

Centrality, Connected Components, and Communities

MIE223
Winter 2025

1 Centrality, Connected Components, and Communities

Note 1. centrality can be determined by degree

1.1 what does degree not capture

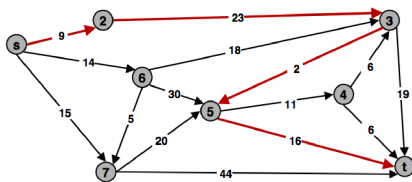
cut vertices connect two components but does not have a high degree

1.2 shortest path in a network

Shortest path network: (V, E, s, t, c) .

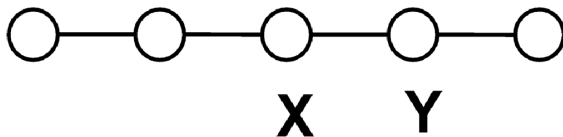
- Directed graph (V, E) .
- Source $s \in V$, sink $t \in V$.
- Arc costs $c(v, w)$.
- Cost of path = sum of arc costs in path.

Cost of path $s - 2 - 3 - 5 - t$
 $= 9 + 23 + 2 + 16$
 $= 48.$



1.3 Betweenness: centrality capturing brokerage

Intuition: how many pairs of individuals would have to go through you in order to reach one another in the minimum number of hops?



1.4 Betweenness: definition

$$C_B(i) = \sum_{j < k} g_{jk}(i) / g_{jk}$$

Where g_{jk} = the number of shortest paths connecting jk
 $g_{jk}(i)$ = the number that vertex i is on.

Usually normalized

by:

$$C'_B(i) = C_B(i) / [(n-1)(n-2)/2]$$

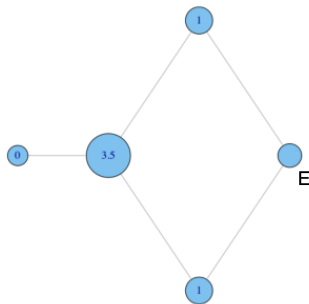
number of pairs of vertices excluding
the vertex itself

a betweenness centrality for node i is the number of nodes other than i who have to through i to interact with each other

$g_{jk}(i)$ is a subset of g_{jk}
 3 ordered pairs with 3 nodes equates to 9 paths

1.5 quiz questions

- What is the betweenness of node E?



- a) 0.5
- b) 1
- c) 1.5
- d) 2

what is betweenness of node E? a) 0.5 to get from 1 to 1 you can go through 3.5 or E
 q2: G high betweenness low degree, A opposite

2 Community detection

Can we discover community structure in an automated way?

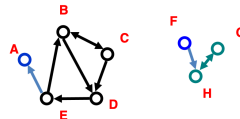
2.1 Connected components

• Strongly connected components

- Each node within the component can be reached from every other node in the component by following directed links

■ Strongly connected components

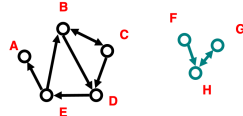
- B C D E
- A
- G H
- F



■ Weakly connected components: every node can be reached from every other node by following links in either direction

■ Weakly connected components

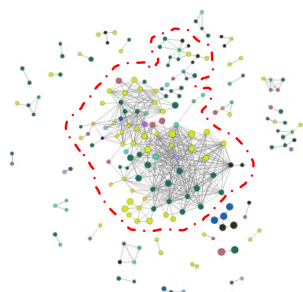
- A B C D E
- G H F



- In undirected networks one talks simply about 'connected components'

2.2 Giant component

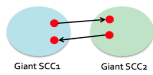
Note 2. if the largest component encompasses a significant fraction of the graph, it is called the giant component



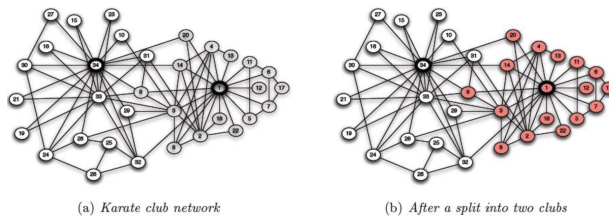
great exam question: generalize equation for betweenness for edges How many paths pass through this edge rather than the node generalize the equation to directed graphs

2.3 No 2 giant SCCs

- There is a giant SCC
- There won't be 2 giant SCCs: Why not?
 - Just takes 1 page from one SCC to link to the other SCC
 - If the components have millions of pages the likelihood of this is very large



