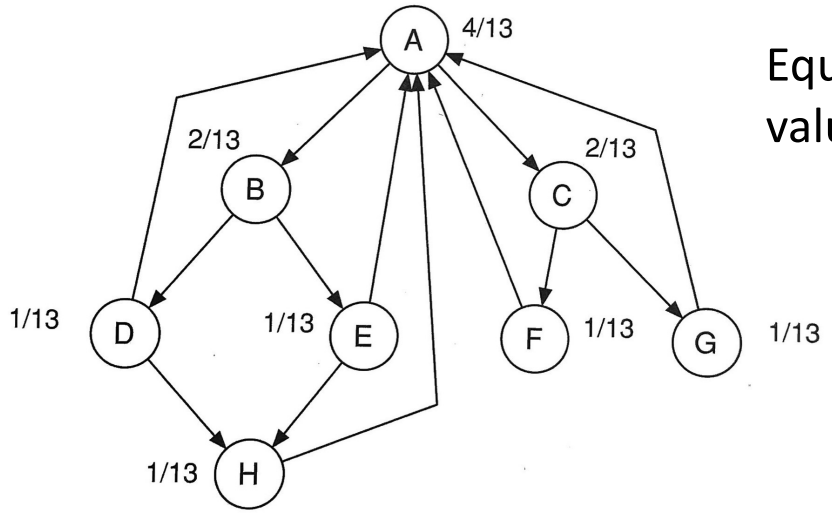


Iteration 0: All nodes start with PageRank of 1/8

Step	A	B	C	D	E	F	G	H
1	$\frac{1}{2}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{8}$
2	$\frac{3}{16}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{16}$

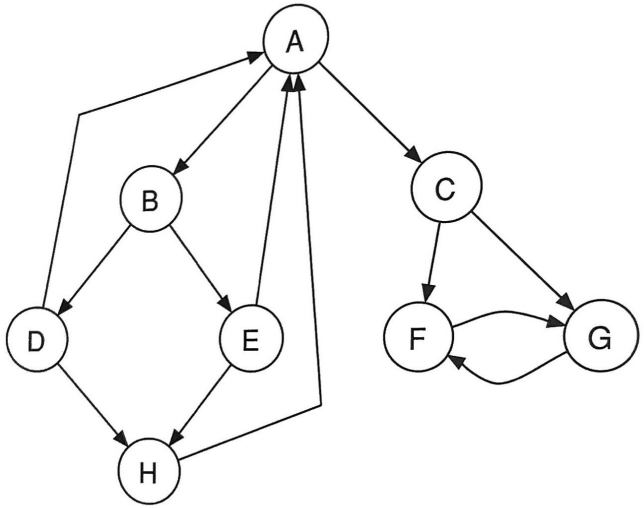
Equilibrium PageRank

If the network is strongly connected, then the iteration converges to a unique set of equilibrium values. That is, it does not matter how PageRank is initially assigned, the values always converge to the same distribution among the pages.



Equilibrium PageRank values

Problem: PageRank Drains Into Cycles



Eventually, as iteration proceeds, everything accumulates in F and G.
The problem is that they form a cycle or loop; PageRank can get in (from C), but can't get out. Any network with a cycle will have the same problem.

Solution: Let it Rain

Each iteration, take some PageRank and randomly distribute it among the pages.

Scaled PageRank Update Rule: First apply the Basic Update Rule. Then scale down all PageRank values by a factor *s*. This means that the total PageRank is reduced from 1 to *s*. Now distribute the residual 1-*s* units of PageRank equally over all nodes, giving (1-*s*)/*n* to each.

This approach converges to a set of equilibrium values, as long as the network is strongly connected.

This is the version used in practice (by Google and probably others) with an *s* value between 0.8 and 0.9.

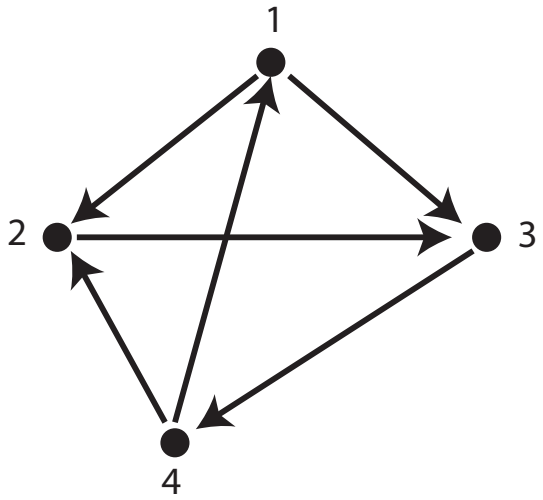
PageRank and Random Walks

Consider someone randomly browsing web pages, starting from a randomly-determined page. They follow links for a sequence of *K* steps: in each step, they pick a random outgoing link (without bias) and follow it. This is a random walk on the network.

The probability of being at page *X* after *K* steps of this random walk is precisely the PageRank of *X* after *K* applications of the Basic PageRank Update Rule!

PageRank as Matrix Iteration

The iterative process for calculating PageRank can be described with matrix-vector multiplication. This uses a matrix *N* that is like the adjacency matrix, but with non-zero entries equal to 1/(out-degree) of each node.



$$N = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1/2 & 1/2 & 0 & 0 \end{bmatrix}$$

Note that each row sums to 1, but not for columns.

If a node has no outward edges, that row will be filled with 0s.

Scaled PageRank Iteration

In terms of matrix-vector multiplication, the basic PageRank update rule is that the value at node *j* is the weighted sum of incoming edges, or the weighted sum down column *j* of matrix *N*: $\vec{r}_k = N^T \vec{r}_{k-1}$.

For the scaled PageRank update, calculate *N* and scale by a factor *s*, then redistribute the remaining (1-*s*) to all nodes. This produces the following matrix (for *s*=0.8):

$$\widetilde{N} = \begin{bmatrix} 0.05 & 0.45 & 0.45 & 0.05 \\ 0.05 & 0.05 & 0.85 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.85 \\ 0.45 & 0.45 & 0.05 & 0.05 \end{bmatrix}$$

The scaled PageRank update rule is then: $\vec{r}_k = \widetilde{N}^T \vec{r}_{k-1}$

At equilibrium (large number of iterations), $\vec{r} = \widetilde{N}^T \vec{r}$, so **the PageRank centrality is the eigenvector of $\widetilde{N}^T$ with (leading) eigenvalue of 1**.

By the Perron-Frobenius Theorem, this eigenvector has non-negative elements.

The End