Community Detection Algorithms

Dr. Natarajan Meghanathan
Professor of Computer Science
Jackson State University, Jackson, MS
E-mail: natarajan.meghanathan@jsums.edu
Community

- Community: It is formed by individuals such that those within a group interact with each other more frequently than with those outside the group.
- Community detection: discovering groups in a network where individuals’ group memberships are not explicitly given.
  - Interactions (edges) between nodes can help determine communities
- Community structures are quite common in real networks. Social networks include community groups based on common location, interests, occupation, etc.
- Metabolic networks have communities based on functional groupings.
- Citation networks form communities by research topic.
- Identifying the community sub structures within a network can provide insight into how network function and topology affect each other.

There is most likely a path from one vertex to another vertex within a community through the vertices that are also part of the same community.

For the Karate Club network (to the left), the internal densities of the two communities are 0.26 and 0.24; the external densities are 0.035; the overall network density is 0.14.
Internal and External Community Densities

Let C be a subset of nodes (V) that form a community. For every node i in C, let $k_i^{\text{int}}$ and $k_i^{\text{ext}}$ be the # links connecting node i to a node in C and outside C respectively.

$$\delta_{\text{int}}(C) = \frac{\sum_i k_i^{\text{int}}}{n_C(n_C - 1)}$$
$$\delta_{\text{ext}}(C) = \frac{\sum_i k_i^{\text{ext}}}{2n_C(n_C - 1)}$$

The internal density of every cluster is significantly larger than the external density as well as the total density of the network.

<table>
<thead>
<tr>
<th>Internal Densities</th>
<th>C1</th>
<th>(4<em>3 + 1</em>4) / (5*4) = 0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C2</td>
<td>(6<em>5)/(6</em>5) = 1.0</td>
</tr>
<tr>
<td></td>
<td>C3</td>
<td>(4<em>3)/(4</em>3) = 1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>External Densities</th>
<th>C1</th>
<th>(1 + 1) / (2<em>5</em>4) = 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C2</td>
<td>(1 + 1 + 1 + 1)/(2<em>6</em>5) = 0.067</td>
</tr>
<tr>
<td></td>
<td>C3</td>
<td>(1 + 1)/(2<em>4</em>3) = 0.083</td>
</tr>
</tbody>
</table>
Schemes for Identifying Communities

• The number of communities within a network is typically unknown and the communities are often of unequal size and/or density.

• Schemes:
  – Clique-based
  – Hierarchical Clustering
    • Bottom-up and Top-down
  – Neighborhood Overlap based
  – Homophily
  – Eigen Vector based

• Evaluation:
  – Modularity Maximization
  – Silhouette Index
Clique-based Schemes
Clique (Complete Mutuality)

- A clique in a graph is a subgraph in which all the constituent nodes are directly reachable from one another.
- It is a NP-hard problem to find the maximum-sized clique in a graph.
  - Independent Set-based Minimum Neighbors Heuristic
    - Could find more than one clique of different sizes
  - Repeated Vertex and Edge Removal Heuristic
    - To find a clique of maximum size (depends on an underlying heuristic)
  - Clique Percolation methods (could find overlapping communities)
- We will use the notion of Independent Sets to find a clique in a graph
  - Independent Set: A subset of vertices such that there is no edge in the graph between any two vertices in the subset
  - For a given graph G, we will find a complement graph G*:
    - The vertices in G and G* are the same.
    - If an edge (u, v) is in G, there is no edge (u, v) in G*
    - If an edge (u, v) is not in G, there is an edge (u, v) in G*
  - An independent set in the complement graph G* is a clique in the original graph G.
Example to Find Independent Set and Clique: Minimum Neighbors Heuristic

Idea: Give preference to vertices with minimal number of (uncovered) neighbors to be part of the Independent Set. A vertex is said to be covered if itself or any of its neighbors in the Independent Set.

Independent Set for the above graph = \{v2, v4, v6\}

This is also the Maximal Independent Set (i.e., there exists no Independent Set of size 4 or more for the above graph). However, the heuristic is not guaranteed in general to give a maximal Independent set.
Example 1 to Determine a Clique Using the Minimum Neighbors Heuristic to Approximate an Independent Set

{v1, v2, v5} is an Independent Set in G* and it is a clique in G.
Example to Find Several Cliques In a Graph

Find the Complement Graph $G^*$

Finding Clique # 1
Max. Independent Set 1 in $G^*$ = \{5, 6, 7, 8\} = Max. Clique in $G$
Removing the edges associated with 5, 6, 7, 8 from the original graph G and generate a new graph G’. Find a new complement Graph G** (for G’)

Remove stub nodes and isolated nodes (nodes with only one edge or no edge)

Reduced graph G’

New Complement Graph G**

[5, 6, 7, 8] is Clique # 1
\{1, 2, 3\} is a clique in $G'$
[1, 2, 3] is Clique # 2
Removing edges associated with 1, 2, 3

Graph G''

Complement Graph of G''
[3, 4, 5] is a clique in $G''$

The three cliques

[5, 6, 7, 8]
[1, 2, 3]
[3, 4, 5]
Pruning Technique: Maximum Clique

• Lets say, we are interested to find a clique of size $k$.
  – All vertices in such clique must have degree at least $k-1$.
  • Repeat the following until we only have vertices with degree $k-1$ or above in the graph
    – Step 1: Remove vertices that have degree less than $k-1$
    – Step 2: Because of the removal of the vertices in Step 1, the degree of some other vertices would have become less than $k-1$.
      » If any such vertices exist, Go to Step 1.
      » Otherwise, exit from the loop
  • If the reduced graph obtained from the above has one or more components in which each component has vertices with degree $k-1$ or above, run any heuristic to find clique (say, the Independent Set-based heuristic) on the reduced graph
Finding the Maximum Clique

Anticipating a Clique of Size 4 (i.e., each vertex in the clique has degree 3)
Remove from the graph all vertices with degree less than 3.
Recursively remove all the vertices and associated edges until each vertex
has degree 3 or above. May not be effective all the time
Apply the Independent Set Heuristic

[5, 6, 7, 8] form an independent set in the complement graph

Hence [5, 6, 7, 8] form a clique in the original graph
Clique Percolation Method

• Used to find overlapping communities
  – **Input**
    • A parameter $k$, and a network
  – **Procedure**
    • Find out all cliques of size $k$ in a given network
    • Construct a *clique graph*. Two cliques are adjacent if they share $k-1$ nodes
    • Each *connected* component in the clique graph forms a community
Example 1: Clique Percolation Method

Communities:
1. \{1, 2, 3, 4\}
2. \{4, 5, 6, 7, 8\}

Cliques of size 3:
1. \{1, 2, 3\}
2. \{1, 3, 4\}
3. \{4, 5, 6\}
4. \{5, 6, 7\}
5. \{5, 6, 8\}
6. \{5, 7, 8\}
7. \{6, 7, 8\}
Example 2: Clique Percolation Method

The following is the clique graph. All the cliques of size 3 are connected. Hence, all the vertices in the given graph are said to be in one single community.