2.4 Splitting Zachary Karate Club



do until (2 disconnected graphs) compute betweenness first for the given graph remove edge has highest betweenness

2.5 betweenness clustering

Algorithm

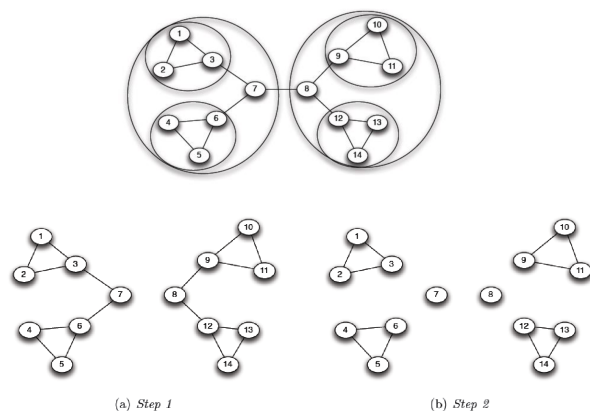
- Compute betweenness for all edges
- while (betweenness of any edge \geq threshold):
- remove edge with highest betweenness
- recalculate betweenness

Betweenness needs to be recalculated at each step

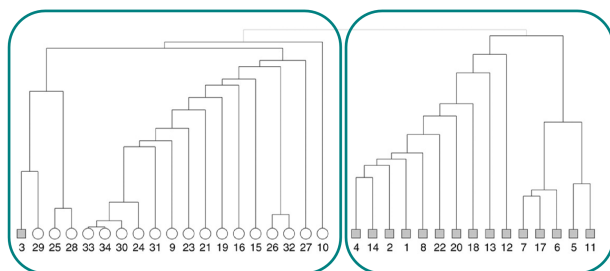
- removal of an edge can impact the betweenness of another edge
- very expensive: all pairs shortest path – $O(N^3)$
- may need to repeat up to N times
- does not scale to more than a few hundred nodes, even with the fastest algorithms

2.6 betweenness clustering:

successively remove edges of highest betweenness (the bridges, or local bridges), breaking up the network into separate components



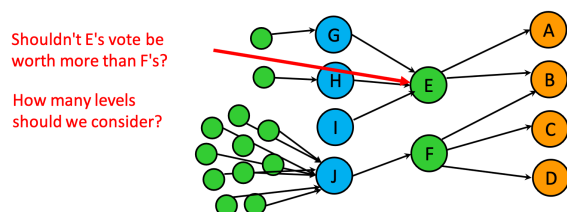
2.7 betweenness clustering algorithm & the karate club data set



dendrogram: tree diagram that shows the hierarchical relationship between objects

3 Google's PageRank

3.1 PageRank: Intuition



Imagine a contest for The Web's Best Page

- Initially, each page has one vote
- Each page votes for all the pages it has a link to
- To ensure fairness, pages voting for more than one page must split their vote equally between them
- Voting proceeds in rounds; in each round, each page has the number of votes it received in the previous round
- In practice, it's a little more complicated - but not much!

3.2 PageRank

Each page i is given a rank x_i

Goal: Assign the x_i such that the rank of each page is governed by the ranks of the pages linking to it:

$$x_i = \sum_{j \in B_i} \frac{1}{N_j} x_j$$

Rank of page i

How do we compute the rank values?

Every page j that links to i

Rank of page j

Number of links out from page j

3.3 Iterative PageRank (simplified)

Initialize all ranks to be equal, e.g.:

$$x_i^{(0)} = \frac{1}{n}$$

Iterate until convergence

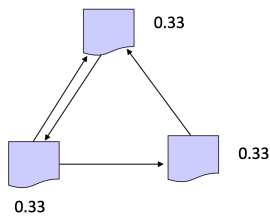
$$x_i^{(k+1)} = \sum_{j \in B_i} \frac{1}{N_j} x_j^{(k)}$$

