HW1 Discussion
Visualizing Generated Networks

<Pic.1-1 ER Model with n = 30, p = 0.3
<Pic.1-2 ER Model with n=30, m=50>
<Pic.1-3 BA Model with m0=3, m=2, n=30>
<Pic.1-2 CG Model with P(k)~k^-alpha, alpha=1.5, n=30>
<Pic.1-4. SW Model with n=30, k=6, p=0.3>

嚴恩勗, 黃適文
上圖為 $G^c(p=2.8*10^{-6})$ 示意圖

上圖為 $G^c(m=1000)$ 示意圖

上圖為 Barabasi-Albert Model ($m=2$ $n=1000$) 示意圖

上圖為 Watts-Strogatz Model ($P=0.0001$) 的示意圖
Common Strategy to Compute APL

• Find Giant Connected Component, and then apply Breadth-First Search from each node
• Compute all-pairs shortest paths by Floyd-Warshall Algorithm
• Random sampling several nodes, and then apply shortest path algorithm
# Theoretical Values

<table>
<thead>
<tr>
<th></th>
<th>Z1</th>
<th>Z2</th>
<th>APL</th>
<th>CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ER(n,p)</td>
<td>Np</td>
<td>(np)^2</td>
<td>ln(n)/ln(z₁)</td>
<td>P</td>
</tr>
<tr>
<td>ER(n,m)</td>
<td>2m/n</td>
<td>(2m/n)^2</td>
<td>ln(n)/ln(z₁)</td>
<td>2m/n*(n-1)</td>
</tr>
<tr>
<td>CF(n,a)</td>
<td>&lt;k&gt;</td>
<td>&lt;k^2&gt; - &lt;k&gt;</td>
<td>ln(n / z₁) + 1 / ln(z₂ / z₁)</td>
<td>(z₂ / z₁)^2 / nz₁</td>
</tr>
<tr>
<td>BA(n,m₀,m)</td>
<td>2m(n-m₀)/n</td>
<td>2m²(N-m₀)ln(N/m)/(2N-z₁)</td>
<td>~ log(n)</td>
<td>N^-.75</td>
</tr>
<tr>
<td>WS(n,k,p)</td>
<td>K</td>
<td></td>
<td>1/p tanh⁻¹ [1 / \sqrt{1 + 2/pL}]</td>
<td>3(k-2)^*(1-p)^3 / [4(k-1)^3]</td>
</tr>
</tbody>
</table>

---

5
### HW1.3 graph_cm1

<table>
<thead>
<tr>
<th>Rank</th>
<th>Authors</th>
<th>Score</th>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>郭英樹, 林瑋詩, 何伯樟</td>
<td>0.2</td>
<td>1. Randomly split to 2 equal groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Run 100000 times: random pick one nodes from each side, <em>if swap them can reduce #edges between 2 groups, do it.</em></td>
</tr>
<tr>
<td>2</td>
<td>游書豪,周融瑋</td>
<td>0.3075</td>
<td>1. Randomly pick one node in GCC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Run BFS until get half GCC size. Mark as two groups</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Remove edges between two groups</td>
</tr>
<tr>
<td>3</td>
<td>顏恩勗, 黃適文</td>
<td>0.31</td>
<td>1. Randomly split to 2 equal groups</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Compute “in” “out” vertices</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. For each node if in &lt; out, exchange</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4. While there exists nodes with in &lt; out</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5. Remove edges between 2 groups</td>
</tr>
</tbody>
</table>
# HW1.3 graph_cm2

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Score</th>
<th>Algorithm Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>陳昱志</td>
<td>0.13</td>
<td><strong>Newman Fast Algorithm (betweenness + modularity)</strong></td>
</tr>
</tbody>
</table>
| 2    | 游書豪, 周融瑋         | 0.148 | 1. Randomly pick one node in GCC.  
                  |          | 2. Run BFS until 50% GCC size. Mark as two groups  
                  |          | 3. Remove edges between two groups                                                    |
| 3    | 林琬瑜, 洪三權         | 0.168 | 1. Run BFS on node with highest degree, mark them as g1 until 50% nodes, others are marked as g2  
                  |          | 2. Remove edges between g1 and g2                                                    |
HW1.3 graph_ws

| Rank 1 | 郭英樹, 林瑋詩, 何伯樟 | 0.0003 (6) | 1. Randomly split to 2 equal groups.  
2. Run 100000 times: random pick one nodes from each side, if swap them can reduce #edges between 2 groups, do it. |
| Rank 1 | 林志鴻 | 0.0003 (6) | **Edge Betweenness Centrality** |
| Rank 3 | 林琬瑜, 洪三權 | 0.0004 | 1. Run BFS on node with highest degree, mark them as g1 until 50% nodes, others are marked as g2  
2. Remove edges between g1 and g2 |
Common Strategy for Attack

• **Degree-based Removal**
  – Sum of Degree of end vertices
  – Incident edges of vertex with highest degree
  – Degree Max, Min

\[
\text{score}_{(x,y) \in E} = (\text{degree}(x) + \text{degree}(y)) / \text{max}(\text{degree}(x), \text{degree}(y))
\]

\[
\text{edge\_score}(n_1, n_2) = \frac{\sqrt{\text{deg}(n_1) \times \text{deg}(n_2)}}{||\text{nb}(n_1) \cap \text{nb}(n_2)||}
\]

Score = |\text{Degree(EndPoint}_1\text{)} - \text{Degree(EndPoint}_2\text{)}|
Score = |\text{Degree(EndPoint}_1\text{)} + \text{Degree(EndPoint}_2\text{)}|

• **Edge Betweennessness Removal**
  – Recalculate
Another Common Strategy

• Consider this ring with $k=6$, $p=0.01$
HW 1.2 Task 1

• Real-world Network Generation
  – Allow addition and deletion
  – Power Law, Low APL, High CC

• Common Strategy
  – Extend Barabasi-Albert Model
  – Add new links to nodes that are neighbors each other (to increase CC)
  – Apply a small probability to control deletion, and delete those with lower degree
利用 power-law distribution 先行製造出一個符合 $\alpha=7$ 的函數圖形

產生頂點個數為 100000 的圖，使其為一個巨大的環，每個頂點的 degree 值皆為 2

根據生成的 distribution 換算一個頂點應有的 degree 數，並將 degree 數減去 2

將每個頂點還需增加的 degree 數套入圖上，使每個頂點都有各自的 degree 數

當任 2 個頂點都沒有機會連成邊，則此圖完成；否則繼續上一步驟

隨機挑取 degree 數未滿的兩個頂點，連接起來，並且將兩邊的 degree 數各自減 1

當選取的兩頂點已有邊相連接，則重新選擇頂點