Cliques of Size 3

[1, 2, 3]
[1, 3, 4]
[3, 4, 5]
[4, 5, 6]
[5, 6, 7]
[5, 6, 8]
[5, 7, 8]
[6, 7, 8]
Modularity Maximization
Modularity Maximization

• Modularity measures the strength of a community partition by taking into account the degree distribution.

• Given a network with \( m \) edges, the expected number of edges between two nodes \( i \) and \( j \) with degrees \( d_i \) and \( d_j \) respectively is \( \frac{d_i d_j}{2m} \).

\[
\text{Expected number of edges between nodes 1 and 2 is } (3)(2) / (2*15) = 0.20
\]

Strength of a Community, \( C \)

\[
\sum_{i \in C, j \in C} A_{i,j} - \frac{d_i d_j}{2m}
\]

For a network with \( k \) communities and a total of \( m \) edges

Modularity:

\[
Q = \sum_{l=1}^{k} \sum_{i \in C_i, j \in C_j} A_{i,j} - \frac{d_i d_j}{2m}
\]

A larger value for \( Q \) indicates a good community structure.
Modularity Maximization

- The intuition behind the idea of modularity is that a community is a structural element of a network that has been formed in a manner far from a random process.

- If we consider the actual density of links in a community, it should be significantly larger than the density we would expect if the links in the network were formed by a random process.
  - In other words, if two nodes i and j are end up being in the same community, there should be more likely a link between them (i.e., $A_{ij} = 1$, leading to an overall high value for $Q$).
  - If i and j end up being in a community such that the chances of having a link between them is just as the same as between any two nodes in the network (i.e., a random network), then the value of $Q$ is more likely to be low (because there could be some $A_{ij} = 0$ that will bring down the value of $Q$).
Evaluating Modularity (Example 1)

**Community [1, 4, 5, 7]**

Edges with $A_{ij} = 1$

<table>
<thead>
<tr>
<th>Edges</th>
<th>Modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 4</td>
<td>$1 - (3)(4)/(2*15) = 0.60$</td>
</tr>
<tr>
<td>4 – 5</td>
<td>$1 - (4)(5)/(2*15) = 0.33$</td>
</tr>
<tr>
<td>5 – 7</td>
<td>$1 - (3)(5)/(2*15) = 0.50$</td>
</tr>
</tbody>
</table>

Edges with $A_{ij} = 0$

<table>
<thead>
<tr>
<th>Edges</th>
<th>Modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 5</td>
<td>$0 - (3)(5)/(2*15) = -0.50$</td>
</tr>
<tr>
<td>1 – 7</td>
<td>$0 - (3)(3)/(2*15) = -0.30$</td>
</tr>
<tr>
<td>4 – 7</td>
<td>$0 - (4)(3)/(2*15) = -0.40$</td>
</tr>
</tbody>
</table>

Total Modularity Score for Community [1, 4, 5, 7]: 0.23

**Community [2, 3, 6, 8]**

Edges with $A_{ij} = 1$

<table>
<thead>
<tr>
<th>Edges</th>
<th>Modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 – 3</td>
<td>$1 - (3)(4)/(2*15) = 0.60$</td>
</tr>
<tr>
<td>2 – 6</td>
<td>$1 - (3)(5)/(2*15) = 0.50$</td>
</tr>
<tr>
<td>6 – 8</td>
<td>$1 - (3)(5)/(2*15) = 0.50$</td>
</tr>
</tbody>
</table>

Edges with $A_{ij} = 0$

<table>
<thead>
<tr>
<th>Edges</th>
<th>Modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 – 8</td>
<td>$0 - (3)(3)/(2*15) = -0.30$</td>
</tr>
<tr>
<td>3 – 6</td>
<td>$0 - (4)(5)/(2*15) = -0.67$</td>
</tr>
<tr>
<td>3 – 8</td>
<td>$0 - (4)(3)/(2*15) = -0.40$</td>
</tr>
</tbody>
</table>

Total Modularity Score for Community [2, 3, 6, 8]: 0.23

Total Modularity for the two Communities: $0.23 + 0.23 = 0.46$
Evaluating Modularity (Example 2)

Community [1, 2, 3, 4]

Edges with $A_{ij} = 1$  
Modularity

$1 - 2$  
$1 - (3)(3)/(2*15) = 0.70$

$1 - 3$  
$1 - (3)(4)/(2*15) = 0.60$

$1 - 4$  
$1 - (3)(4)/(2*15) = 0.60$

$2 - 3$  
$1 - (3)(3)/(2*15) = 0.70$

$3 - 4$  
$1 - (4)(4)/(2*15) = 0.47$

Edges with $A_{ij} = 0$

$2 - 4$  
$0 - (3)(4)/(2*15) = -0.40$

Total Modularity Score for Community [1, 2, 3, 4]  
2.67

Community [5, 6, 7, 8]

Edges with $A_{ij} = 1$  
Modularity

$5 - 6$  
$1 - (5)(5)/(2*15) = 0.17$

$5 - 7$  
$1 - (3)(5)/(2*15) = 0.50$

$5 - 8$  
$1 - (3)(5)/(2*15) = 0.50$

$6 - 7$  
$1 - (3)(5)/(2*15) = 0.50$

$6 - 8$  
$1 - (3)(5)/(2*15) = 0.50$

$7 - 8$  
$1 - (3)(3)/(2*15) = 0.70$

Total Modularity Score for Community [5, 6, 7, 8]  
2.87

Total Modularity for the two Communities: 2.67 + 2.87 = 5.54
Evaluating Clusters with Silhouette Index
Silhouette Index

• Let there be \( r \) communities (clusters) \( C_1, C_2, \ldots, C_r \).
• For every node \( i \), determine the shortest path distances (\# hops) to every other node \( j \).
• Find the average distance of node \( i \) to the nodes in each of the \( r \) clusters.
• For a node \( i \) in cluster \( C_k \), find the minimum average distance \( \min \left\{ \bar{d} \min \right\} \) to a cluster \( C_j \) (\( j \neq k \)).

