Social Network Analysis

COSC–254 — March 25–April 10, 2019
Outline

Paths, shortest paths, diameter, and Breadth-First Search

Social networks properties

Degree centrality and prestige

Closeness centrality

Betweenness centrality

Finding communities with betweenness centrality

Link prediction
Paths and shortest paths

Finding shortest paths: Breadth-First Search

2-Approximation algorithm for the diameter
Graph $G = (V, E)$.
Two vertices $u, v \in V$.

A path $p$ from $u$ to $v$ is an ordered sequence of vertices:

$$p = (u, w_1, \ldots, w_\ell, v)$$

such that $(u, w_1) \in E, (w_i, w_{i+1}) \in E$ for each $i = 1, \ldots, \ell - 1, w_\ell, v \in E$.

$p = (1, 3, 5, 2)$ is a path from 1 to 2
**Shortest Paths**

*Length* of a path \( p = (w_1, \ldots, w_\ell) \) is \( \ell - 1 \) (number of edges on the path)

The *distance* \( d(u, v) \) from \( u \) to \( v \) is the *minimum length* of any path from \( u \) to \( v \).

A *shortest path* (SP) from \( u \) to \( v \) is a path from \( u \) to \( v \) of length \( d(u, v) \).

There may be multiple SPs between \( u \) and \( v \).

\[
p = (2, 5, 1, 3, 4) \text{ is a path from } 2 \text{ to } 4 \text{ of length } 4.
\]

\[
(2, 5, 3, 4) \text{ is a SP from } 2 \text{ to } 4 \text{ of length } 3, \text{ so } d(2, 4) = 3
\]

\[
(2, 1, 3, 4) \text{ is also a SP from } 2 \text{ to } 4
\]
Section outline

✓ Paths and shortest paths

Finding shortest paths: Breadth-First Search

2-Approximation algorithm for the diameter
Finding the Shortest Paths

The *Breadth-First-Search* algorithm computes, for a *source* vertex \( v \), the SP distance \( d(v, u) \) for every \( u \in V \).

Idea:

1. Start from \( v \) and *visit* all its neighbors.  
   They have distance 1 from \( v \);
2. Now explore all the unvisited neighbors of the neighbors of \( v \).  
   They have distance 2 from \( v \);
   …(continue as long as there are no unvisited nodes)

*All nodes* at distance \( k \) from \( v \) are *visited before any node* at distance \( k + 1 \) from \( v \).
**BFS**

**INPUT:** graph $G$, source node $v \in V$

**OUTPUT:** a list $d$ such that $d[u] = d(v, u)$

For each $u \in V$: $T[u] \leftarrow \text{false}$, $d[u] \leftarrow \infty$;

$Q \leftarrow \emptyset$

$d[v] \leftarrow 0$, $T[v] \leftarrow \text{true}$, Enqueue $v$ in $Q$

While $Q \neq \emptyset$:

$u \leftarrow \text{Dequeue}(Q)$

For each $z \in N(u)$:

If $T[z] = \text{false}$:

$d[z] \leftarrow d[u] + 1$

$T[z] \leftarrow \text{true}$

Enqueue $z$ in $Q$

Return $d$

When we dequeue a node $u$, we iterate through $N(u)$ and enqueue the unvisited neighbors.

Takes time

$O(|N(u)|) = O(\deg(u))$

We dequeue each node exactly once.

So the running time of BFS is

$O(\sum_{u \in V} \deg(u)) = O(|E|)$
Section outline

- Paths and shortest paths
- Finding shortest paths: Breadth-First Search

2-Approximation algorithm for the diameter
Diameter of a graph

The \textit{diameter of} $G$ is the \textit{longest distance} between a pair of vertices of $V$:

$$\text{diam}(G) = \max\{d(u, v) : u, v \in V\}$$

Computing \text{diam}(G) requires computing \textit{All-Pair Shortest Paths}

With $n$ BFSs, we can compute \text{diam}(G) in time $O(|V| \cdot |E|)$

\begin{tabular}{ccc}
\hline
$u$ & $v$ & $d(u, v)$ \\
\hline
1 & 2 & 1 \\
1 & 3 & 1 \\
1 & 4 & 2 \\
1 & 5 & 1 \\
2 & 3 & 2 \\
2 & 4 & 3 \\
2 & 5 & 1 \\
3 & 4 & 1 \\
3 & 5 & 1 \\
4 & 5 & 2 \\
\hline
\end{tabular}
Triangle Inequality

Given $d(u, v)$ and $d(v, w)$, what can we say about $d(u, w)$?

