Online Social Networks and Media

Graph Partitioning (cuts, spectral clustering, density),
Community evolution
modules, cluster, communities, groups, partitions
(more on this today)
Outline

PART I
1. Introduction: what, why, types?
2. Cliques and vertex similarity
3. Background: Cluster analysis
4. Hierarchical clustering (betweenness)
5. Modularity
6. How to evaluate
Outline

PART II

1. Cuts
2. Spectral Clustering
3. Dense Subgraphs

4. Community Evolution
Graph partitioning

The general problem

- Input: a graph \( G = (V, E) \)
  - edge \((u, v)\) denotes similarity between \(u\) and \(v\)
  - weighted graphs: \textit{weight} of edge captures the degree of similarity

Partitioning as an optimization problem:

- Partition the nodes in the graph such that nodes \textit{within clusters} are \textit{well interconnected} (high edge weights), and nodes \textit{across clusters} are \textit{sparser interconnected} (low edge weights)

- most graph partitioning problems are NP hard
Graph Partitioning
Graph Partitioning

Undirected graph $G(V, E)$:

Bi-partitioning task:
Divide vertices into two disjoint groups $A, B$

How can we define a “good” partition of $G$?
How can we efficiently identify such a partition?
Graph Partitioning

What makes a good partition?

- Maximize the number of within-group connections
- Minimize the number of between-group connections
Graph Cuts

Express *partitioning objectives* as a function of the “edge cut” of the partition

**Cut:** Set of edges with only one vertex in a group:

\[ \text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \]

![Graph with labeled vertices and edges](image)

\[ \text{cut}(A, B) = 2 \]
Min Cut

**min-cut**: the min number of edges such that when removed cause the graph to become disconnected

Minimizes the number of connections between partition

\[
\arg \min_{A,B} \text{cut}(A,B)
\]

This problem can be solved in polynomial time

Min-cut/Max-flow algorithm
An example
Min Cut

Problem:
  – Only considers external cluster connections
  – Does not consider internal cluster connectivity
Graph Bisection

• Since the minimum cut does not always yield good results we need extra constraints to make the problem meaningful.

• **Graph Bisection** refers to the problem of partitioning the nodes of the graph into two equal sets.

• **Kernighan-Lin algorithm**: Start with random equal partitions and then swap nodes to improve some quality metric (e.g., cut, modularity, etc).
Cut Ratio

Ratio Cut

Normalize cut by the *size* of the groups

\[
\text{Ratio-cut} = \frac{\text{Cut}(U,V-U)}{|U|} + \frac{\text{Cut}(U,V-U)}{|V-U|}
\]
Normalized Cut

Normalized-cut

Connectivity between groups relative to the density of each group

\[
\text{Normalized-cut} = \frac{\text{Cut}(U,V-U)}{\text{Vol}(U)} + \frac{\text{Cut}(U,V-U)}{\text{Vol}(V-U)}
\]

\(\text{Vol}(U)\): total weight of the edges with at least one endpoint in \(U\): \(\text{vol}(U) = \sum_{i \in U} d_i\)

Why use these criteria?

- Produce more balanced partitions
Red is Min-Cut

Ratio-Cut(Red) = $\frac{1}{1} + \frac{1}{8} = \frac{9}{8}$

Ratio-Cut(Green) = $\frac{2}{5} + \frac{2}{4} = \frac{18}{20}$

Normalized-Cut(Red) = $\frac{1}{1} + \frac{1}{27} = \frac{28}{27}$

Normalized-Cut(Green) = $\frac{2}{12} + \frac{2}{16} = \frac{14}{48}$

Normalized is even better for Green due to density
An example

Which of the three cuts has the best (min, normalized, ratio) cut?
Graph expansion:

$$\alpha = \min_U \frac{\text{cut}(U, V - U)}{\min\{|U|, |V - U|\}}$$

Graph conductance (similar for volume)
Graph Cuts

Ratio and normalized cuts can be reformulated in matrix format and solved using spectral clustering.
SPECTRAL CLUSTERING
Matrix Representation

Adjacency matrix \( (A) \):