\[
\text{Silhouette Index of a node } i: \\
\quad s(i) = \frac{\bar{d} \min - \bar{d}_i, C_k}{\max\{\bar{d}_i, \bar{d}_j, C_k\}}
\]

\[
\text{Silhouette Index of a cluster } C_k: \\
\quad s(C_k) = \frac{1}{n_{C_k}} \sum_{i=1}^{n_{C_k}} s(i)
\]

\[
\text{Silhouette Index of a graph } G: \\
\quad s(G) = \frac{1}{r} \sum_{j=1}^{r} s(C_j)
\]

\( s(i) \to 1 \) implies the node is in the best possible cluster
\( s(i) \to -1 \) implies the node is in the wrong Cluster
\( s(i) \to 0 \), implies it would be possible to move the node to an adjacent cluster
Silhouette Index: Correct Clustering

Mem. cluster | C1: 5, 6, 7, 8 | C2: 1, 2, 3, 4 | $\bar{d}_{\min}$ (outside own cluster) | $\bar{d}_{i, C^k}$ (Mem.cluster) | $s(i) = \frac{\bar{d}_{\min} - \bar{d}_{i, C^k}}{\max\{\bar{d}_{\min}, \bar{d}_{i, C^k}\}}$
---|---|---|---|---|---
1 | C2 | 2.5 | 0.75 | 2.5 | 0.75 | 0.7 | s(C2)
2 | C2 | 1.75 | 1.0 | 2.0 | 1.0 | 0.5
3 | C2 | 2.0 | 0.75 | 2.0 | 0.75 | 0.625 | Avg(s_i)
4 | C2 | 1.5 | 1.0 | 1.5 | 1.0 | 0.333
5 | C1 | 0.75 | 1.5 | 1.5 | 0.75 | 0.5 | s(C1)
6 | C1 | 0.75 | 1.5 | 1.5 | 0.75 | 0.5
7 | C1 | 0.75 | 2.25 | 2.25 | 0.75 | 0.67 | Avg(s_i)
8 | C1 | 0.75 | 2.25 | 2.25 | 0.75 | 0.67

Overall Silhouette Index of the Network = Average of the index of all clusters = **0.565**
Silhouette Index: Wrong Clustering

Mem. cluster | C1 2, 3, 6, 8 | C2 1, 4, 5, 7 | $d_{\text{min}}$ (outside own cluster) | $d_{i,C_k}$ (Mem.cluster) | $s(i)$
--- | --- | --- | --- | --- | ---
1 | C2 | 1.75 | 1.5 | 1.75 | 1.5 | 0.143 s(C2)
2 | C1 | 1.0 | 2.0 | 2.0 | 1.0 | 0.5 Avg(si)
3 | C1 | 1.5 | 1.25 | 1.25 | 1.5 | -0.2
4 | C2 | 1.5 | 1.0 | 1.5 | 1.0 | 0.333
5 | C2 | 1.25 | 1.0 | 1.25 | 1.0 | 0.2 s(C1)
6 | C1 | 1.0 | 1.25 | 1.25 | 1.0 | 0.2 Avg(si)
7 | C2 | 1.5 | 1.5 | 1.5 | 1.5 | 0
8 | C1 | 1.25 | 1.75 | 1.75 | 1.25 | 0.286

Overall Silhouette Index of the Network = Average of the index of all clusters = **0.183**
Hierarchical Clustering
(Complete Linkage Clustering)

Bottom-Up Approach
(Agglomerative)
Complete Linkage Clustering

- Compute the “pair-wise” distance matrix $P$ between any two vertices.
- Initially, start with each vertex in its own cluster.
- Merge the two “closest” vertices (clusters)
  - In case of a tie (between two or more pairs of clusters), choose the pair with the minimum value for the total pair-wise distance / sum of the two pair sizes
- Remove the entries from $P$, for the two vertices (clusters) merged, and add an entry corresponding to the merged vertex (cluster).
  - Update this entry with the longest distance between any vertex in the merged cluster with the vertices in the other clusters in $P$.

$$\text{Distance}(C_i, C_j) = \max \left[ \forall_{u \in C_i} \forall_{v \in C_j} \text{MinHops}(u, v) \right]$$

- Repeat the above step of merging and removing/adding entries to $P$ until there is only one cluster.
Complete Linkage Clustering (Example 1)

Break the tie by choosing the Pair with the minimum total Pair-wise distance / Pair size

(1,2,3) and (5): 5/4 = 1.25

(4,6) and (5): 3/3 = 1.0
Complete Linkage Clustering
Complete Linkage Clustering

Modularity(1,2,3)
Mod(1,2) = 1 – (2*3)/(2*7) = 0.571
Mod(1,3) = 1 – (2*3)/(2*7) = 0.571
Mod(2,3) = 1 – (3*3)/(2*7) = 0.357

Modularity(4,5,6)
Mod(4,5) = 0 – (2*2)/(2*7) = -0.286
Mod(4,6) = 1 – (2*2)/(2*7) = 0.714
Mod(5,6) = 1 – (2*2)/(2*7) = 0.714

Total Modularity = 2.641
Complete Linkage Clustering

Total Modularity = 1.285

Mod(1,2) = 1 – (2*3)/(2*7) = 0.571
Mod(3) = 0
Mod(5) = 0
Mod(4,6) = 1 – (2*2)/(2*7) = 0.714
Final Partition

Total Modularity = 2.641
Complete Linkage Clustering (Example 2)

```
1 2 3 4 5 6 7 8
1 0 1 1 1 2 2 3 3
2 0 1 2 2 1 2 2 2
3 0 1 1 2 2 3 3 3
4 0 1 1 2 2 2 3 3
5 0 1 1 1 1 1 1 1
6 0 1 1 1 1 1 1 1
7 0 1 1 1 1 1 1 1
8 0 1 1 1 1 1 1 1
```

```
1,2 3 4 5 6 7 8
1,2 0 1 2 2 2 3 3 3
3 0 1 1 2 2 3 3 3
4 0 1 1 2 2 2 3 3
5 0 1 1 1 1 1 1 1
6 0 1 1 1 1 1 1 1
7 0 1 1 1 1 1 1 1
8 0 1 1 1 1 1 1 1
```

```
1,2, 3
1,2  3
1,2   3
1,2,
1,2   3
1,2
```
Break the tie by choosing the Pair with the minimum total Pair-wise distance / Sum of pair size
(1,2,3) and (4,5,6): 14/6 = 2.33
(4,5,6) and (7,8): 8/5 = 1.6
Modularity(4,5,6,7,8)
Mod(4,5) = 1 – (4*5)/(2*15) = 0.33
Mod(4,6) = 1 – (4*5)/(2*15) = 0.33
Mod(4,7) = 0 – (3*4)/(2*15) = -0.4
Mod(4,8) = 0 – (3*4)/(2*15) = -0.4
Mod(5,6) = 1 – (5*5)/(2*15) = 0.17
Mod(5,7) = 1 – (3*5)/(2*15) = 0.50
Mod(5,8) = 1 – (3*5)/(2*15) = 0.50
Mod(6,7) = 1 – (3*5)/(2*15) = 0.50
Mod(6,8) = 1 – (3*5)/(2*15) = 0.50
Mod(7,8) = 1 – (3*3)/(2*15) = 0.70

