Network Algorithms

Michael Goodrich

Some slides adapted from:
Networked Life (NETS) 112, Univ. of Penn., 2018, Prof. Michael Kearns
Determining the Diameter of Small World Networks, Frank W. Takes & Walter A. Kosters, Leiden University, The Netherlands
Structure and models of real-world graphs and networks, Jure Leskovec, Carnegie Mellon University
Complex (Biological) Networks, by Elhanan Borenstein, Roded Sharan, and Tomer Shlomi
Computational Representation of Networks

- Which is the most useful representation?
- Should you use them in combination?

**List or hash table of edges:** (ordered) pairs of nodes

\{(A,C), (C,B), (D,B), (D,C)\}

**Adjacency Matrix**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Adjacency List**

- Name: D
  - ngr:
    - p1
    - p2

- Name: C
  - ngr:
    - p1

- Name: B
  - ngr:

- Name: A
  - ngr:
Network Structures

• Network structures characterize how networks “look”:
  – Large or small diameter?
  – Number of edges: sparse or dense?
  – Degree distributions: heavy/long tail with a power law?
  – Clustering coefficient: high or low?

• These are empirical phenomena

• How do you compute them?

Image from https://matrix.berkeley.edu/research/social-networks-history
Degree Distribution

- x axis: number of neighbors (degree)
- y axis: number of vertices with that degree

A long tail (also known as a “heavy tail”)
Degree Distribution Algorithm

1. Compute the degree, \( \text{deg}(v) \), of each vertex, \( v \).
   - If \( G \) is represented as an adjacency list, count the number of elements in \( v \)'s list.
2. Create a histogram count array, \( H \), of size \( n \), and initialize each \( H[i] = 0 \).
3. For each vertex, \( v \), increment \( H[\text{deg}(v)] \).
4. Plot the values of \( H \) from 0 to \( n-1 \) on a regular and log-log scale.
5. If the values on the log-log plot form a straight line, determine its slope to find the exponent of the power law degree distribution.
Example 1

- Degree distribution without a long/heavy tail.
- Does not exhibit a power law.
Example 2

- Degree distribution with a long/heavy tail.
- Does exhibit a power law, with exponent $-2.5$. 

Slope = $-2.5$
Distance

• The **distance** between two vertices is the length of the shortest path connecting them.
  – This assumes the network has only a single connected component
  – If two vertices are in different components, their distance is infinite

*Image from [https://www.sci.unich.it/~francesc/teaching/network/geodesic.html](https://www.sci.unich.it/~francesc/teaching/network/geodesic.html)*
Diameter

- The **diameter** of a network is the maximum distance between a pair of vertices in the network.
  - It measures how near or far typical individuals are from each other.

*The dolphin network with the diameter (the longest shortest path) highlighted in red. The diameter is 8 edges long.*

*From [https://users.dimi.uniud.it/~massimo.franceschet/bottlenose/bottlenose.html](https://users.dimi.uniud.it/~massimo.franceschet/bottlenose/bottlenose.html)*
Definitions

- Consider a connected undirected graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges

- **Distance** $d(v, w)$: length of shortest path between nodes $v, w \in V$

- **Diameter** $D(G)$: maximal distance (longest shortest path length) over all node pairs: $\max_{v, w \in V} d(v, w)$

- **Eccentricity** $e(v)$: length of a longest shortest path from $v$: $e(v) = \max_{w \in V} d(v, w)$

- **Diameter** $D(G)$ (alternative definition): maximal eccentricity over all nodes: $\max_{v \in V} e(v)$

- Eccentricity distribution: (relative) frequency $f(x)$ of each eccentricity value $x$

$$f(x) = \frac{|\{u \in V \mid e(u) = x\}|}{n}$$
Example

- A graph with diameter 6
- Numbers next to nodes denote eccentricity values
Naïve Algorithm

- **Diameter** is equal to the largest value returned by an All Pairs Shortest Path (APSP) algorithm.
- Brute-force: for each vertex \( v \), execute a Breadth First Search (BFS) from \( v \) in \( O(m) \) time to find \( v \)'s eccentricity. Return the largest value found.
- Time complexity \( O(nm) \)
- Problematic if \( n = 8 \) million and \( m = 1 \) billion.
  - If one BFS takes 6 seconds on a 3.4GHz machine, this brute-force algorithm takes 1.5 years to compute the diameter . . .
Heuristic Idea 1

• If we can find one of the nodes in a diameter pair, we can compute the diameter with one more BFS.

