Graph Mining
SD212
4. Betweenness centrality

Thomas Bonald

2017 – 2018

Motivation

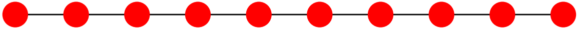
How to identify the most “central” nodes in a graph?

Useful for:

- viral marketing
- information spreading
- content recommendation
- security

We focus on **undirected, unweighted** graphs; extensions are discussed later.

Why is the degree not sufficient?



Outline

1. Notion of betweenness centrality
2. Naive algorithm
3. Brandes' algorithm
4. Extensions
5. Other centrality metrics

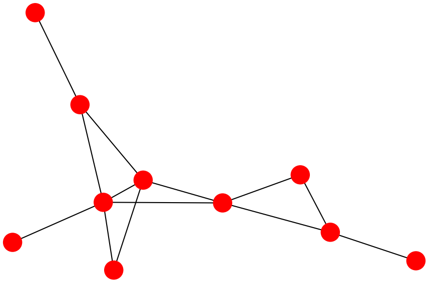
Betweenness centrality

The betweenness centrality of node u is the **fraction of shortest paths** going through u, summed over all source-destination pairs:

$$C(u) = \sum_{s,t \neq u; s<t} \frac{\sigma_{st}(u)}{\sigma_{st}}$$

where

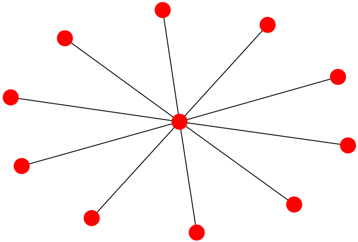
- σ_{st} = number of shortest paths between s and t
- $\sigma_{st}(u)$ = number of shortest paths between s and t **through** u



Normalization

Observe that:

$$0 \leq C(u) \leq \binom{n-1}{2}$$



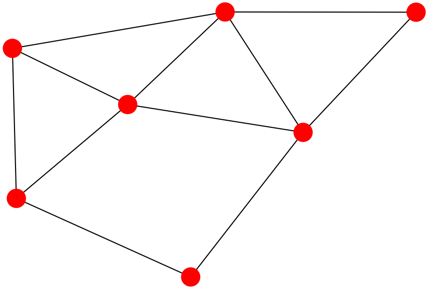
The **normalized** betweenness centrality of node u is the **probability** that a random shortest path goes through u:

$$\bar{C}(u) = \frac{1}{\binom{n-1}{2}} \sum_{s,t \neq u; s<t} \frac{\sigma_{st}(u)}{\sigma_{st}} \implies 0 \leq \bar{C}(u) \leq 1$$

Example

Counting the number of shortest paths

Adaptation of BFS (Breadth First Search)



First algorithm

Observe that

$$\sigma_{st}(u) = \begin{cases} \sigma_{su}\sigma_{ut} & \text{if } d_{st} = d_{su} + d_{ut} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Algorithm

- Compute the distance and the number of shortest paths between **each** pair of nodes
- Return the betweenness centrality of each node u using (1)

Complexity: $O(n^3)$ in time, $O(n^2)$ in memory

Recursion

Let

$$\delta_s(u) = \sum_{t \neq u} \frac{\sigma_{st}(u)}{\sigma_{st}}$$

Theorem (Brandes 2001)

$$\delta_s(u) = \sum_{v: d_{sv} = d_{su} + 1} \frac{\sigma_{su}}{\sigma_{sv}} (1 + \delta_s(v))$$

Brandes' algorithm

$$\delta_s(u) = \sum_{v:d_{sv}=d_{su}+1} \frac{\sigma_{su}}{\sigma_{sv}}(1+\delta_s(v)) \qquad (2)$$

Algorithm

For each node s:

- apply BFS from node s
- init $\delta_s(u) = 0$ for all nodes u
- apply the recursion (2) starting from the most distant nodes

Return $\frac{1}{(n-1)(n-2)} \sum_{s \neq u} \delta_s(u)$

Complexity: $O(nm)$ in time, $O(n)$ in memory

Approximation by node sampling

Algorithm

Choose some set $S \subset V$ of k nodes (e.g., at random)
For each node $s \in S$:

- apply BFS from node s
- init $\delta_s(u) = 0$ for all nodes u
- apply the recursion (2) starting from the most distant nodes