- \( n \times n \) matrix
- \( A = [a_{ij}] \), \( a_{ij} = 1 \) if edge between node \( i \) and \( j \)

**Important properties:**
- Symmetric matrix
- Eigenvectors are real and orthogonal

If the graph is weighted, \( a_{ij} = w_{ij} \)
Matrix Representation

Degree matrix (D):

- $n \times n$ diagonal matrix

- $D = [d_{ii}]$, $d_{ii} =$ degree of node $i$
Matrix Representation

Laplacian matrix \((L)\):

\(- n \times n\) symmetric matrix

\[ L = D - A \]
Spectral Graph Partitioning

\( x \) is a vector in \( \mathbb{R}^n \) with components \((x_1, \ldots, x_n)\)

– Think of it as a label/value of each node of \( G \)

- What is the meaning of \( A \cdot x \)?

\[
\begin{bmatrix}
  a_{11} & \cdots & a_{1n} \\
  \vdots & \ddots & \vdots \\
  a_{n1} & \cdots & a_{nn}
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n
\end{bmatrix}
= 
\begin{bmatrix}
  y_1 \\
  \vdots \\
  y_n
\end{bmatrix}
\]

\[
y_i = \sum_{j=1}^{n} A_{ij} x_j = \sum_{(i,j) \in E} x_j
\]

Entry \( y_i \) is a sum of labels \( x_j \) of neighbors of \( i \)
Spectral Analysis

$i^{th}$ coordinate of $A \cdot x$:
- Sum of the $x$-values of neighbors of $i$
- Make this a new value at node $j$

Spectral Graph Theory:
- Analyze the “spectrum” of a matrix representing $G$
- Spectrum: Eigenvectors $x_i$ of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues $\lambda_i$: $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\}$ where $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$

Spectral clustering: use the eigenvectors of $A$ or graphs derived by it

Most based on the graph Laplacian

$$A \cdot x = \lambda \cdot x$$
Laplacian Matrix properties

• The matrix $L$ is symmetric and positive semi-definite
  – all eigenvalues of $L$ are positive
  positive definite: if $z^TMz$ is non-negative, for every non-zero column vector $z$

• The matrix $L$ has 0 as an eigenvalue, and corresponding eigenvector $w_1 = (1,1,...,1)$
  – $\lambda_1 = 0$ is the smallest eigenvalue

Proof: Let $w_1$ be the column vector with all 1s -- show $Lw_1 = 0w_1$
The second smallest eigenvalue (also known as Fielder value) $\lambda_2$ satisfies

$$\lambda_2 = \min_{x \perp w_1, \|x\| = 1} x^\top Lx$$
The second smallest eigenvalue

- For the Laplacian
  
  \[ x \perp w_1 \quad \implies \quad \sum_i x_i = 0 \]

- The expression:
  
  \[ x^T L x \]

  is

  \[ \sum_{(i,j) \in E} (x_i - x_j)^2 \]
The second smallest eigenvalue

Thus, the eigenvector for eigenvalue $\lambda_2$ (called the Fielder vector) minimizes

$$\min_{x \neq 0} \sum_{(i,j) \in E} (x_i - x_j)^2 \quad \text{where} \quad \sum_i x_i = 0$$

- Intuitively, minimum when $x_i$ and $x_j$ close whenever there is an edge between nodes $i$ and $j$ in the graph.
- $x$ must have some positive and some negative components.
Cuts + eigenvalues: intuition

- A *partition* of the graph by taking:
  - one set to be the nodes $i$ whose corresponding vector component $x_i$ is *positive* and
  - the other set to be the nodes whose corresponding vector component is *negative*.

- The *cut* between the two sets will have a small number of edges because $(x_i - x_j)^2$ is likely to be *smaller* if both $x_i$ and $x_j$ have the *same sign* than if they have different signs.

- Thus, *minimizing* $x^T L x$ under the required constraints will end giving $x_i$ and $x_j$ the same sign if there is an edge $(i, j)$. 
What we know about $x$?