The *triangle inequality* must hold:

$$d(u, w) \leq d(u, v) + d(v, w)$$

(allowing properties of *distances* and *norms*).

$v = 3, u = 2, w = 5$, 

<table>
<thead>
<tr>
<th>$u$</th>
<th>$v$</th>
<th>$d(u, v)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
Upper bound to the distances

Fix a vertex $v$, and assume to know $d(v, z)$ for every $z \in V$.

From the triangle inequality it holds that for any $u$ and $w$

$$d(u, w) \leq d(u, v) + d(v, w)$$

Can we find an upper bound to the r.h.s.?

There must be a node $q$ such that

$$d(v, q) = \max\{d(v, z), z \in V\}$$

but then, $d(u, v) \leq d(v, q)$, and $d(v, w) \leq d(v, q)$. Hence,

$$d(u, w) \leq 2 \max\{d(v, z), z \in V\}$$
\[ d(u, w) \leq 2 \max\{d(v, z), z \in V\} \]

\begin{align*}
v &= 3, \quad u = 2, \quad w = 5,
\end{align*}
Upper bound to the diameter

For any $v, u, \text{ and } w$, it holds $d(u, w) \leq 2 \max \{d(v, z), z \in V\}$

True for any $u, w$, thus for any $u^*, w^*$ such that $d(u^*, w^*) = \text{diam}(G)$. So,

$$\text{diam}(G) \leq 2 \max \{d(v, z), z \in V\}$$

$v = 3$

<table>
<thead>
<tr>
<th>$u$</th>
<th>$v$</th>
<th>$d(u, v)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
Approximation algorithm for the diameter

**Input:** A graph $G = (V, E)$,  
**Output:** a 2-approximation of $\text{diam}(G)$.

1. Choose any vertex $v \in V$
2. Run BFS from $v$, to obtain $\{d(v, z), z \in V\}$
3. Return $2 \max\{d(v, z), z \in V\}$
Section outline

✓ Paths and shortest paths

✓ Finding shortest paths: Breadth-First Search

✓ 2-Approximation algorithm for the diameter
Outline

Paths, shortest paths, diameter, and Breadth-First Search

Social networks properties

Degree centrality and prestige

Closeness centrality

Betweenness centrality

Finding communities with betweenness centrality
Section outline

Homophily, triadic closure, and clustering coefficient

Preferential attachment and its consequences
Social networks

Facebook, Twitter, Instagram, *phone calls, collaborations, scientific citations* …

*Actors*: entities participating in the network, represented as nodes

They have *features* that characterizes them

*Relationships* between entities, represented as edges

Could be *directed or undirected*

The graph is “*not completely random*”, especially at a local level (i.e., at small scale).
Homophily

A first example of *local non-randomness*

Vertices that are connected to each other, are *likely* to have *similar features*

That’s what the edge represents:

there is some connection, similarity, shared aspect of life between the vertices

A property of *features* that is *inferred* from the *graph structure*

A property of the structure that is *caused* by a property of features
Triadic closure

**Triadic closure**: If two actors $u, v$ have a common friend $w$, then they are *more likely to be connected* than two actors that have no common friends.

If $u, v$ are not connected now, they are *more likely to become connected* in the future.

**Consequence of homophily**: $u$ and $w$ (may) have similar features, and so so for $v$ and $w$, thus $v$ and $u$ may have similar features.

Inherently a property of the *structure*
A measure of *how strong triadic closure is* for the neighbors of a node $v$

Cluster coefficient of $v$: *How often* are two of $v$’s friends also *friends with each other*?

Neighbors of $v$: $N(v) = \{(u,v) \in E\}$, Cluster coefficient of $v$:

$$\eta(v) = \frac{|\{(u,w) \in E : u,w \in N(v)\}|}{\left(\frac{|N(v)|}{2}\right)} \in [0, 1]$$
Clustering coefficient

\[ \eta(v) = \frac{|\{(u, w) \in E : u, w \in N(v)\}|}{\binom{n_v}{2}} \]

\( v = 3, \ N(v) = \{4, 5, 1\}, \ \eta(v) = \frac{1}{3} \)

\( v = 5, \ N(v) = \{3, 1, 2\}, \ \eta(v) = \frac{2}{3} \)
A community $A$ is a subset of $V$ such that the density

$$\text{density}(A) = \frac{|\{(u, v) \in E : u, v \in A\}|}{\binom{|A|}{2}}$$

is high.