Return $\frac{1}{|S \setminus \{u\}|(n-2)} \sum_{s \in S, s \neq u} \delta_s(u)$

Complexity: $O(km)$ in time, $O(n)$ in memory

Proof of the recursion

Lemma 1

For any $s, t \neq u$,

$$\sigma_{st}(u) = \sum_{v : d_{sv} = d_{su} + 1} \sigma_{st}(u, v)$$

Lemma 2

For any $s, t \neq u$ and v such that $d_{sv} = d_{su} + 1$,

$$\sigma_{st}(u, v) = \frac{\sigma_{su}}{\sigma_{sv}} \sigma_{st}(v)$$

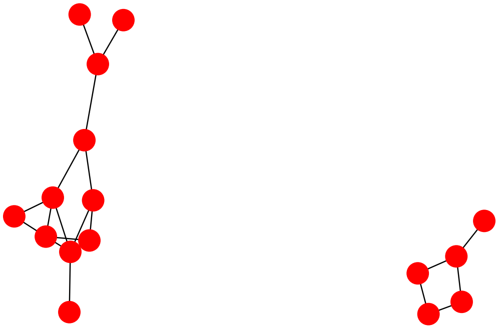
Outline

1. Notion of betweenness centrality
2. A first algorithm
3. Brandes' algorithm
4. **Extensions**
5. Other centrality metrics

Disconnected graphs

Betweenness centrality

$$C(u) = \sum_{s,t \neq u; s<t; \sigma_{st}>0} \frac{\sigma_{st}(u)}{\sigma_{st}}$$



Edge betweenness centrality

The betweenness centrality of edge $\{u, v\}$ is the **fraction of shortest paths** going through edge $\{u, v\}$, summed over all source-destination pairs:

$$C(u, v) = \sum_{s,t \neq u,v; s<t; \sigma_{st}>0} \frac{\sigma_{st}(u, v)}{\sigma_{st}}$$

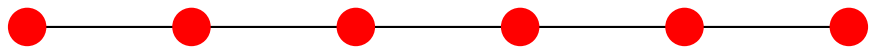
where

- σ_{st} = number of shortest paths between s and t
- $\sigma_{st}(u, v)$ = number of shortest paths between s and t **through** $\{u, v\}$

Normalization:

$$\bar{C}(u, v) = \frac{1}{\binom{n-2}{2}} \sum_{s,t \neq u,v; s<t; \sigma_{s,t}>0} \frac{\sigma_{st}(u, v)}{\sigma_{st}}$$

Example



Computation

For any $s \neq u, v$, let

$$\delta_s(u,v) = \sum_{t \neq u,v} \frac{\sigma_{st}(u,v)}{\sigma_{st}}$$

Then

$$\delta_s(u,v) = \begin{cases} \delta_s(v)\frac{\sigma_{su}}{\sigma_{sv}} & \text{if } d_{sv} = d_{su} + 1, \\ 0 & \text{otherwise} \end{cases}$$

Complexity: $O(nm)$ in time, $O(m)$ in memory

Directed graphs

- Betweenness centrality

$$C(u) = \sum_{s,t \neq u; \sigma_{st} > 0} \frac{\sigma_{st}(u)}{\sigma_{st}}$$

- Normalization

$$\bar{C}(u) = \frac{1}{(n-1)(n-2)} C(u)$$

- Brandes' algorithm

$$C(u) = \sum_{s \neq u} \delta_s(u)$$

Weighted graphs

Assume additive weights

- BFS $\rightarrow$ Dijkstra
- Recursion

$$\delta_s(u) = \sum_{v:d_{sv}=d_{su}+w_{uv}} \frac{\sigma_{su}}{\sigma_{sv}}(1+\delta_s(v))$$

- Complexity $O(nm) \rightarrow O(mn + n^2 \log n)$

Outline

1. Notion of betweenness centrality
2. Naive algorithm
3. Brandes' algorithm
4. Extensions
5. **Other centrality metrics**

Closeness centrality

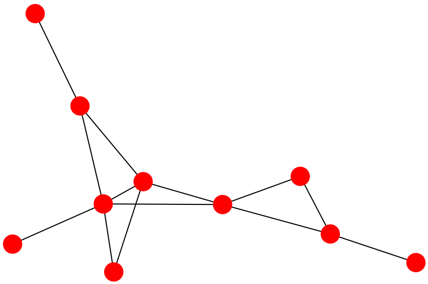
The closeness centrality is related to the **average distance** to all other nodes,

$$CC(u) = \left(\frac{1}{n-1} \sum_{v \neq u} d_{uv} \right)^{-1}$$

Complexity: $O(nm)$ in time, $O(n)$ in memory

Random walk betweenness centrality

- Shortest paths $\rightarrow$ random paths
- The random walk betweenness centrality of node u is the **expected number of visits** to u for a random path starting from s and ending in t, summed over all pairs s, t
- Analogy with the **electric current**



Summary

Betweenness centrality:

- A useful metric to quantify the "centrality" of nodes in terms of **shortest paths**
- High **complexity**: $O(nm)$ for exact calculation, $O(km)$ for approximate value
- May be **combined** with other metrics
  (degree, closeness, random walk betweenness, PageRank, etc.)