1. Perform a BFS from a random sample of nodes, recording nodes with maximum found distance, d.
2. Perform a BFS from all the far nodes (if small) or a random sample of this set (if large).
Heuristic Idea 2

1. Let $r$ be a random vertex and set $D_{\text{max}} = 0$.
2. Perform a BFS from $r$.
3. Select the farthest node, $w$, in this BFS.
   - If the distance from $r$ to $w$ is larger than $D_{\text{max}}$, set $D_{\text{max}}$ to this distance, let $r = w$, and repeat the above two steps.
Plot Results as a Function of $n$

- If the networks exhibit the **small world** phenomenon, then diameters are small.
- So plot diameters as a function of $n$ on a lin-log scale:

![The log n function looks like a straight line](LinLogScale.png)
Clustering Coefficient

- “friend of a friend is a friend”
- If $a$ connects to $b$, and $b$ to $c$, then with high probability $a$ connects to $c$.

- Clustering coefficient $C$:
  \[ C = \frac{3 \times \text{number of triangles}}{\text{number of 2-edge paths}} \]

\[ C = \frac{3 \times 1}{1 + 1 + 6 + 0 + 0} = \frac{3}{8} = 0.375 \]
Clustering Coefficient (2)

- Clustering coefficient might have a power law:
  \[ C(k) \sim k^{-1} \]

- It is speculated that in real networks:
  \[ C = O(1) \text{ as } n \rightarrow \infty \]
Clustering Coefficient Algorithm

- Clustering coefficient $C$:
  \[ C = \frac{3 \times \text{number of triangles}}{\text{number of 2-edge paths}} \]
- Computing the denominator is easy:
  - For each vertex $v$, let $\text{deg}(v)$ denote its degree.
  - The number of paths of length 2 with $v$ in the middle is $\text{deg}(v) \choose 2 = \text{deg}(v)(\text{deg}(v)-1)/2$.
  - So, to get the denominator for $C$, sum up $\text{deg}(v)(\text{deg}(v)-1)/2$ for all vertices, $v$, in $G$.

Number of 2-edge paths with $v$ in the middle is $4(3)/2 = 6$. 
Counting Triangles

• To get the numerator for $C$, we need to count the number of triangles in the graph, $G$.

• Naïve algorithm:
  – For every triple, $u$, $v$, $w$ in $G$, see if they form a triangle. If so, add 1 to a running count.
  – Running time is $O(n^4)$ if $G$ is represented with an adjacency list.

This is bad.
Counting Triangles: Slight Improvement

• Put every edge, \((v,w)\), into a hash table, \(T\), so we can do a lookup to see if an edge exists in \(O(1)\) expected time, i.e., with a \(\text{get}((v,w))\).

• Slightly better naïve algorithm:
  – For every triple, \(u, v, w\) in \(G\), see if they form a triangle. If so, add 1 to a running count.
  – Running time is \(O(n^3)\) expected if edges in \(G\) are stored in a hash table.

This is still bad.
Graph Degeneracy

- The **degeneracy** of a graph is the smallest value of $d$ for which every subgraph has a vertex of degree at most $d$.
- If a graph has degeneracy $d$, then there exists an ordering of the vertices of $G$ in which each vertex has at most $d$ neighbors that are earlier in the ordering.

An ordering for a graph with degeneracy 2:
Real-World Graphs

- Real-world graphs tend to have small degeneracy, $d$.