Vertices in a community are very likely to be connected to each other.

The more communities $v$ belongs to, the lower $\eta(v)$ is likely to be.
Homophily, triadic closure, and clustering coefficient

Preferential attachment and its consequences
Modeling network formation

Social networks and many other graphs (e.g., the Web) *grows over time*

Most of the time, new edges are created. From time to time, nodes are added. Rarely, edges and vertices are deleted.

We can *model the evolution* with a *random process* of network formation.

The properties of a network are *determined by the model* we choose.

“All models are wrong, but some are useful” — G. Box (attrib.)

Useful models *create networks* with properties *similar to real networks*
**Preferential attachment**

A useful model of network formation, although not perfect

There are better ones, but too complex for us

Parameters: new-vertex probability $a \in (0, 1)$, preference parameter $g > 0$

At time $t$:

1. With probability $a$, add a *new unconnected* vertex;
2. Add an edge $(u, v)$, choosing $u$ and $v$ independently, with probability *proportional to their current degree*

$$\pi_t(w) \propto \text{deg}^g(w) = |N(v)|^g$$

Different values of $g$ create different networks.

For social networks, usually $g \approx 1$. 
Consequences of preferential attachment

Power-law degree distribution

Friendship paradox

Giant connected component

Densification

Small-world property

Shrinking diameter
Power-law degree distribution

*Few high-degree nodes* attract most of the newly-added edges:

The richer gets richer

The fraction $P(k)$ of nodes with degree $k$ has a *power-law* distribution:

$$P(k) \propto k^{-\gamma}, \quad \text{with } 2 < \gamma < 3$$

Values of $\gamma$ closer to 2 have a less skewed distribution
Friendship paradox

Most people have *fewer* friends than their friends have, *on average*.

Average number of friends: \( \frac{1}{|V|} \sum_{v \in V} \deg(v) = \frac{2|E|}{|V|} = \mu \)

Number of friends of an average friend: How to compute it?

\[
\frac{1}{2|E|} \sum_{(u,z) \in E} (\deg(z) + \deg(u)) = \frac{1}{2|E|} \sum_{v \in V} \deg(v)^2
\]

Let’s show that

\[
\mu \leq \frac{1}{2|E|} \sum_{v \in V} \deg(v)^2
\]
\[
\frac{1}{2|E|} \sum_{v \in V} \deg(v)^2 = \frac{1}{\mu} \frac{1}{2|E|} \sum_{v \in V} \deg(v)^2 \quad \text{(recall } \mu = \frac{2|E|}{|V|})
\]

\[
= \frac{1}{\mu} \frac{1}{|V|} \left( \sum_{v \in V} \deg(v)^2 + \sum_{v \in V} \left( \frac{4|E|^2}{|V|^2} - \frac{4|E|}{|V|} \deg(v) \right) - \sum_{v \in V} \left( \frac{4|E|^2}{|V|^2} - \frac{4|E|}{|V|} \deg(v) \right) \right)
\]

\[
= \frac{1}{\mu} \frac{1}{|V|} \left( \sum_{v \in V} \left( \deg(v)^2 + \frac{4|E|^2}{|V|^2} - \frac{4|E|}{|V|} \deg(v) \right) - \sum_{v \in V} \left( \frac{4|E|^2}{|V|^2} - \frac{4|E|}{|V|} \deg(v) \right) \right)
\]

\[
= \frac{1}{\mu} \frac{1}{|V|} \left( \sum_{v \in V} \left( \deg(v) - \frac{2|E|}{|V|} \right)^2 - \frac{4|E|^2}{|V|} + \frac{4|E|}{|V|} \sum_{v \in V} \deg(v) \right)
\]

\[
= \frac{1}{\mu} \frac{1}{|V|} \left( \sum_{v \in V} (\deg(v) - \mu)^2 - \frac{4|E|^2}{|V|} + \frac{8|E|^2}{|V|} \right) = \frac{1}{\mu} \left( \frac{1}{|V|} \sum_{v \in V} (\deg(v) - \mu)^2 + \frac{1}{|V|} \frac{4|E|^2}{|V|} \right)
\]

\[
= \frac{1}{\mu} \left( \sigma^2 + \mu^2 \right) = \frac{\sigma^2}{\mu} + \mu \geq \mu
\]
Giant connected component

There is a *path from most nodes to most nodes*

Why?
1. Most high-degree nodes are connected to each other;
2. Most other nodes are connected to at least one high-degree node.
Densification

The graph density

\[ \frac{|E|}{\left(\frac{|V|^2}{2}\right)} \]

grows over time.