- $x$ is unit vector: $\sum_i x_i^2 = 1$
- $x$ is orthogonal to 1$^{\text{st}}$ eigenvector $(1, \ldots, 1)$ thus:
  $\sum_i x_i \cdot 1 = \sum_i x_i = 0$

$$\lambda_2 = \min \sum_{(i,j) \in E} (x_i - x_j)^2 \over \sum_i x_i^2$$

All labelings of nodes $i$ so that $\sum x_i = 0$

We want to assign values $x_i$ to nodes $i$ such that few edges cross 0.
(we want $x_i$ and $x_j$ to subtract each other)

Balance to minimize
Example

![Graph Image]

<table>
<thead>
<tr>
<th>Eigenvalue</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenvector</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>-5</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
<td>-5</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-2</td>
<td>4</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Spectral Partitioning Algorithm

Three basic stages:

Pre-processing
- Construct a matrix representation of the graph

Decomposition
- Compute eigenvalues and eigenvectors of the matrix

Grouping
- Assign points to two or more clusters, based on the new representation
Spectral Partitioning Algorithm

Pre-processing:
Build Laplacian matrix $L$ of the graph

Decomposition:
– Find eigenvalues $\lambda$ and eigenvectors $x$ of the matrix $L$

– Map vertices to corresponding components of $\lambda_2$

1  2  3  4  5  6
1  3  -1 -1  0 -1  0
2  -1  2 -1  0  0  0
3  -1 -1  3 -1  0  0
4  0  0  -1  3 -1 -1
5  -1  0  0 -1  3 -1
6  0  0  0 -1 -1  2

$\lambda = \begin{pmatrix}
0.4 & 0.3 & -0.5 & -0.2 & -0.4 & -0.5 \\
0.4 & 0.6 & 0.4 & -0.4 & 0.4 & 0.0 \\
0.4 & 0.3 & 0.1 & 0.6 & -0.4 & 0.5 \\
0.4 & -0.3 & 0.1 & 0.6 & 0.4 & -0.5 \\
0.4 & -0.3 & -0.5 & -0.2 & 0.4 & 0.5 \\
0.4 & -0.6 & 0.4 & -0.4 & -0.4 & 0.0 
\end{pmatrix}$

$x = \begin{pmatrix}
1 \\
2 \\
3 \\
4 \\
5 \\
6 
\end{pmatrix} = \begin{pmatrix}
0.3 \\
0.6 \\
0.3 \\
-0.3 \\
-0.3 \\
-0.6 
\end{pmatrix}$

How do we now find the clusters?
Spectral Partitioning Algorithm

Grouping:
- Sort components of reduced 1-dimensional vector
- Identify clusters by splitting the sorted vector in two

• How to choose a splitting point?
  - Naïve approaches:
    • Split at 0 or median value
  - More expensive approaches:
    • Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td>-0.3</td>
</tr>
<tr>
<td>5</td>
<td>-0.3</td>
</tr>
<tr>
<td>6</td>
<td>-0.6</td>
</tr>
</tbody>
</table>

Split at 0:
- **Cluster A**: Positive points
- **Cluster B**: Negative points

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
<td>4</td>
<td>-0.3</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
<td>5</td>
<td>-0.3</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td>6</td>
<td>-0.6</td>
</tr>
</tbody>
</table>
Example: Spectral Partitioning

![Graph illustration with nodes and edges]

- Value of $x_2$
- Rank in $x_2$
**k-Way Spectral Clustering**

How do we partition a graph into $k$ clusters?

- **Recursively apply a bi-partitioning algorithm** in a hierarchical divisive manner
  - Disadvantages: Inefficient, unstable
$k$-Way Spectral Clustering

Use *several of the eigenvectors* to partition the graph.