3.4 Example

Example: Step 0

Initialize all ranks to be equal

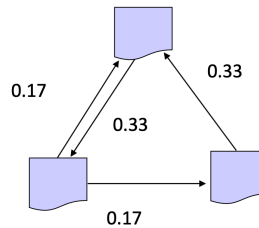
$$x_i^{(0)} = \frac{1}{n}$$



Example: Step 1

Propagate weights across out-edges

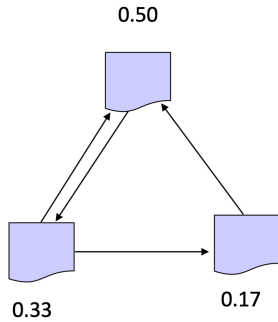
$$x_i^{(k+1)} = \sum_{j \in B_i} \frac{1}{N_j} x_j^{(k)}$$



Example: Step 2

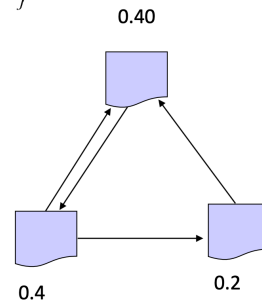
Compute weights
based on in-edges

$$x_i^{(1)} = \sum_{j \in B_i} \frac{1}{N_j} x_j^{(0)}$$



Example: Convergence

$$x_i^{(k+1)} = \sum_{j \in B_i} \frac{1}{N_j} x_j^{(k)}$$



3.5 Naïve PageRank Algorithm Restated

Let

- $N(p)$ = number outgoing links from page p
- B_p = set of pages that back-link to page p

$$PageRank(p) = \sum_{b \in B_p} \frac{1}{N(b)} PageRank(b)$$

Each page b distributes its importance to all of the pages it points to (so we scale by $1/N(b)$)
Page p 's importance is increased by the importance of its back set

3.6 In Linear Algebra formulation

– Create an $m \times m$ matrix M to capture links:

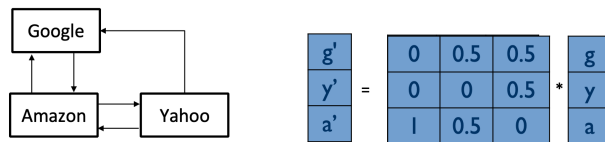
$$M(i, j) = \begin{cases} \frac{1}{N_j} & \text{if page } i \text{ is pointed to by page } j \text{ and} \\ & \text{page } j \text{ has } N_j \text{ outgoing links} \\ 0 & \text{otherwise} \end{cases}$$

– Initialize all PageRanks to 1, multiply by M repeatedly until all values converge:

$$\begin{bmatrix} PageRank(p_1') \\ PageRank(p_2') \\ \dots \\ PageRank(p_m') \end{bmatrix} = M \begin{bmatrix} PageRank(p_1) \\ PageRank(p_2) \\ \dots \\ PageRank(p_m) \end{bmatrix}$$

– Computes **principal eigenvector** via **power iteration**

3.7 A Brief Example



Running for multiple iterations:

$$\begin{bmatrix} g \\ y \\ a \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0.5 \\ 1.5 \end{bmatrix}, \begin{bmatrix} 1 \\ 0.75 \\ 1.25 \end{bmatrix}, \dots, \begin{bmatrix} 1 \\ 0.67 \\ 1.33 \end{bmatrix}$$

Total rank sums to number of pages

3.8 Random Surfer Model

PageRank has an intuitive basis in random walks on graphs
Imagine a random surfer, who starts on a random page and, in each step,

- with probability d , clicks on a random link on the page
- with probability $1-d$, jumps to a random page (bored?)

The PageRank of a page can be interpreted as the fraction of steps the surfer spends on the corresponding page

Transition matrix can be interpreted as a Markov Chain

3.9 Recap: PageRank

Estimates absolute 'quality' or 'importance' of a given page based on inbound links

- Can be computed via fixpoint iteration
- Can be interpreted as the fraction of time a 'random surfer' would spend on the page
- Several refinements (not covered here)

An important factor for Google ranking of web pages, but not the only one

Overall ranking is based on many factors (Google: ~200)

Aside: What could be the other 200 factors?

	Positive	Negative
On-page	Keyword in title? URL? Keyword in domain name? Page freshness Rate of change ...	Links to 'bad neighborhood' Keyword stuffing Over-optimization Hidden content (text has same color as background) Automatic redirect/refresh ...
Off-page	High PageRank Anchor text of inbound links Links from authority sites Links from well-known sites Domain expiration date ...	Fast increase in number of inbound links (link buying?) Link farming Different pages user/spider Content duplication ...

- Note: This is entirely speculative!