Why? We are \textit{adding more edges than nodes}.

If \( n(t) \) is the number of nodes at time \( t \), the number of edges is

\[ e(t) \propto n(t)^\beta \quad \text{with} \quad 1 \leq \beta \leq 2 \]

\( \beta = 1 \Rightarrow \) average degree does not change over time
\( \beta = 2 \Rightarrow \) the network does not become denser
Small-world Property

Small-world: the \textit{average SP distance} is small.

Milgram’s experiment: \( \leq 6 \textit{degrees of separation} \) between most individuals in the U.S. (the experiment failed) Why do we see this effect?

High-degree nodes allow to \textit{move very fast} across the network.
Many models predict the average SP distance to grow as $O(\log n)$.

In practice, the *average SP distance shrinks*

Why? We are adding more edges than nodes,
    each edge shortens the distance between at least some pairs of nodes

This behavior was unexpected when it was observed
    (before the preferential attachment model was developed)
Consequences of preferential attachment

The network *evolves* over time, and we can study *how different properties change.*

- ✓ Power-law degree distribution
- ✓ Friendship paradox
- ✓ Giant connected component
- ✓ Densification
- ✓ Small-world property
- ✓ Shrinking diameter
Section outline

- Homophily, triadic closure, and clustering coefficient
- Preferential attachment and its consequences
Outline

- Paths, shortest paths, diameter, and Breadth-First Search
- Social networks properties
  - Degree centrality and prestige
  - Closeness centrality
  - Betweenness centrality
  - Finding communities with betweenness centrality
- Link prediction
Section outline

Degree centrality and prestige

Closeness centrality: definition and computation

Approximating closeness centrality
Analysis of social networks focuses on finding *important actors*, a.k.a. *central nodes*

Need a *formal concept of importance*

Depends on the applications or the focus of the study

*Centrality measure* (or centrality score):

$$ f : V \rightarrow \mathbb{R}^+ $$

The *higher* is $f(v)$, the *more central* is $v$. 

---
Degree centrality and prestige

Degree centrality: the degree of $v$, i.e., the number of neighbors of $v$:

$$f_{\text{deg}}(v) = |N(v)| = |\{(v, u) \in E\}|$$

It is a natural choice, given the preferential attachment model.

Prestige: how many followers do you have?

Solves shortcomings of degree on directed graphs:

only counts incoming edges:

$$f_{\text{pres}}(v) = \left| \{(u, v) \in E\} \right|$$

in-degree of $v$
Shortcomings

Degree centrality and prestige are *myopic* measures:

They only take into account the *immediate neighborhood*, not the *whole structure*

Figure from C. C. Aggarwal, *Data Mining — The Textbook*
Degree centrality and prestige

Closeness centrality: definition and computation

Approximating closeness centrality
What’s our intuition of the word *central*?
A node is central when the *average distance* to any other node else is small

$$\text{avgd}(v) = \frac{1}{n} \sum_{u \in V} d(v, u)$$

Is this a centrality measure?
No, because it is *high* when the node is *not central*, and low otherwise.

**Closeness centrality:**

$$\text{cc}(v) = \frac{1}{\text{avgd}(v)}$$

Only for *undirected connected* graphs
Closeness centrality

But on *preferential attachment graphs* ... nodes with *high degree* often have *high closeness*.
Computing closeness centrality

$$cc(v) = \frac{1}{\frac{1}{n} \sum_{u \in V} d(v, u)}$$

How to get $cc(v)$ for a single node $(v)$?

Run $BFS$ from $v$ to get all the $d(v, u)$, $u \in V$, then average, and take inverse.

Takes time $O(|E|)$.

How to compute $cc(v)$ for all nodes $v \in V$?

There’s no efficient algorithm: must run $BFS$ from each $v$.

Takes time $O(|V||E|)$.

“Efficient” has a different meaning when working with large data: more than quadratic is too slow. (Sometimes even quadratic)
Section outline

✓ Degree centrality and prestige

✓ Closeness centrality: definition and computation

Approximating closeness centrality
Approximating closeness centrality

Why computing *approximations* of the closeness centrality (CC) of *all* vertices?