Total Modularity 4.63
Final Partition

From the previous slides, We know that the optimal Partitioning of the graph is:

Total Modularity = 4.63

Complete Linkage Clustering

Modularity (1, 2, 3, 4) + Modularity (5, 6, 7, 8) = 5.54

Thus, complete linkage clustering need not always give the optimal solution.
Hierarchical Clustering
Edge Betweenness

Top-Down Approach
(Divisional)
Edge Betweenness as Flow along the Edge

• Here, we model edge betweenness as a measure of the total amount of flow (proportional to the number of shortest paths the edge is part of) it carries, counting flow between all pairs of nodes using this edge.

• Note that in this graph below, there is only one shortest path between any two nodes. Hence, the total amount of flow through an edge is the number of shortest paths through that edge.

Example 1

Edge 7-8 in the graph here carries flow from each of the 7 nodes on the left (incl. node 7) and the 7 nodes on the right (incl. node 8). The number of shortest paths through Edge 7-8 is 7*7 = 49.
Edge Betweenness: Motivating Example

# Shortest Paths through edge 3 – 7 is: $3 \times 11 = 33$
Similarly, the # Shortest Paths through edges 6 – 7, 8 – 9 and 8 – 12 are 33 each.
# Shortest Paths through edge 1 – 3 is 12. Similarly, the # Shortest Paths through edges 2 – 3, 4 – 6, 5 – 6, 9 – 10, 9 – 11, 12 – 13 and 12 – 14 are 12 each.
# Shortest Paths through edge 1 – 2, 4 – 5, 10 – 11 and 13 – 14 are 1 each.
Edge Betweenness: Motivating Example 1

Step 2
# Shortest Paths
3 – 7: $3 \times 4 = 12$
6 – 7: $3 \times 4 = 12$
1 – 2: 1
4 – 5: 1
1 – 3: 5
5 – 6: 5
2 – 3: 5
4 – 6: 5
Partition Tree

1, 2, 3, ..., 14

1, 2, 3, 4, 5, 6, 7

8, 9, 10, ..., 14

1, 2, 3

4, 5, 6

9, 10, 11

8

12, 13, 14

1

2

3

7

4

5

6

9

10

11

8

12

13

14
Modularity \((1, 2, 3, 4, 5, 6, 7)\)

\[
\begin{align*}
\text{Mod (1, 2)} &= 1 - \frac{(2\times2)}{(2\times17)} = 0.882 \\
\text{Mod (1, 3)} &= 1 - \frac{(2\times3)}{(2\times17)} = 0.824 \\
\text{Mod (1, 4)} &= 0 - \frac{(2\times2)}{(2\times17)} = -0.118 \\
\text{Mod (1, 5)} &= 0 - \frac{(2\times2)}{(2\times17)} = -0.118 \\
\text{Mod (1, 6)} &= 0 - \frac{(2\times3)}{(2\times17)} = -0.176 \\
\text{Mod (1, 7)} &= 0 - \frac{(2\times3)}{(2\times17)} = -0.176 \\
\text{Mod (2, 3)} &= 1 - \frac{(2\times3)}{(2\times17)} = 0.824 \\
\text{Mod (2, 4)} &= 0 - \frac{(2\times2)}{(2\times17)} = -0.118 \\
\text{Mod (2, 5)} &= 0 - \frac{(2\times2)}{(2\times17)} = -0.118 \\
\text{Mod (2, 6)} &= 0 - \frac{(2\times3)}{(2\times17)} = -0.176 \\
\text{Mod (2, 7)} &= 0 - \frac{(2\times3)}{(2\times17)} = -0.176 \\
\end{align*}
\]

\[
\begin{align*}
\text{Mod (3, 4)} &= 0 - \frac{(2\times3)}{(2\times17)} = -0.176 \\
\text{Mod (3, 5)} &= 0 - \frac{(2\times3)}{(2\times17)} = -0.176 \\
\text{Mod (3, 6)} &= 0 - \frac{(3\times3)}{(2\times17)} = -0.265 \\
\text{Mod (3, 7)} &= 1 - \frac{(3\times3)}{(2\times17)} = 0.735 \\
\text{Mod (4, 5)} &= 1 - \frac{(2\times2)}{(2\times17)} = 0.882 \\
\text{Mod (4, 6)} &= 1 - \frac{(2\times3)}{(2\times17)} = 0.824 \\
\text{Mod (4, 7)} &= 0 - \frac{(2\times3)}{(2\times17)} = -0.176 \\
\text{Mod (5, 6)} &= 1 - \frac{(2\times3)}{(2\times17)} = 0.824 \\
\text{Mod (5, 7)} &= 0 - \frac{(2\times3)}{(2\times17)} = -0.176 \\
\text{Mod (6, 7)} &= 1 - \frac{(3\times3)}{(2\times17)} = 0.735 \\
\end{align*}
\]

Modularity \((1, 2, 3, 4, 5, 6, 7)\) = 4.385
Modularity \((8, 9, 10, \ldots, 14)\) = 4.385

**Total Modularity = 8.77**
Modularity \((1, 2, 3)\)

- \(\text{Mod} (1, 2) = 1 - \frac{(2 \times 2)}{(2 \times 17)} = 0.882\)
- \(\text{Mod} (1, 3) = 1 - \frac{(2 \times 3)}{(2 \times 17)} = 0.824\)
- \(\text{Mod} (2, 3) = 1 - \frac{(2 \times 3)}{(2 \times 17)} = 0.824\)
- \(\text{Mod} (1, 2, 3) = 2.53\)

Due to symmetry,
- \(\text{Modularity} (4, 5, 6) = 2.53\)
- \(\text{Modularity} (9, 10, 11) = 2.53\)
- \(\text{Modularity} (12, 13, 14) = 2.53\)

Total Modularity = 10.12
Example 1

Final Partitioning into Communities
Example 2

Note: For an edge like 2 – 5, it could be part of shortest paths from 1 to 5, 6, 7, …, 11. But, there are also shortest paths from 1 to 5, 6, , …, 11 through edge 3 – 5. Hence, the net # shortest paths on 2 – 5 from 1 to the seven vertices 5, 6, 7, …, 11 is 7/2 = 3.5

