INTRODUCTION

Network Robustness – the network’s structure plays a role in its ability to survive
- Random failures
- Deliberate attacks
MOTIVATIONS

Biology: some mutations lead to diseases while others do not

Ecology: failure of an ecosystem based on human activity

Engineering: communication systems, power grids, and component failures
VULNERABILITY METRICS

Centrality Metrics

Degree
Closeness
Centroid
Eccentricity
Betweenness
Eigenvector

Robustness Measure

Average path length
Efficiency (power grids)
Largest connected component
PSEUDOCODE — BRANDES ALGORITHM

Algorithm 1: Betweenness centrality in unweighted graphs

\[ C_B[v] \leftarrow 0, \; v \in V; \]
for \( s \in V \) do
\[ S \leftarrow \text{empty stack}; \]
\[ P[w] \leftarrow \text{empty list}, \; w \in V; \]
\[ \sigma[t] \leftarrow 0, \; t \in V; \; \sigma[s] \leftarrow 1; \]
\[ d[t] \leftarrow -1, \; t \in V; \; d[s] \leftarrow 0; \]
\[ Q \leftarrow \text{empty queue}; \]
\[ \text{enqueue } s \rightarrow Q; \]
while \( Q \) not empty do
\[ \text{dequeue } v \rightarrow Q; \]
\[ \text{push } v \rightarrow S; \]
\[ \text{foreach neighbor } w \text{ of } v \text{ do} \]
\[ \text{// } w \text{ found for the first time?} \]
\[ \text{if } d[w] < 0 \text{ then} \]
\[ \text{enqueue } w \rightarrow Q; \]
\[ d[w] \leftarrow d[v] + 1; \]
end
\[ \text{// shortest path to } w \text{ via } v? \]
\[ \text{if } d[w] = d[v] + 1 \text{ then} \]
\[ \sigma[w] \leftarrow \sigma[w] + \sigma[v]; \]
append \( v \rightarrow P[w]; \]
end
end
\[ \delta[v] \leftarrow 0, \; v \in V; \]
\[ \text{// } S \text{ returns vertices in order of non-increasing distance from } s \]
while \( S \) not empty do
\[ \text{pop } w \leftarrow S; \]
for \( v \in P[w] \) do
\[ \delta[v] \leftarrow \delta[v] + \frac{\sigma[w]}{\sigma[w]} \cdot (1 + \delta[w]); \]
end
if \( w \neq s \) then
\[ C_B[w] \leftarrow C_B[w] + \delta[v]; \]
end

Uses BFS for unweighted graphs

Runtime: \( O(m \times n) \) on unweighted

Space: \( O(m+n) \)

Weighted networks:

Runtime: \( O(m \times n + n^2 \log(n)) \)
LARGEST CONNECTED COMPONENT

Algorithm 2: Largest Connected Component

Data: Graph $G = (V, E)$
Result: Largest Connected Component of unweighted graphs

$Vis[v] \leftarrow 0, v \in V$;

for $v \in V$ do
  if $Vis[v] == 0$ then
    $DFSuntil(v)$;
  end
end

Data: Node $v$, $Vis[]$, Graph $G = (V, E)$

$v \leftarrow 1$ for $u$ adjacent to $v$ do
  if $Vis[u] == 0$ then
    $DFSuntil(u, Vis[], G)$;
  end
end

Can be performed with either BFS or DFS

Runtime: $O(m+n)$
BETWEENNESS CENTRALITY — ROBUSTNESS

Pipeline
- Calculate Betweenness Centrality
- Remove nodes/edges
- Measure robustness
- Compare with initial measure

IB removal | RB removal
SEQUENTIAL IMPLEMENTATION

Python
- Networkx
- Uses Brandes Algorithm
SCALE-FREE MODEL

**SF-BA Lcc**

- **Node Removed**
  - X-axis: Node Removed (0 to 30)
  - Y-axis: Lcc (0 to 1)

- **SF Betweenness Average Time**
  - **Average Time (s)**
    - X-axis: Number of Nodes (100 to 3200)
    - Y-axis: Average Time (0.015625 to 64)

**Legend**
- Green line: 100
- Light green line: 200
- Yellow line: 400
- Orange line: 800
- Light blue line: 1600
- Blue line: 3200
SMALL WORLD MODEL

![Graph 1: SW Lcc](image1)

![Graph 2: SW Betweenness Average Time](image2)
Complete Graph Model

Complete Graph Lcc

Complete Graph Betweenness Average Time
RESULTS

5000 nodes
6500 edges
90-Percentile Diameter: 28.17
Average Shortest Path: 20.09
FUTURE WORK

- Parallelize
- Streaming calculation
- Comparison of IB to RB
- Introduce cascading failure
QUESTIONS/COMMENTS
BACKUP SLIDES