1. The exact algorithm is *too expensive*;
2. The network *evolves*: the exact values of CC keep changing, so why chasing them?
3. No application requires the *exact* ranking of nodes w.r.t. closeness, and ranking is *almost preserved* by approximations.

Big-data tenet: Don’t do more work than *necessary for the application*. 
Approximating closeness centrality

Exact algorithm to compute the CC of all vertices:

**INPUT:** Undirected connected graph $G$

For each source $v \in V$:

- Run **BFS** from $v$ to obtain $d(v, u)$ for every $u \in V$
- Return $cc(v) = n / \sum_{u \in V} d(v, u)$

The distance $d(v, u)$ computed when running BFS from $v$ also contributes to $cc(u)$.

After $k$ iterations, we have $k$ terms of $\sum_{z \in V} d(u, z)$ for each $u$ that was not a source yet.

The average of these $k$ terms is an estimate for $\text{avgd}(u)$:

$\text{avgd}(u)$ is a population mean: estimate it with a sample mean.
Approximating closeness centrality

**INPUT:** Undirected connected graph \( G \), sample size \( k \)

For each \( v \in V \): \( S[v] \leftarrow 0 \) // To accumulate the \( k \) terms

For each \( i \leftarrow 1 \) to \( k \):

- Select a node \( v_i \) uniformly at random
- Run BFS from \( v_i \) to obtain \( d(v_i, u) \) for every \( u \in V \)
- For each \( u \in V \): \( S[u] \leftarrow S[u] + d(v_i, u) \) // Can be done while running BFS from \( v \)

Return \( \left( k/S[u] \right)_{u \in V} \) // Estimations of \( cc(u) = 1/\text{avgd}(u), u \in V \)

The *pseudocode is (often) a lie*: think carefully when you write your implementation
Approximating closeness centrality

**INPUT:** Undirected connected graph $G$, sample size $k$

**OUTPUT:** Estimate $\tilde{cc}(u)$ of $cc(v)$, for each $u \in V$

We want, with *high probability* that

$$|\tilde{cc}(u) - cc(u)| < \varepsilon \text{ for each } u \in V$$

for some $\varepsilon \in (0, 1)$

How *large should $k$ be* to achieve this (probabilistic) guarantee?

Depends on $1/\varepsilon^2$, $|V|$, and $\text{diam}(G)$

$\text{diam}(G)$ “controls” the *variance* of $\tilde{\text{avgd}}(v) = \frac{1}{k} \sum_{i=1}^{k} d(u, v_i)$
Degree centrality and prestige

Closeness centrality: definition and computation

Approximating closeness centrality
Outline

- Paths, shortest paths, diameter, and Breadth-First Search
- Social networks properties
- Degree centrality and prestige
- Closeness centrality
- Betweenness centrality
- Finding communities with betweenness centrality
- Link prediction
Section outline

Betweenness centrality: definition and properties

Computation

Approximating BC

Finding communities with betweenness centrality
Who is central/important?

1. HIGHEST DEGREE CENTRALITY
2. HIGHEST CLOSENESS CENTRALITY
What is betweenness centrality?

**Intuition:** Assume that
1. Every node sends a message to every other node; and
2. Messages follow *Shortest Paths (SPs).*

Then, *the more SPs go through \( v \in V \), the more important \( v \) is.*

**Def:** *Betweenness Centrality (BC):* \( b : V \rightarrow [0,1] : \)

\[
b(x) = \frac{1}{n(n-1)} \sum_{u \neq x \neq v \in V} \frac{\sigma_{uv}(x)}{\sigma_{uv}}
\]

- \( \sigma_{uv} \): number of SPs from \( u \) to \( v \), for \( u, v \in V \);
- \( \sigma_{uv}(x) \): number of SPs from \( u \) to \( v \) that go through \( x \).

\( b(x) \) measures *robustness* of communication to *deletion* of \( x \).
Example of BC

<table>
<thead>
<tr>
<th>Node $x$</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b(x)$</td>
<td>0</td>
<td>0.250</td>
<td>0.125</td>
<td>0.036</td>
<td>0.054</td>
<td>0.080</td>
<td>0.268</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of BC

HIGHEST DEGREE CENTRALITY

HIGHEST BETWEENNESS CENTRALITY

HIGHEST CLOSENESS CENTRALITY
Section outline

- Betweenness centrality

  Computation (Naïve Algo)

  Approximating BC

  Finding communities with betweenness centrality
Computing BC

\[ b(x) = \frac{1}{n(n-1)} \sum_{u \neq x \neq v \in V} \frac{\sigma_{uv}(x)}{\sigma_{uv}} \]

Q: Given \( u \in V \), how to compute \( \sigma_{uv} \) (no. SPs \( u \rightarrow v \)), for each \( v \in V \)?
A: Through BFS, with minor modifications.