<table>
<thead>
<tr>
<th>graph</th>
<th>$n$</th>
<th>$m$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>zachary [48]</td>
<td>34</td>
<td>78</td>
<td>4</td>
</tr>
<tr>
<td>dolphins [35]</td>
<td>62</td>
<td>159</td>
<td>4</td>
</tr>
<tr>
<td>power [47]</td>
<td>4,941</td>
<td>6,594</td>
<td>5</td>
</tr>
<tr>
<td>polbooks [28]</td>
<td>105</td>
<td>441</td>
<td>6</td>
</tr>
<tr>
<td>adjnoun [29]</td>
<td>112</td>
<td>425</td>
<td>6</td>
</tr>
<tr>
<td>football [15]</td>
<td>115</td>
<td>613</td>
<td>8</td>
</tr>
<tr>
<td>lesmis [25]</td>
<td>77</td>
<td>254</td>
<td>9</td>
</tr>
<tr>
<td>celegensneural [47]</td>
<td>297</td>
<td>1,248</td>
<td>9</td>
</tr>
<tr>
<td>netscience [39]</td>
<td>1,589</td>
<td>2,742</td>
<td>19</td>
</tr>
<tr>
<td>internet [40]</td>
<td>22,963</td>
<td>48,421</td>
<td>25</td>
</tr>
<tr>
<td>condmat-2005 [38]</td>
<td>40,421</td>
<td>175,693</td>
<td>29</td>
</tr>
<tr>
<td>polblogs [4]</td>
<td>1,490</td>
<td>16,715</td>
<td>36</td>
</tr>
<tr>
<td>astro-ph [38]</td>
<td>16,706</td>
<td>121,251</td>
<td>56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>graph</th>
<th>$n$</th>
<th>$m$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>roadNet-CA [34]</td>
<td>1,965,206</td>
<td>2,766,607</td>
<td>3</td>
</tr>
<tr>
<td>roadNet-PA [34]</td>
<td>1,088,092</td>
<td>1,541,898</td>
<td>3</td>
</tr>
<tr>
<td>roadNet-TX [34]</td>
<td>1,379,917</td>
<td>1,921,660</td>
<td>3</td>
</tr>
<tr>
<td>amazon0601 [30]</td>
<td>403,394</td>
<td>2,443,408</td>
<td>10</td>
</tr>
<tr>
<td>email-EuAll [31]</td>
<td>265,214</td>
<td>364,481</td>
<td>37</td>
</tr>
<tr>
<td>email-Enron [24]</td>
<td>36,692</td>
<td>183,831</td>
<td>43</td>
</tr>
<tr>
<td>web-Google [2]</td>
<td>875,713</td>
<td>4,322,051</td>
<td>44</td>
</tr>
<tr>
<td>soc-wiki-Vote [33]</td>
<td>7,115</td>
<td>100,762</td>
<td>53</td>
</tr>
<tr>
<td>soc-slashdot0902 [34]</td>
<td>82,168</td>
<td>504,230</td>
<td>55</td>
</tr>
<tr>
<td>cit-Patents [18]</td>
<td>3,774,768</td>
<td>16,518,947</td>
<td>64</td>
</tr>
<tr>
<td>soc-Epinions1 [42]</td>
<td>75,888</td>
<td>405,740</td>
<td>67</td>
</tr>
<tr>
<td>soc-wiki-Talk [33]</td>
<td>2,394,385</td>
<td>4,659,565</td>
<td>131</td>
</tr>
<tr>
<td>web-berkstean [34]</td>
<td>685,231</td>
<td>6,649,470</td>
<td>201</td>
</tr>
</tbody>
</table>
Degeneracy Ordering Algorithm

• Degeneracy can be computed by a simple greedy algorithm:
  – Repeatedly find and remove the vertex of smallest degree, adding it to the end of the list.
  – The degeneracy is then the highest degree, \( d \), of any vertex at the moment it is removed.
  – The ordering is a \( d \)-degeneracy ordering.
Linear-time Implementation

1. Initialize an output list, $L$, to be empty.
2. Compute a number, $d_v$, for each vertex $v$ in $G$, which is the number of neighbors of $v$ that are not already in $L$. Initially, $d_v$ is just the degrees of $v$.
3. Initialize an array $D$ such that $D[i]$ contains a list of the vertices $v$ that are not already in $L$ for which $d_v = i$.
4. Let $N_v$ be a list of the neighbors of $v$ that come before $v$ in $L$. Initially, $N_v$ is empty for every vertex $v$.
5. Initialize $k$ to 0.
6. Repeat $n$ times:
   - Let $i$ be the smallest index such that $D[i]$ is nonempty.
   - Set $k$ to $\max(k, i)$.
   - Select a vertex $v$ from $D[i]$. Add $v$ to the beginning of $L$ and remove it from $D[i]$. Mark $v$ as being in $L$ (e.g., using a hash table, $H_L$).
   - For each neighbor $w$ of $v$ not already in $L$ (you can check this using $H_L$):
     • Subtract one from $d_w$
     • Move $w$ to the cell of $D$ corresponding to the new value of $d_w$, i.e., $D[d_w]$
     • Add $w$ to $N_v$
Triangle Counting Algorithm

• Compute a \(d\)-degeneracy ordering of the vertices, e.g., using the algorithm of the previous slide.

• Process the vertices according to this ordering, \(L\):
  
  For each vertex, \(v\):
  
  For each pair of vertices, \(u\) and \(w\), adjacent to \(v\) and earlier in the ordering, i.e., \(u\) and \(v\) are in the list \(N_v\) from the degeneracy algorithm:

  If \((u,w)\) is an edge in the graph, then add one to the triangle count.

• Running time is \(O(d^2n) = O(dm)\) expected, assuming edges are stored in a hash table.