13: Betweenness Centrality

Machine Learning and Real-world Data (MLRD)

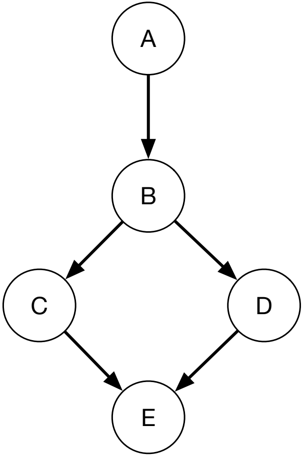
Paula Buttery (based on slides by Simone Teufel)

Last session: some simple network statistics

- You measured the **degree** of each node and the **diameter** of the network.
- Next two sessions:
  - Today: finding gatekeeper nodes via **betweenness centrality**.
  - Next session: using betweenness centrality of edges to split graph into cliques.
- Reading for social networks (all sessions):
  - Easley and Kleinberg for background: Chapters 1, 2, 3 and first part of Chapter 20.
  - Brandes algorithm: two papers by Brandes (links in practical notes).

Centralities help us talk about interesting nodes

- **Degree**: the number of edges connected to a node (can be split into incoming and outgoing) (discovers direct influencers)
- **Closeness**: average of the distances from the node (discovers indirect influencers)
- **Betweenness**: relative number of shortest paths that rely on the node (discovers gatekeepers)



Gatekeepers nodes are associated with local bridges

- Last time we saw the concept of local bridge: an edge which increased the shortest paths if cut.

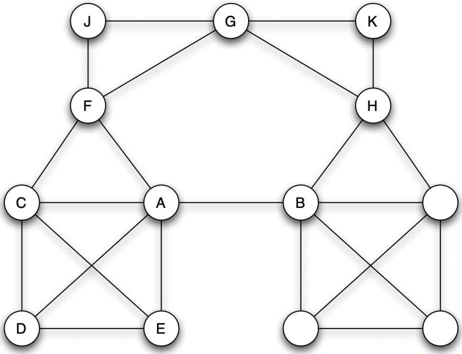
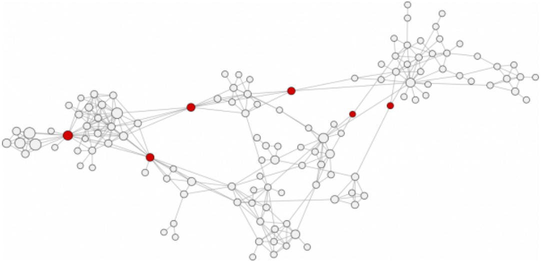


Figure 3-4 from Easley and Kleinberg (2010)

- A–B is a local bridge here.

Nodes with high betweenness are on relatively many shortest paths

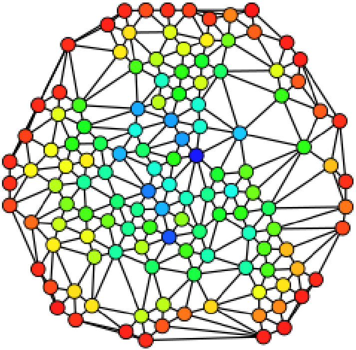
- The betweenness centrality of a node V is defined in terms of the proportion of shortest paths that go through V for each pair of nodes.



https://www.linkedin.com/pulse/wtf-do-you-actually-know-who-influencers-walter-pike

- Here: the red nodes have high betweenness centrality. because we only care about shortest paths.

Betweenness: example



Claudio Rocchini: https://commons.wikimedia.org/wiki/File:Graph_betweenness.svg

- Betweenness: red is minimum; dark blue is maximum.

Betweenness centrality, formally

- Directed graph $G =< V, E >$
- $\sigma(s, t)$: number of shortest paths between nodes s and t
- $\sigma(s, t|v)$: number of shortest paths between nodes s and t that pass through v.
- $C_B(v)$, the betweenness centrality of v:

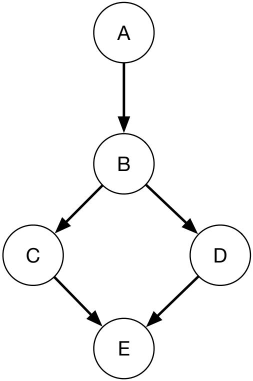
$$C_B(v) = \sum_{s,t \in V} \frac{\sigma(s, t|v)}{\sigma(s, t)}$$

Calculating betweenness verbosely

V = {A, B, C, D, E}
E = {(A,B), (B,C), (B,D), (C,E), (D,E)}

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

path	route	A	B	C	D	E
(A,B)	A-B					
(A,C)	A-B-C		1			
(A,D)	A-B-D		1			
(A,E)	A-B-C-E		0.5	0.5		
	A-B-D-E		0.5		0.5	
(B,C)	B-C					
(B,D)	B-D					
(B,E)	B-C-E			0.5		
	B-D-E				0.5	
(C,E)	C-E					
(D,E)	D-E					
		0	3	1	1	0



Calculating Betweenness with Brandes

1) Find number of shortest paths:

- $\sigma(s,t)$ can be calculated recursively:

$$\sigma(s,t) = \sum_{u \in Pred(t)} \sigma(s,u)$$

  - $Pred(t) = \{u\colon (u,t) \in E, d(s,t) = d(s,u) + 1\}$
    predecessors of t on shortest path from s
  - $d(s,u)$: Distance between nodes s and u

- Using a Breadth First search with each node as source s once, gives total complexity of $O(V(V+E))$.