Q: Given \( u, x \in V \), how to compute \( \sigma_{uv}(x) \) (no. SPs \( u \rightarrow v \) through \( x \)), for each \( v \in V \)?

Observations:
1. \( \sigma_{uv}(x) = \sigma_{ux} \times \sigma_{xv} \)
2. Recursive relation \( \sigma_{xv} = \sum_{w \in \text{Succ}_u(x)} \sigma_{wv} \) (base: \( \sigma_{wv} = 1 \) for \( w \in \text{Pred}_u(v) \))

A: run modified BFS then backtrack from \( v \) to \( u \) along reversed SP DAG.
Example

\[ \sigma_{uv}(x) = \sigma_{ux} \times \sigma_{xv}, \quad \sigma_{xv} = \sum_{w \in \text{Succ}_u(x)} \sigma_{uw} \quad (\text{base: } \sigma_{uw} = 1 \text{ for } w \in \text{Pred}_u(v)) \]

\[ u = 1, \ v = 8 \]

\[
\begin{array}{c|c|c|c}
 x & \sigma_{1x} & \sigma_{x8} & \sigma_{18}(x) \\
\hline
 1 & 0 & 3 & 0 \\
 2 & 1 & 11 & 1 \\
 3 & 1 & 122 & 2 \\
 4 & 1 & \text{N/A} & 0 \\
 5 & 1 & 11 & 1 \\
 6 & 1 & 11 & 1 \\
 7 & 1 & 11 & 1 \\
 8 & 3 & 0 & 0 \\
 9 & 3 & \text{N/A} & 0 \\
\end{array}
\]
Computing BC

**Algorithm**: Input: connected, undirected $G$,  Output: $b(x)$, for each $x \in V$

For each $u \in V$:

Run *modified BFS* to obtain SPs from $u$ to $v$ and $\sigma_{uv}$, $v \in V$;

For each $v \in V$:

*Backtrack* from $v$ towards $u$ to compute $\sigma_{uv}(x)$, $x \in V$;

For each $x \in V$:

Return $b(x) \leftarrow \frac{1}{n(n-1)} \sum_{u \neq x \neq v \in V} \frac{\sigma_{uv}(x)}{\sigma_{uv}}$;

**Time complexity**: $n$ (BFS + $n$ Backtracks) + $n$ (sum of $O(n^2)$ terms)

$$n(O(m) + n \times O(n)) + O(n^3) = O(n^3)$$

**Don’t implement / run this algorithm!**
A better algorithm for BC

**Time complexity:** \( n \) (BFS + \( n \) Backtracks) + \( n \) (sum of \( O(n^2) \) terms)

\[
n(O(m) + n \times O(n)) + O(n^3) = O(n^3)
\]

The sums at the end could be done *inside the loop*, like we saw for CC.

Costly step: after each BFS, we *backtrack through the graph n times*, each taking \( O(n) \).

Can we backtrack just *once per BFS*?

*Brandes’ algorithm* uses a recurrence relation to *backtrack once* per BFS, and computes *all* BCs in \( O(nm) \)

Social networks are *sparse*: \( m \approx O(n \log n) \), so \( O(nm) \) is much better than \( O(n^3) \).

Still not very efficient: there are *sampling*-based *approximation algorithms*. 
Section outline

 ✓ Betweenness centrality

 ✓ Computation (Naïve Algo)

 Approximating BC

 Finding communities with betweenness centrality
Approximating BC

\[ b(x) = \frac{1}{n(n-1)} \sum_{u \neq x \neq v \in V} \frac{\sigma_{uv}(x)}{\sigma_{uv}} \]

BC is “just” another population mean

The population is the set of pairs of distinct vertices:

\[ b(x) = \frac{1}{n(n-1)} \sum_{u \neq v \in V} \frac{\sigma_{uv}(x)}{\sigma_{uv}}, \quad \text{assuming } \frac{0}{0} = 0 \]

How can we approximate it? With a sample mean!