- Use $m$ eigenvectors, and set a threshold for each,
- Get a partition into $2^m$ groups, each group consisting of the nodes that are above or below threshold for each of the eigenvectors, in a particular pattern.
If we use both the 2\textsuperscript{nd} and 3\textsuperscript{rd} eigenvectors, nodes 2 and 3 (negative in both) 5 and 6 (negative in 2\textsuperscript{nd}, positive in 3\textsuperscript{rd}) 1 and 4 alone

<table>
<thead>
<tr>
<th>Eigenvalue</th>
<th>0</th>
<th>1</th>
<th>3</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenvector</td>
<td>1</td>
<td>1</td>
<td>-5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>-2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
<td>-5</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-2</td>
<td>4</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

- Note that each eigenvector except the first is the vector \(x\) that minimizes \(x^\mathsf{T}Lx\), subject to the constraint that it is orthogonal to all previous eigenvectors.
- Thus, while each eigenvector tries to produce a minimum-sized cut, successive eigenvectors have to satisfy more and more constraints => the cuts progressively worse.
Other properties of $L$

Let $G$ be an undirected graph with non-negative weights. Then

- the multiplicity $k$ of the eigenvalue $0$ of $L$ equals the number of connected components $A_1, \ldots, A_k$ in the graph
- the eigenspace of eigenvalue $0$ is spanned by the indicator vectors $1A_1$, $\ldots$, $1A_k$ of those components
Proof (sketch)

If connected \( k = 1 \)

\[
0 = x^\top L x = \sum_{(i,j) \in E} (x_i - x_j)^2
\]

Assume \( k \) connected components, both \( A \) and \( L \) block diagonal, if we order vertices based on the connected component they belong to (recall the “tile” matrix)

\[
L = \begin{pmatrix} L_1 & & \\ & L_2 & \\ & & \ddots \end{pmatrix}
\]

\( L_i \) Laplacian of the \( i \)-th component

For all block diagonal matrices, the spectrum is given by the union of the spectra of each block, and the corresponding eigenvectors are the eigenvectors of the block, filled with 0 at the positions of the other blocks.
Normalized Graph Laplacians

\[ L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \]

\[ L_{rw} = D^{-1} L = I - D^{-1} W \]

\( L_{rw} \) closely connected to random walks (to be discussed in future lectures)

\[ x^\tau L_{sym} x = \sum_{(i,j) \in E} \left( \frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}} \right)^2 \]
Cuts and spectral clustering

\[
cut(A_1, \ldots, A_k) := \sum_{i=1}^{k} \text{cut}(A_i, \overline{A}_i)
\]

\[
\text{RatioCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A}_i)}{|A_i|}
\]

\[
\text{Ncut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A}_i)}{\text{vol}(A_i)}.
\]

Relaxing Ncut leads to normalized spectral clustering, while relaxing RatioCut leads to unnormalized spectral clustering.
Spectral Clustering

- Use the lowest $k$ eigenvalues of $L$ to construct the $nxk$ graph $G'$ that has these eigenvectors as columns

- The $n$-rows represent the graph vertices in a $k$-dimensional Euclidean space

- Group these vertices in $k$ clusters using $k$-means clustering or similar techniques
Spectral clustering (besides graphs)

Can be used to cluster any points (not just vertices), as long as an appropriate similarity matrix

Needs to be symmetric and non-negative

How to construct a graph:

• **ε-neighborhood graph**: connect all points whose pairwise distances are smaller than ε
• **k-nearest neighbor graph**: connect each point with each k nearest neighbor
• **full graph**: connect all points with weight in the edge (i, j) equal to the similarity of i and j
Summary

• The values of $x$ minimize

$$\min_{x \neq 0} \sum_{(i, j) \in E} (x_i - x_j)^2 \quad \sum_i x_i = 0$$

• For weighted matrices

$$\min_{x \neq 0} \sum_{(i, j)} A[i, j](x_i - x_j)^2 \quad \sum_i x_i = 0$$

• The ordering according to the $x_i$ values will group similar (connected) nodes together

• Physical interpretation: The stable state of springs placed on the edges of the graph
Thanks to Aris Gionis

MAXIMUM DENSEST SUBGRAPH
Finding dense subgraphs

• Dense subgraph: A collection of vertices such that there are a lot of edges between them
  – E.g., find the subset of email users that talk the most between them
  – Or, find the subset of genes that are most commonly expressed together

• Similar to community identification but we do not require that the dense subgraph is sparsely connected with the rest of the graph.
Definitions

