CS 6604: Data Mining Large Networks and Time-series

B. Aditya Prakash
Lecture #2: More Graph Properties and the Web-graph
Networks with attributes

- Edges and Nodes can have attributes
- **Weighted graphs**
  - Numerical Attribute on edges
  - Can be negative! (trust vs distrust, transcription regulatory networks)
  - E.g. frequency of communication on a phone call graph
- Other types of attributes
  - Ranking (bff, second bff, ...)
  - Type (relative, co-worker, ...)
  - More aggregate global properties (centralities etc.)
Adjacency Matrices

- Representing edges (who is adjacent to whom) as a matrix \( \rightarrow \) [matrix algebra!]

\[
A_{ij} = 1 \text{ if node } i \text{ has an edge to node } j \\
A_{ij} = 0 \text{ otherwise}
\]

\[
\begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}
\]

A is symmetric for an undirected graph
Other ways

- **Incidence Matrix**
  - $b_{ij} = 1$ if vertex $i$ and edge $j$ are incident

- **Adjacency lists**
  - “edge lists”
  - $\{(2,3), (2,5) \ldots \}$
  - More efficient if network is sparse and large
Bipartite Networks

- Two-mode networks
  - Edges occur only across groups
  - The red and blue nodes are ‘independent sets’
  - E.g. people-to-places they visit

- Can be folded into ‘one-mode’ networks
  - People who visit the same places
Networks

- DBLP: bi-partite network author-papers folded author-author collaboration ....
- Twitter follower-followee: directed weighted
- Facebook friendship: undirected, unweighted
- Mobile phone calls: directed, weighted
- Protein-protein interactions: undirected, unweighted with self-interactions
A CASE STUDY: THE WEB-GRA PH

(based on some slides by Prof. Jure Leskovec)
The Web graph

**Question:** What is the structure of the web-graph?

First have to answer the following questions:

- What is the web?
- How to represent the web as a graph?
- How to find the structure?
What is the web?

- All web-pages
- Should we include pages dynamically created?
  - What about the ones from a database?
How to represent the web graph?

- Nodes: webpages
- Edges: hyperlinks

- It is a directed graph [Broder et. al. 2000]
How to find the structure?

- Any directed graph can be expressed using two concepts:
  - Strongly Connected Components (SCC) (maximal strongly connected sets)
  - Directed Acyclic Graphs (DAG) (directed graphs with no cycles in them)
How to find the structure?

- Which nodes are **reachable from** a given node \( v \)?
- Reverse question: which nodes can **reach** \( v \)

Out(B) = \{B, C, E, D, F\}

In(B) = \{B, A, D, E, C\}
Quick: A Directed Graph

- is **Strongly Connected** if
  - For all nodes $v$, $\text{In}(v) = \text{Out}(v) = \text{the whole set}$

- is **Directed Acyclic** if
  - For any pair of nodes $u$ and $v$,
    - if $u \in \text{In}(v) \rightarrow u \notin \text{Out}(v)$

$(\varepsilon == \text{element of})$
Strongly Connected Component

- Is one which is
  - Strongly connected
  - S is not part of a bigger strongly connected set
SCCs Properties

1. Every node belongs to exactly one SCC \([\text{easy: why?}]\)

2. The Graph of SCCs (where each SCC is a node) is a DAG (a skeleton of the graph) \([\text{little harder}]\)

---

Fact: Every directed graph is a DAG on its SCCs

(1) SCCs partitions the nodes of \(G\)

(2) If we build a graph \(G'\) whose nodes are SCCs, and with an edge between nodes of \(G'\) if there is an edge between corresponding SCCs in \(G\), then \(G'\) is a DAG

---

Q: Is \(S_3 = S_1 \cup S_2\) a SCC?

A: YES!

Proved.
Proof: Graph of SCCs is a DAG

- If not, then there is a cycle
- But all nodes in the cycle can reach each other
  – They should be in the same SCC!
- Contradiction
How to find the structure?

- **Answer**: Understand the DAG of SCCs of the web-graph

- **Computational issue**: How to find the SCC containing node $v$?
  - SCC containing $v = \text{In}(v) \cap \text{Out}(v)$
  - E.g.
    
    
    
    
    
    
    
    
    
    
    
    
    SCC (B) = \text{In}(B) \cap \text{Out}(B) = \{B, C, D, E\}
  
  - Follows from definitions (try to prove it!)
Two Giant SCCs?