# Shortest Paths through edge 5 – 7 is: 5 * 5 = 25
# Shortest Paths through edge 5 – 6 (from 1, 2, .., 5 to 6) and 6 – 7 is only 5 each
# Shortest Paths through edge 2 – 5 is: 7 (from 2 to 5..11) + 3.5 (from 1 to 5…11) = 10.5
Similarly, # Shortest Paths through edge 3 – 5, 7 – 9 , 7 – 10 are 10.5 each.
# Shortest Paths through 4 – 5 (4 to 5, 6, …, 11) and 7 – 8 are 7 each
# Shortest Paths through 2 – 4, 3 – 4, 8 – 9 and 8 – 10 are only 1 each
# Shortest Paths through 2 – 3 and 9 – 10 are only 1 each
# Shortest Paths through 1 – 2 is 1 + 4.0 (1/2 for each of 4, 5, …, 11) = 5.0
# Shortest Paths through 1 – 3, 9 – 11, 10 – 11 are also 5 each.
Note that after Step 1, the number of shortest paths through edge 5 – 6 becomes: $5 \times 6 = 30$
Likewise, the number of shortest paths through edge 6 – 7 becomes: $5 \times 6 = 30$

After Step 2, the number of shortest paths through edge 1 – 2 is: $1 \text{ (1 to 2)} + 1 \text{ (1/2 from 1 to 4 and 5)} = 2.0$. Similarly, the number of shortest paths through edge 1 – 3 is 2.0
The number of shortest paths through edge 2 – 4 is $1 + \frac{1}{2} \text{ (from 1 to 4)} = 1.5$

Hence, edges 1 – 2 and 1 – 3 have high betweenness (2 each) than edges 2 – 4 and 3 – 4 (1.5 each)
Partition Tree

1, 2, 3, ..., 11

1, 2, 3, 4, 5

6

7, 8, 9, 10, 11

11

2, 3, 4, 5

6

7, 8, 9, 10
Modularity(1, 2, 3, 4, 5) = 4.657
Similarly, Modularity(7, 8, 9, 10, 11) = 4.657
Total Modularity (Level 1) = 9.314

Modularity (2, 3, 4, 5) = 3.5
Modularity (7, 8, 9, 10) = 3.5
Total Modularity (Level 2) = 7.0

Total Modularity (Level 1) > Total Modularity (Level 2)
Hence, we will go with Level 1 partitioning
Final Partitioning
Finding the # Shortest Paths through an Edge

• For graphs in which there is more than one paths between one or more pair of vertices, the total flow through an edge is not equal to the total # shortest paths through the edge.

• We will now see an algorithm proposed by Girvan and Newman to determine the total flow through an edge.

• Repeat the following for every vertex
  – Perform a Breadth First Search (BFS) of the graph, starting from the first vertex, say A.
  – Determine the # shortest paths from A to each other node using the BFS levels of the nodes
  – Based on the above numbers of shortest paths, determine the amount of flow from A to all the other vertices that uses each edge.

• The total flow through an edge is the sum (for directed graph) or half of the sum (for undirected graph) of the flows determined through that edge when BFS is run from every vertex in the graph.
  – For undirected graph, we divide by the total sum of the flows by 2 because an edge is counted twice on the shortest path between any two vertices.
  – For example A – B – C; the edge A – B is counted twice (once on the shortest path from A to C and once on the shortest path from C to A
Computing the Flow Values
We assume one unit of flow originates at each node. We start with the node at the bottom most level. Let 1 unit of flow start from node D. Node D gets 2 of its Shortest paths to node A through F and 1 through G. So, node A sends 2/3 of the flow to F and 1/3 of the flow to G. Node F adds 2/3 flow received to 1 unit of flow originating at itself and splits the resulting 1.67 equally and sends 0.835 to each of B and E. G merely adds the 0.33 flow units to the 1 units of flow originating at itself and sends 1.33 to B.
BFS run on B

Node Levels

# Shortest Paths from Node B to every other Node

Flow on each edge
BFS run on G

Node Levels

# Shortest Paths from Node G to every other Node

Flow on each edge
BFS run on E

Node Levels

# Shortest Paths from Node E to every other Node

Flow on each edge
BFS run on F

Node Levels

# Shortest Paths from Node F to every other Node

Flow on each edge
BFS run on D

Node Levels

# Shortest Paths from Node D to every other Node

Flow on each edge
Analysis of: Girvan and Newman Algorithm

• After we get the total flow through each edge (betweenness of the edges), we remove the edge with the largest betweenness.

• We re-run BFS on each vertex and find the betweenness of every edge and remove the edge with the largest betweenness henceforth.

• We repeat this process until we divide the graph into individual vertices.

• We keep track of the communities that get generated with each edge removal and then decide on the level of partition (to stop the edge removal process) by evaluating the modularity scores of the community scores formed at different levels.

• The Girvan and Newman algorithm, though effective in delineating communities with high modularity scores, is very inefficient as it requires BFS (of time complexity $\Theta(E+V)$) to be run on each vertex for every edge removal.
  – For a graph with $E$ edges and $V$ vertices, the overall time complexity will be $\Theta(EV(E+V))$
Neighborhood Overlap based Approach
Principle of Triadic Closure

• If two people in a social network have a friend in common, then there is an increased likelihood that they will become friends themselves at some point in the future.

• If we observe snapshots of a social network at two distinct points in time, then in the later snapshot, we generally find a significant number of new edges that have formed through this triangle-closing operation, between two people who had a common neighbor in the earlier snapshot.
Neighborhood Overlap: Strong/Weak Tie

Neighborhood Overlap = \frac{\text{number of nodes who are neighbors of both } A \text{ and } B}{\text{number of nodes who are neighbors of at least one of } A \text{ or } B}

Note that one should not count neither A nor B as part of the neighbors in the denominator.

For every edge, determine its neighborhood overlap. If it is above a threshold, then the edge could be classified to be of “strong tie”, otherwise, we say weak tie.