• Input: undirected graph \( G = (V, E) \).
• Degree of node \( u \): \( \deg(u) \)
• For two sets \( S \subseteq V \) and \( T \subseteq V \):
  \[
  E(S, T) = \{(u, v) \in E : u \in S, v \in T\}
  \]
• \( E(S) = E(S, S) \): edges within nodes in \( S \)
• Graph Cut defined by nodes in \( S \subseteq V \):
  \( E(S, \bar{S}) \): edges between \( S \) and the rest of the graph
• Induced Subgraph by set \( S \) : \( G_S = (S, E(S)) \)
Definitions

• How do we define the density of a subgraph?

• Average Degree:

\[ d(S) = \frac{2|E(S)|}{|S|} \]

• Problem: Given graph G, find subset S, that maximizes density d(S)
  – Surprisingly there is a polynomial-time algorithm for this problem.
Min-Cut Problem

Given a graph* \( G = (V, E) \),
A source vertex \( s \in V \),
A destination vertex \( t \in V \)

Find a set \( S \subseteq V \)
Such that \( s \in S \) and \( t \in \overline{S} \)
That minimizes \( E(S, \overline{S}) \)

* The graph may be weighted

Min-Cut = Max-Flow: the minimum cut maximizes the flow that can be sent from \( s \) to \( t \). There is a polynomial time solution.
Decision problem

• Consider the decision problem
  – Is there a set $S$ with $d(S) \geq c$?

• $d(S) \geq c$

• $2|E(S)| \geq c|S|$

• $\sum_{v \in S} \deg(v) - E(S, \overline{S}) \geq c|S|$

• $2|E| - \sum_{v \in \overline{S}} \deg(v) - E(S, \overline{S}) \geq c|S|$

• $\sum_{v \in \overline{S}} \deg(v) + E(S, \overline{S}) + c|S| \leq 2|E|$
Transform to min-cut

• For a value $c$ we do the following transformation

• We ask for a min s-t cut in the new graph
Transformation to min-cut

- There is a cut that has value $2|E|$
Transformation to min-cut

- Every other cut has value:
- $\sum_{v \in S^c} \deg(v) + E(S, S^c) + c|S|$
Transformation to min-cut

- If $\sum_{v \in S} \deg(v) + E(S, \bar{S}) + c|S| \leq 2|E|$ then $S \neq \emptyset$ and $d(S) \geq c$
Algorithm (Goldberg)

Given the input graph $G$, and value $c$

1. Create the \textit{min-cut instance} graph
2. Compute the \textit{min-cut}
3. If the set $S$ is not empty, return \textbf{YES}
4. Else return \textbf{NO}

How do we find the set with \textit{maximum} density?
Min-cut algorithm

• The min-cut algorithm finds the optimal solution in polynomial time $O(nm)$, but this is too expensive for real networks.

• We will now describe a simpler approximation algorithm that is very fast
  – Approximation algorithm: the ratio of the density of the set produced by our algorithm and that of the optimal is bounded.
    • We will show that the ratio is at most $\frac{1}{2}$
    • The optimal set is at most twice as dense as that of the approximation algorithm.

• Any ideas for the algorithm?
Greedy Algorithm

Given the graph \( G = (V, E) \)
1. \( S_0 = V \)
2. For \( i = 1 \ldots |V| \)
   a. Find node \( v \in S \) with the minimum degree
   b. \( S_i = S_{i-1} \setminus \{v\} \)
3. Output the densest set \( S_i \)
Example
Analysis

• We will prove that the optimal set has density at most 2 times that of the set produced by the Greedy algorithm.

• Density of optimal set: \( d_{opt} = \max_{S \subseteq V} d(S) \)

• Density of greedy algorithm \( d_g \)

• We want to show that \( d_{opt} \leq 2 \cdot d_g \)
Upper bound

• We will first upper-bound the solution of optimal
• Assume an arbitrary assignment of an edge \((u, v)\) to either \(u\) or \(v\)

• Define:
  – \(IN(u)\) = # edges assigned to \(u\)
  – \(\Delta = \max_{u \in V} IN(u)\)

• We can prove that
  – \(d_{opt} \leq 2 \cdot \Delta\)

This is true for any assignment of the edges!
Lower bound

• We will now prove a **lower bound** for the density of the set produced by the **greedy** algorithm.