- No i.e. \( \Pr[\text{Two SCCs}] \ll \text{small} \)

- Why?
  - It is low effort to link two pages
  - So two giant sets with millions of pages not linking each other at all is improbable
Web-graph structure (Broder et al)

- Altavista web crawl from Oct. 1999
  - 203 Million URLs
  - 1.5 Billion Links
- Computer: server with 12GB of memory
- See how the SCCs ‘fit together’
Web-graph structure (Broder et al)

**Finding 1:** 91% of nodes in the largest weakly connected component

**Finding 2:** Giant SCC has 28% of the nodes (56M)

**Finding 3:** $\text{Sizeof}(\text{Out}(v)) \approx \text{Sizeof}(\text{In}(v)) \approx 50\%$ (100M) for a random node $v$
Conclusions?

- Bow-Tie structure of the web [Broder et al. 2000]
Bow-tie

- Conceptual Organization

- VERY high level view though!
  - What is the internal structure of the Giant SCC?
  - What are the most important pages?
  
  ....
NETWORK FEATURES
How to characterize a graph?

- Think of them as network ‘features’
Degree Distribution

- Degree $k_i = \text{the number (or the total weight) of edges incident to node } i$

- For directed networks: in-degree and out-degree

- More terms: source node has in-degree 0, sink node has out-degree 0
Degree Distribution $P[k]$

- $P[k] = \text{probability that a randomly chosen node has degree } k$
- Empirically,

\[ P[k] = \frac{N_k}{N} \]

$(N_k = \text{the total number of degree } k \text{ nodes})$
A **path** is a sequence of nodes $n_i$ where each pair $n_i n_{i+1}$ have an edge.

**IMPORTANT:** a path can intersect itself, pass through same edge multiple times

- E.g. $AB$ is a path
  - $ABCEDA$ is a path
  - $ABA$ is **NOT** a path
Matrix Method: Number of paths between u and v

- **Length 1:** if $A_{uv} = 1$ then 1, otherwise 0
- **Length 2:** if $A_{uk}A_{kv} = 1$, then there is a path from u to v via k. so $P^{(2)}_{uv} = \sum A_{uk}A_{kv} = [A^2]_{uv}$

- ..... 
- ..... 
- **Length l:** $P^{(l)}_{uv} = [A^l]_{uv}$
Distance in a graph

- \( \text{Distance}(u, v) = \# \text{ nodes on the shortest path connecting } u \text{ and } v \)
  
  - if no path between \( u \) and \( v \), then \( \text{dist}(u,v) = \infty \)

\[
\begin{align*}
\text{Dist}(B, C) &= 1 \\
\text{Dist}(C, B) &= 3 \\
\text{Note: in directed graphs, dist(.) is NOT symmetric}
\end{align*}
\]
How far are the nodes?

- **Diameter:** Longest *shortest* path between *any* u and v

\[
d = \max_{u,v \neq u} \text{dist}(u,v)
\]

- **Average path length:** (typically done for connected components)

\[
\bar{d} = \frac{1}{\# \text{pairs}} \sum_{u,v \neq u} \text{dist}(u,v)
\]

\[
\# \text{pairs} = \binom{N}{2} = \frac{N(N-1)}{2}
\]
Finding shortest paths

- **Classic** Computer Science problem
- Breadth-First Search
- Dijkstra’s algorithm
- Bellman-ford algorithm
- Floyd-Warshall algorithm
- ............

In any CS algorithms textbook
Clustering Co-efficient $C_i$

- Is a friend of my friend, my friend?
- $C_i = \text{proportion of node } i\text{'s neighbors who are connected}$

$$C_i = \frac{2e_i}{k_i(k_i-1)}$$

- $e_i$ is the number of edges between the neighbors of node $i$
- Similarly we can define, average $C_i$, max $C_i$, etc.
Practice Question

- Get the clustering coefficient for node $i$ from the matrix method of finding #paths?
Summary: Network Properties

- Key properties:
  - Degree Distribution: $P[k]$
  - Path length: $l$
  - Clustering Co-efficient: $C_i$

- Useful to characterize a network with

- Many others: e.g. largest eigenvalue of the adjacency matrix, the laplacian eigen-gap etc. etc. (we will see some of these later!)
Example: Chain and 1-d lattice

- $P[k] = ?$
- Path length: $l_{\text{max}} = ?, l_{\text{avg}} = ?$
- Clustering Co-efficient = ?
- What if $N \rightarrow \infty$ ?