<table>
<thead>
<tr>
<th>Edge</th>
<th>Neighborhood Overlap (NOVER) Score</th>
<th>Tie</th>
</tr>
</thead>
<tbody>
<tr>
<td>A – B</td>
<td>0/(4+2) = 0</td>
<td>Weak</td>
</tr>
<tr>
<td>B – G</td>
<td>1/(2+1) = 0.33</td>
<td>Weak</td>
</tr>
<tr>
<td>B – D</td>
<td>1/(2+1) = 0.33</td>
<td>Weak</td>
</tr>
<tr>
<td>G – D</td>
<td>2/(2+0) = 1.0</td>
<td>Strong</td>
</tr>
<tr>
<td>D – H</td>
<td>1/(2+0) = 0.5</td>
<td>Strong</td>
</tr>
<tr>
<td>A – I</td>
<td>1/(4+0) = 0.25</td>
<td>Weak</td>
</tr>
<tr>
<td>A – E</td>
<td>3/(4+0) = 0.75</td>
<td>Strong</td>
</tr>
<tr>
<td>E – I</td>
<td>1/(3+0) = 0.33</td>
<td>Weak</td>
</tr>
<tr>
<td>E – F</td>
<td>2/(3+0) = 0.67</td>
<td>Strong</td>
</tr>
<tr>
<td>F – C</td>
<td>2/(2+0) = 1.0</td>
<td>Strong</td>
</tr>
<tr>
<td>A – F</td>
<td>2/(4+0) = 0.5</td>
<td>Strong</td>
</tr>
</tbody>
</table>

Let threshold NOVER score = 0.5
Weak Ties for Community Detection

- Weak ties serve to link together different tightly-knit communities that each contain a large number of stronger ties.
- Remove the Weak ties in the increasing order of their neighborhood overlap value.
  - The graph will get decomposed into several components (communities).

<table>
<thead>
<tr>
<th>Edge</th>
<th>Neighborhood Overlap (NOVER) Score</th>
<th>Tie</th>
</tr>
</thead>
<tbody>
<tr>
<td>A – B</td>
<td>0/(4+2) = 0</td>
<td>Weak</td>
</tr>
<tr>
<td>B – G</td>
<td>1/(2+1) = 0.33</td>
<td>Weak</td>
</tr>
<tr>
<td>B – D</td>
<td>1/(2+1) = 0.33</td>
<td>Weak</td>
</tr>
<tr>
<td>G – D</td>
<td>2/(2+0) = 1.0</td>
<td>Strong</td>
</tr>
<tr>
<td>D – H</td>
<td>1/(2+0) = 0.5</td>
<td>Strong</td>
</tr>
<tr>
<td>A – I</td>
<td>1/(4+0) = 0.25</td>
<td>Weak</td>
</tr>
<tr>
<td>A – E</td>
<td>3/(4+0) = 0.75</td>
<td>Strong</td>
</tr>
<tr>
<td>E – I</td>
<td>1/(3+0) = 0.33</td>
<td>Weak</td>
</tr>
<tr>
<td>E – F</td>
<td>2/(3+0) = 0.67</td>
<td>Strong</td>
</tr>
<tr>
<td>F – C</td>
<td>2/(2+0) = 1.0</td>
<td>Strong</td>
</tr>
<tr>
<td>A – F</td>
<td>2/(4+0) = 0.5</td>
<td>Strong</td>
</tr>
</tbody>
</table>
Strength of an Edge and Strong Triadic Closure Property

• We classify edges to be either a strong tie (more frequent, trusted, incentive, opportunity) or a weak tie (not much frequent interaction, less known)

• In the context of social networks,
  – Strong tie  →  Friend
  – Weak tie  →  Acquaintance

• Strong Triadic Closure Property: If a node A has strong ties to two neighbor nodes B and C, then there should be an edge between B and C (at least a weak tie).
  – A node is said to violate the strong triadic closure property if there is no edge between any two of its neighbor nodes with which it has strong ties.
Example: Strong Triadic Closure
Strength of Weak Ties: Motivating Study

• Granovetter Interviews
  – 54 people who found their jobs via social tie:
    • 16.7% via strong tie (at least two interactions/ week)
    • 55.7% via medium tie (at least one interaction/ year)
    • 27.6% via a weak tie (less than one interaction/ year)
  – Weak ties are like bridges, used to transfer less redundant (but more useful) information.
Social Affiliation Network

• Social Affiliation Network: A network that connects two people, and people with their foci (groups, activities, etc) they are associated with.

• As the network evolves,
  – Two people are likely to be linked to each other if they are affiliated to one or more common foci or common people.
  – A person is likely to be linked to a foci, because of one or more common neighbors who are already linked to the foci.

![Diagram of Social Affiliation Network]

Robert

Peter

Joseph

Org. A

Org. B

Org. C

Org. D
Closing of Triangle in Social Affiliation Networks

**Triadic Closure**

**Focal Closure**

**Membership Closure**
Closing of Triangle in Social Affiliation Networks: Example

Before triangle Closing

After triangle Closing
Triangle Closure: Probabilistic Analysis

- For any two nodes, the larger the number of common neighbors, the larger is the probability of a link to be formed between the two.
- Let $p$ be the probability that a common neighbor node could trigger the formation of a link between two particular nodes.
- If there are $k$ such common neighbors,
  - the probability that a link is still not formed is given by: $(1-p)^k$
  - the probability that a link will be formed (due to triangle closure) is: $1 - (1-p)^k$
  - As $k$ increases, the probability that a link will be formed due to triangle closure increases.
Edge and Node Embeddedness

- The “embeddedness of an edge” is the number of common neighbors the two endpoints have.

- The “embeddedness of a node” is the average of the embeddedness of its associated edges.

- A network could be partitioned into communities around nodes with lower embeddedness.

- The “embeddedness of a community” is the average of the embeddedness of the edges forming the community.

- Structural Hole: The node with lower embeddedness is said to constitute a structural hole, especially when it connects two or more communities with larger embeddedness.

- A split typically occurs along a weak interface between two densely connected regions.
Edge and Node Embeddedness

Nodes B, D and N are structural holes in the above network.
Eigen vectors based approach
Using Eigenvectors to Identify Components

• Compute the Eigenvalues and Eigenvectors of the Adjacency matrix $A$
• The principal Eigenvector is the one that corresponds to the largest Eigenvalue.
• If all the entries in the “principal Eigenvector” are positive, then it implies that all the nodes are in one component.
  – Else, the vertices with the positive entries are in one component and those with the negative entries are in another component. (Note: 0 is considered positive).
• We apply the above interpretation to all the subsequent Eigenvectors (in the decreasing order of the corresponding Eigenvalues) and identify the smaller communities within the larger components.
Example 1: Eigenvectors to Identify Communities (1)

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

The Eigenvalues in the decreasing order and their corresponding Eigenvectors are:

<table>
<thead>
<tr>
<th>Eigenvalue</th>
<th>Eigenvector</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4142</td>
<td>[0.35; 0.35; 0.5; 0.5; 0.35; 0.35]</td>
<td>All entries are +ve; hence all vertices are in one single component</td>
</tr>
<tr>
<td>1.7321</td>
<td>[-0.44; -0.44; -0.33; 0.33; 0.44; 0.44]</td>
<td>Vertices 1, 2, 3 form one community</td>
</tr>
<tr>
<td>-0.4142</td>
<td>[0.35; 0.35; -0.5; -0.5; 0.35; 0.35]</td>
<td>Vertices 4, 5, 6 form another comm.</td>
</tr>
<tr>
<td>-1</td>
<td>[-0.71; 0.71; 0; 0; 0; 0]</td>
<td>Within 1-2-3; 3 is in one comm.</td>
</tr>
<tr>
<td>-1</td>
<td>[0; 0; 0; 0; -0.71; 0.71]</td>
<td>Within 4-5-6; 4 is in one comm.</td>
</tr>
<tr>
<td>-1.7321</td>
<td>[0.23; 0.23; -0.63; 0.63; -0.23; -0.23]</td>
<td></td>
</tr>
</tbody>
</table>
Example 1: Eigenvectors to Identify Communities (2)

2.4142 \ [0.35; 0.35; 0.5; 0.5; 0.35; 0.35] 
1.7321 \ [-0.44; -0.44; -0.33; 0.33; 0.44; 0.44]

Modularity of \([1, 2, 3]\)

<table>
<thead>
<tr>
<th>Edge</th>
<th>Modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 2</td>
<td>(1 - \frac{(2<em>2)}{(2</em>7)}) = 0.71</td>
</tr>
<tr>
<td>1 – 3</td>
<td>(1 - \frac{2<em>3}{(2</em>7)}) = 0.57</td>
</tr>
<tr>
<td>2 – 3</td>
<td>(1 - \frac{2<em>3}{(2</em>7)}) = 0.57</td>
</tr>
<tr>
<td>Total Modularity = 1.85</td>
<td></td>
</tr>
</tbody>
</table>

Modularity of \([4, 5, 6]\)

<table>
<thead>
<tr>
<th>Edge</th>
<th>Modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 – 5</td>
<td>(1 - \frac{(2<em>3)}{(2</em>7)}) = 0.57</td>
</tr>
<tr>
<td>4 – 6</td>
<td>(1 - \frac{2<em>3}{(2</em>7)}) = 0.57</td>
</tr>
<tr>
<td>5 – 6</td>
<td>(1 - \frac{(2<em>2)}{(2</em>7)}) = 0.71</td>
</tr>
<tr>
<td>Total Modularity = 1.85</td>
<td></td>
</tr>
</tbody>
</table>

Total Modularity of \([1, 2, 3]\) and \([4, 5, 6]\) = 3.7
Example 1: Eigenvectors to Identify Communities (3)

Modularity of [3] and [4] are 0 each
Modularity of [1, 2] and [5, 6] are 0.71 each

Total Modularity of [1, 2] and [3] is 0.71
is less than the modularity of [1, 2, 3]. Hence,
We stay with [1, 2, 3] as a community.

Total Modularity of [5, 6] and [4] is 0.71
is less than the modularity of [4, 5, 6]. Hence,
We stay with [4, 5, 6] as a community.

Final Partition

Total Modularity Score = 3.7
Example 2

Eigenvalue # 8 (2.1175)

Eigenvalue # 7 (1.6723)

Eigenvalue # 9 (4.3515)
Example 2 (1)

Eigenvalue # 8 (2.1175)

Modularity for Community [0, 3, 4, 5]

<table>
<thead>
<tr>
<th>Edge</th>
<th>Modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 3</td>
<td>1 – {(4<em>3)/(2</em>20)} = 0.7</td>
</tr>
<tr>
<td>0 – 4</td>
<td>1 – {(4<em>3)/(2</em>20)} = 0.7</td>
</tr>
<tr>
<td>0 – 5</td>
<td>1 – {(4<em>6)/(2</em>20)} = 0.4</td>
</tr>
<tr>
<td>3 – 4</td>
<td>0 – {(3<em>3)/(2</em>20)} = -0.23</td>
</tr>
<tr>
<td>3 – 5</td>
<td>1 – {(3<em>6)/(2</em>20)} = 0.55</td>
</tr>
<tr>
<td>4 – 5</td>
<td>1 – {(3<em>6)/(2</em>20)} = 0.55</td>
</tr>
</tbody>
</table>

Total Modularity Score for [0, 3, 4, 5] = 2.67

Modularity for Community [1, 2, 6, 7, 8, 9]

<table>
<thead>
<tr>
<th>Edge</th>
<th>Modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 2</td>
<td>1 – {(6<em>2)/(2</em>20)} = 0.7</td>
</tr>
<tr>
<td>1 – 6</td>
<td>0 – {(6<em>4)/(2</em>20)} = -0.6</td>
</tr>
<tr>
<td>1 – 7</td>
<td>1 – {(6<em>5)/(2</em>20)} = 0.25</td>
</tr>
<tr>
<td>1 – 8</td>
<td>1 – {(6<em>4)/(2</em>20)} = 0.4</td>
</tr>
<tr>
<td>1 – 9</td>
<td>0 – {(6<em>3)/(2</em>20)} = -0.45</td>
</tr>
<tr>
<td>2 – 6</td>
<td>0 – {(2<em>4)/(2</em>20)} = -0.2</td>
</tr>
<tr>
<td>2 – 7</td>
<td>0 – {(2<em>5)/(2</em>20)} = -0.25</td>
</tr>
<tr>
<td>2 – 8</td>
<td>1 – {(2<em>4)/(2</em>20)} = 0.8</td>
</tr>
</tbody>
</table>
Modularity for Community [1, 2, 6, 7, 8, 9]