Calculating Betweenness with Brandes

2) Find dependency on specific nodes for specific shortest paths:

- There are a cubic number of pairwise dependencies $\delta(s, t|v)$ where:

$$\delta(s, t|v) = \frac{\sigma(s, t|v)}{\sigma(s, t)}$$

- Brandes algorithm intuition: the dependencies can be aggregated without calculating them all explicitly.
- Can calculate dependency of s on v based on dependencies one step further away.

Calculating Betweenness with Brandes

2) Find dependency on specific nodes for specific shortest paths:

- Define one-sided dependencies (how dependant are shortest paths from s on v):

$$\delta(s|v) = \sum_{t \in V} \delta(s,t|v)$$

- Then Brandes (2001) shows:

$$\delta(s|v) = \sum_{\substack{(v,w)\in E \\ w:\, d(s,w)=d(s,v)+1}} \frac{\sigma(s,v)}{\sigma(s,w)}.(1+\delta(s|w))$$

And: $C_B(v) = \sum_{s \in V} \delta(s|v)$

Calculating Betweenness with Brandes

Algorithm:

- For all vertices $s \in V$:
- Calculate $\delta(s|v)$ for all $v \in V$ in two phases:
  1. Breadth-first search, calculating distances and shortest path counts from s, (push all vertices onto stack as they're visited).
  2. Visit all vertices in reverse order (pop off stack), aggregating dependencies according to equation.

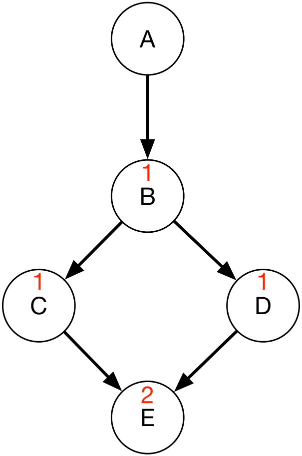
Calculating Betweenness with Brandes: example

$$\sigma(s,t) = \sum_{u \in Pred(t)} \sigma(s,u)$$

where

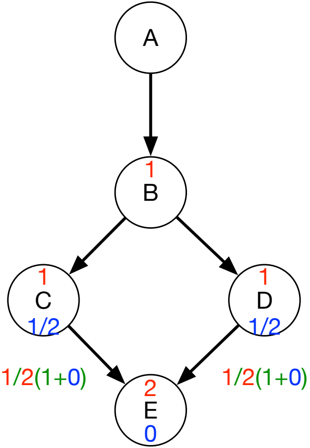
$$Pred(t) = \{u\colon (u,t) \in E, d(s,t) = d(s,u)+1\}$$

and $d(s,t)$ is distance between s and t



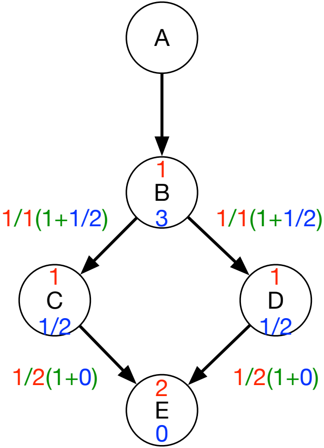
Calculating Betweenness with Brandes: example

$$\delta(s|v) = \sum_{\substack{(v,w)\in E \\ w:\, d(s,w)=d(s,v)+1}} \frac{\sigma(s,v)}{\sigma(s,w)}.(1+\delta(s|w))$$



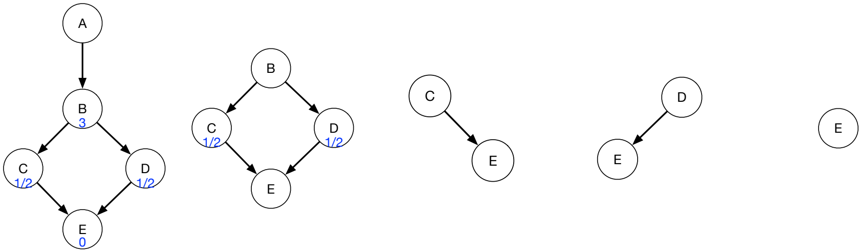
Calculating Betweenness with Brandes: example

$$\delta(s|v) = \sum_{\substack{(v,w)\in E \\ w:\, d(s,w)=d(s,v)+1}} \frac{\sigma(s,v)}{\sigma(s,w)}.(1+\delta(s|w))$$



Calculating Betweenness with Brandes: example

$$C_B(v) = \sum_{s \in V} \delta(s|v)$$



$$C_B(A) = 0$$

$$C_B(B) = 3$$

$$C_B(C) = 1$$

$$C_B(D) = 1$$

$$C_B(E) = 0$$

Calculating Betweenness with Brandes

- The algorithm is for directed graphs.
- But undirected graphs are easy: the algorithm works in exactly the same way, except that each pair is considered twice, once in each direction.
- Therefore: halve the scores at the end for undirected graphs.
- Brandes has lots of other variants, including edge betweenness centrality, which we'll use in the next session.

Today

- Task 11: Implement the Brandes algorithm for efficiently determining the betweenness of each node.

Literature

- Detailed notes on the Brandes algorithm on course page / Moodle.
- Easley and Kleinberg (2010, page 79-82). But this is an informal description.
- Ulrich Brandes (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*. 25:163–177.
- Ulrich Brandes (2008) On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*. 30 (2008), pp. 136–145