We can estimate BC efficiently from random sample of the population
Approximating BC

**INPUT:** Graph $G$, *sample size* $k$

For each $x \in V$, $S[x] \leftarrow 0$ // to store the partial sums

For each $i = 1 \ldots, k$:

- Select distinct $u, v$ uniformly at random
- Run **BFS** from $u$, *stopping* when we reach $v$ // get $\sigma_{uz}$ for each $z$ on a SP from $u$ to $v$
- **Backtrack** from $v$ towards $u$, // get $\sigma_{uv}(z)$

For each $z$ on a SP from $u$ to $v$: $S[z] \leftarrow S[z] + \sigma_{uv}(z)/\sigma_{uv}$ // done while backtracking

Return $(S[x]/k)_{x \in V}$
Approximating BC

**INPUT:** Graph $G$, *sample size* $k$

**OUTPUT:** Estimate $\tilde{b}(u)$ of $b(v)$, for each $u \in V$

We want, with *high probability* that

$$|\tilde{b}(u) - b(u)| < \varepsilon$$

for each $u \in V$ for some $\varepsilon \in (0, 1)$

How large should $k$ be to achieve this (probabilistic) guarantee?

Depends on $1/\varepsilon^2$, $|V|$, and $\text{diam}(G)$

$\text{diam}(G)$ “controls” the *variance* of $\tilde{b}(v)$ (in a different way than it did for $\tilde{cc}(v)$)
Recap on centrality measures

Many ways of formalizing the *importance* of a node in a graph:

Based on degree, SP distances, and number of SPs, and more!

Centrality scores are often *inefficient to compute*

Can be expressed as *population means*, estimated from *random sample*

The problem of *how many samples* are really needed is still open
Section outline

- Betweenness centrality
- Computation (Naïve Algo)
- Approximating BC

Finding communities with betweenness centrality
Finding communities

A community $A$ is a subset of $V$ such that the density

$$\text{density}(A) = \frac{|\{(u, v) \in E : u, v \in A\}|}{\binom{|A|}{2}}$$

is high.

Just one of the many definitions of what a community is.

Community detection: find a partitioning of $V$ into equivalence classes $C_1, C_2, \ldots, C_k$ s.t.

$$C_i \cap C_j = \emptyset \text{ for } i \neq j, \quad \text{and} \quad \bigcup_{i=1}^{k} C_i = V$$

Essentially a form of clustering

A.k.a. Graph summarization ← predates social network analysis
Finding communities

*Community detection:* find a *partitioning* of $V$ into *equivalence classes* $C_1, C_2, \ldots, C_k$ s.t.

$$C_i \cap C_j = \emptyset \text{ for } i \neq j, \quad \text{and} \quad \bigcup_{i=1}^{k} C_i = V$$

Some partitionings are *better* than others (i.e., the classes “look more like communities”)

We must define the *cost* of a partitioning

$$q(C_1, \ldots, C_k) \in \mathbb{R}^+$$

*Lower* cost = *better* partitioning

(Possible) cost: number of *edges with extremes in different classes.*
Finding communities

Challenges in community detection due to the structure of social networks:

All node distances are small: distance-based clustering methods do not work.

High-degree nodes connect multiple communities: to which community do they belong? (figure)

The giant component is often dense, but it can’t be a community by itself
(we usually want balanced communities of roughly the same size)
Hubs and communities

(Hubs = high degree nodes)

Figure from C. Aggarwal — Data Mining, The Textbook, p. 628
(Possible) cost: number of *edges with extremes in different classes*.

A *cut* is a set of edges such that their removal *increases* the no. of *connected components*.

The set of edges from *one community to (all) the others* is a cut.

So maybe we want to find *small cuts*.
Finding communities

We want to find small cuts to identify communities.

The *minimum 2-way cut* can be found in $O(n^2 \log^c n)$ (Karger’s algorithm).

Run Krager’s algo $k - 1$ times to find a partitioning in $k$ classes.

The obtained classes are very *un-balanced*, not good for community detection.

Modifications exist to find more balanced partitions.
Finding communities with betweenness centrality

Different way of thinking about *what edges to remove* to find communities

I.e., *partition* the vertices

If an *edge* has *high betweenness*, then the SPs of many pairs of nodes *go through* that edge.