<table>
<thead>
<tr>
<th>Edge</th>
<th>Modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 2</td>
<td>1 – {(6<em>2)/(2</em>20)} = 0.7</td>
</tr>
<tr>
<td>1 – 6</td>
<td>0 – {(6<em>4)/(2</em>20)} = -0.6</td>
</tr>
<tr>
<td>1 – 7</td>
<td>1 – {(6<em>5)/(2</em>20)} = 0.25</td>
</tr>
<tr>
<td>1 – 8</td>
<td>1 – {(6<em>4)/(2</em>20)} = 0.4</td>
</tr>
<tr>
<td>1 – 9</td>
<td>0 – {(6<em>3)/(2</em>20)} = -0.45</td>
</tr>
<tr>
<td>2 – 6</td>
<td>0 – {(2<em>4)/(2</em>20)} = -0.2</td>
</tr>
<tr>
<td>2 – 7</td>
<td>0 – {(2<em>5)/(2</em>20)} = -0.25</td>
</tr>
<tr>
<td>2 – 8</td>
<td>1 – {(2<em>4)/(2</em>20)} = 0.8</td>
</tr>
<tr>
<td>2 – 9</td>
<td>0 – {(2<em>3)/(2</em>20)} = -0.15</td>
</tr>
<tr>
<td>6 – 7</td>
<td>1 – {(4<em>5)/(2</em>20)} = 0.5</td>
</tr>
<tr>
<td>6 – 8</td>
<td>0 – {(4<em>4)/(2</em>20)} = -0.4</td>
</tr>
<tr>
<td>6 – 9</td>
<td>1 – {(4<em>3)/(2</em>20)} = 0.7</td>
</tr>
<tr>
<td>7 – 8</td>
<td>1 – {(5<em>4)/(2</em>20)} = 0.5</td>
</tr>
<tr>
<td>7 – 9</td>
<td>1 – {(5<em>3)/(2</em>20)} = 0.63</td>
</tr>
<tr>
<td>8 – 9</td>
<td>1 – {(4<em>3)/(2</em>20)} = 0.7</td>
</tr>
</tbody>
</table>

Total Modularity Score for [1, 2, 6, 7, 8, 9] = 3.13
Example 2 (3)

Modularity of Community [0, 3] = 0.7
Modularity of Community [4, 5] = 0.55
Total Modularity of [0, 3] and [4, 5] = 1.25
is less than the Modularity of [0, 3, 4, 5]
= 2.67
Hence, we stay with community [0, 3, 4, 5]
without further partitioning it.
Example 2 (4)

Modularity of Community [1, 2, 8]
Edge Modularity
1 – 2 \(1 - \frac{(6\times2)}{(2\times20)} = 0.7\)
1 – 8 \(1 - \frac{(6\times4)}{(2\times20)} = 0.4\)
2 – 8 \(1 - \frac{(2\times4)}{(2\times20)} = 0.8\)
Total Modularity of [1, 2, 8] = \(1.9\)

Modularity of Community [6, 7, 9]
Edge Modularity
6 – 7 \(1 - \frac{(4\times5)}{(2\times20)} = 0.5\)
6 – 9 \(1 - \frac{(4\times3)}{(2\times20)} = 0.7\)
7 – 9 \(1 - \frac{(5\times3)}{(2\times20)} = 0.63\)
Total Modularity of [6, 7, 9] = \(1.83\)

Modularity of [1, 2, 8] = 1.9
Modularity of [6, 7, 9] = 1.83
Total Modularity of [1, 2, 8] and [6, 7, 9] = 3.73
is larger than the modularity of [1, 2, 6, 7, 8, 9] = 3.13
Hence, we allow the partitioning of [1, 2, 6, 7, 8, 9] into [1, 2, 8] and [6, 7, 9]
Example 2 (5)

Final Partition

Modularity of \([0, 3, 4, 5]\) = 2.67
Modularity of \([1, 2, 8]\) = 1.9
Modularity of \([6, 7, 9]\) = 1.83
Total Modularity = 6.4
Homophily: Networks and their Surrounding Contexts
Homophily

- In a social network, a person is more closer to people who are similar to him/her with respect to certain characteristics.
  - some immutable characteristics (things that cannot be changed) such as race and ethnicity
  - some mutable characteristics (things the could change over time) such as places we live, we work, beliefs, interests, etc.
- Homophily can produce a division of a network into densely connected homogeneous parts that are weakly connected to each other.

Nodes are colored according to their race
Nodes are students in middle school and high schools of a particular region.

The left and right regions of the network represent communities of nodes belonging to the same race.
The top and bottom regions of the network represent communities of nodes in the middle school and high school respectively.

Figure 4.1 Easley and Kleinberg
Homophily Examples (1)

• National Sample: only 8% of people have *any* people of another race that they `discuss important matters” with (Marsden 87)

• Interracial marriages U.S.: 1% of white marriages, 5% of black marriages, 14% of Asian marriages (Fryer 07)

• In middle school, less than 10% of ``expected” cross-race friendships exist (Shrum et al 88)

• Closest friend: 10% of men name a woman, 32% of women name a man (Verbrugge 77)
  • Far less than 50% if the connections were random
Homophily Examples (2)

Social Connections among Students in High School in US
(Currarini, Jackson, Pin; 2009, 2010)

Smaller groups tend to have different characteristics than larger groups.

<table>
<thead>
<tr>
<th>Percent</th>
<th>52</th>
<th>38</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>White</td>
<td>Black</td>
<td>Hispanic</td>
<td>Other</td>
</tr>
<tr>
<td>White</td>
<td>86</td>
<td>7</td>
<td>47</td>
<td>74</td>
</tr>
<tr>
<td>Black</td>
<td>4</td>
<td>85</td>
<td>46</td>
<td>13</td>
</tr>
<tr>
<td>Hispanic</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Other</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Measuring Homophily

- We will see a technique to find out whether a network exhibits homophily according to a particular characteristic of interest (like race or age).
- If $p$ and $q$ are the fraction of nodes of two particular types (say, male and female), then on average, the probability that a cross-community link is randomly formed between two nodes of different types is $2pq$ (considering bi-directionality).
- If the fraction of cross-community links in the network is significantly less than $2pq$, then the network exhibits **homophily**.
  - That is, a majority of the links are between people belonging to the same type.
  - The definition for “significantly less” is subjective, say 25% of $2pq$.
- If the fraction of cross-community links is in the vicinity of $2pq$, then there is no homophily [the links are more random, not based on node types].
- If the fraction of cross-community links in the network is far greater than $2pq$, then the network exhibits **inverse homophily**
  - A network of romantic relationships.
- We can also extend the analysis to finding homophily involving nodes of more than two types. We have to just estimate the fraction of cross-community links that could be formed involving any two different node types.
Example: Measuring Homophily

Consider a network of classmates
4 – female and 6 – male students

Fraction of male students = 0.6
Fraction of female students = 0.4
Expected fraction of cross-gender links = $2 \times 0.4 \times 0.6 = 0.48$

A total of 20 links are in the network
Of these, 6 are cross-gender links
Fraction of cross-gender links is $6/20 = 0.3$. 

If the threshold for cross-community links is 100%, then $0.3 < 0.48$
The network is said to exhibit homophily.
Removing all the cross-community links would lead us to identify the
nodes forming the different communities

If the threshold for cross-community links is 25%, then $0.3 > (25\%)(0.48)$
The network does not exhibit homophily.