Idea: remove edges with high BC, in decreasing order, until we get $k$ classes
Finding communities with betweenness centrality

**Girvan-Newmann algorithm** for finding communities

**INPUT**: graph $G$, number of classes $k$

Repeat

- Compute the BC of each edge $e \in E$
- Remove edge $e^*$ with highest BC

until $G$ has $k$ connected components

Create a community for the vertices in each connected component

Recomputing the edge BCs is expensive but does not have to be done from scratch

There are *incremental* Shortest Path algorithms.
Finding communities — Recap

There are many ways of *defining* what a community is.

We can define a *cost* function to express how well a partitioning identifies communities.

We can find the partitioning that (approximately) *minimizes the cost* or *go with our intuition* of what edges to remove.

There are many other methods, and still a lot of research (e.g., *overlapping communities*)
Section outline

✓ Betweenness centrality

✓ Computation (Naïve Algo)

✓ Approximating BC

✓ Finding communities with betweenness centrality
Outline

- Paths, shortest paths, diameter, and Breadth-First Search
- Social networks properties
- Degree centrality and prestige
- Closeness centrality
- Betweenness centrality
- Finding communities with betweenness centrality

Link prediction
Link prediction

Social networks evolve over time

Natural prediction task: what edge is likely to appear next?

Why: (commercial) online social network need more and more connections between users

To increase the number of connection they recommend new friends
Recommending friends

*Structural* approach: rely on *triadic closure*:
  two nodes with many *common neighbors* are likely to be(come) friends.
  Works well on all domains, pretty *effective for predictions*.
  Why: Triadic closure is a graph property, and we want to predict edges

*Content* approach: rely on *homophily*:
  two nodes with similar *features* are likely to be(come) friends.
  Only works on some networks from some domains, very sensitive to noise.
  Why: lack of homophily, “crossing” domains in using it to predict edges.
Neighborhood-based measures

Structural approach: rely on triadic closure.

Idea: use the number of Common Neighbors (CN) between nodes \( u \) and \( v \) to quantify the likelihood of an edge between them (in the future).

Different measures, as the number of common neighbors can be used in different ways:

- Common neighbors measure
- Jaccard measure
- Adamic-Adar measure

The higher the measure for \((u, v)\), the higher the likelihood of the edge \((u, v)\) to appear.
Common neighbors measure

It is what the name implies:

\[ \text{cn}(u, v) = |N(u) \cap N(v)| \]

\[ N(\text{Alice}) = \{\text{John, Jack, Jill, Mary, Sayani, Jim, Micheal}\} \]
\[ N(\text{Bob}) = \{\text{John, Jack, Jill, Mary, Nicole, Peter}\} \]
\[ N(\text{Alice}) \cap N(\text{Bob}) = \{\text{John, Jack, Jill, Mary}\} \]
\[ \text{cn}(\text{Alice, Bob}) = 4 \]

Weakness of \text{cn}: does not account for the \textit{relative} no. of CN vs. the no. of \textit{all} neighbors.
Jaccard measure

Fraction of CNs over the number of all neighbors:

\[ j(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \]

N(Alice) = \{John, Jack, Jill, Mary, Sayani, Jim, Micheal\}
N(Bob) = \{John, Jack, Jill, Mary, Nicole, Peter\}
N(Alice) \cap N(Bob) = \{John, Jack, Jill, Mary\}
N(Alice) \cup N(Bob) = \text{(everyone but Alice and Bob)}

\[ j(\text{Alice, Bob}) = \frac{4}{9} \]

Weakness of \( j \): does not account for the varying degree of CNs.
Adamic-Adar measure

Idea: make high-degree CNs have less impact on edge likelihood than low-degree CNs.

\[ \text{aa}(u, v) = \sum_{w \in N(u) \cap N(v)} \frac{1}{\log |N(w)|} \]

- \( N(Alice) = \{John, Jack, Jill, Mary, Sayani, Jim, Micheal\} \)
- \( N(Bob) = \{John, Jack, Jill, Mary, Nicole, Peter\} \)
- \( N(Alice) \cap N(Bob) = \{John, Jack, Jill, Mary\} \)
- \( \text{aa}(Alice, Bob) = \frac{1}{\log 2} + \frac{1}{\log 4} + \frac{1}{\log 2} + \frac{1}{\log 4} = \frac{3}{\log 2} \)
Outline

- Paths, shortest paths, diameter, and Breadth-First Search
- Social networks properties
- Degree centrality and prestige
- Closeness centrality
- Betweenness centrality
- Finding communities with betweenness centrality
- Link prediction
Beyond what we covered

Other models of network evolution

Axioms of centrality

Link prediction as a classification problem

Social influence analysis

Knowledge graphs