• For the lower bound we consider a **specific** assignment of the edges that we create as the greedy algorithm progresses:
  
  – When removing node \( u \) from \( S \), assign all the edges to \( u \)

• So: \( \text{IN}(u) = \text{degree of } u \text{ in } S \leq d(S) \leq d_g \)

• This is true for all \( u \) so \( \Delta \leq d_g \)

• It follows that \( d_{opt} \leq 2 \cdot d_g \)
The k-densest subgraph

- The **k-densest subgraph** problem: Find the set of $k$ nodes $S$, such that the density $d(S)$ is maximized.
  - The k-densest subgraph problem is **NP-hard**!
QUANTIFYING SOCIAL GROUP EVOLUTION

G Palla, AL Barabási, T Vicsek, *Nature* 446 (7136), 664-667
Datasets

- monthly *list of articles* in the Cornell University Library e-print condensed matter (cond-mat) archive spanning *142 months*, with over *30,000 authors*,

- *phone calls* between the customers of a mobile phone company spanning *52 weeks* (accumulated over two-week-long periods) containing the communication patterns of over *4 million users*. 
Datasets

black nodes/edges do not belong to any community,
red nodes belong to two or more communities
Datasets

Different local structure:

- **Co-authorship**: dense network with significant overlap among communities (co-authors of an article form cliques) -- **Phone-call**: communities less interconnected, often separated by one or more inter-community node/edge

- **Co-authorship** long-term collaborations -- **Phone-call**: the links correspond to instant communication events

Fundamental differences suggest that any common features represent potentially generic characteristics
Communities at each time step extracted using the clique percolation method (CPM)

Why CPM?
their members can be reached through well connected subsets of nodes, and communities may overlap

Parameters
$k = 4$
Weighted graph – use a weight threshold $w^*$ (links weaker than $w^*$ are ignored)
Evaluation

compare the average weight of the links inside communities, to the average weight of the inter-community links

$n_{\text{real}}$: the size of the largest subset of people having the same zipcode averaged over time steps and the set of available communities,

$n_{\text{rand}}$ represents the same average but with randomly selected users

$s$: size
Basic Events

- Growth
  - $t \rightarrow t + 1$

- Contraction
  - $t \rightarrow t + 1$

- Merging
  - $t \rightarrow t + 1$

- Splitting
  - $t \rightarrow t + 1$

- Birth
  - $t \rightarrow t + 1$

- Death
  - $t \rightarrow t + 1$
For each pair of consecutive time steps $t$ and $t+1$, construct a joint graph consisting of the union of links from the corresponding two networks, and extract the CPM community structure of this joint network.

- Any community from either the $t$ or the $t+1$ snapshot is contained in exactly one community in the joint graph.
- If a community in the joint graph contains a single community from $t$ and a single community from $t+1$, then they are matched.
- If the joint group contains more than one community from either time steps, the communities are matched in descending order of their relative node overlap.
Results

s: size

\( t \): age

s and t are positively correlated: larger communities are on average older
Results

Auto-correlation function

\[ C(t) \equiv \frac{|A(t_0) \cap A(t_0 + t)|}{|A(t_0) \cup A(t_0 + t)|} \]

where A(t) members of community A at t

- the collaboration network is more “dynamic” (decays faster)
- in both networks, the auto-correlation function decays faster for the larger communities, showing that the membership of the larger communities is changing at a higher rate.
Results

\[ \zeta = \frac{\sum_{t=t_0}^{t_{\text{max}}-1} C(t, t+1)}{t_{\text{max}} - t_0 - 1} \]

1-\(\zeta\): the average ratio of members changed in one step
\(\tau^*\): lifetime, stationarity \(\zeta\)
the average life-span \(<\tau^*\rangle\) (color coded) as a function of \(\zeta\) and \(s\)
  - for **small communities** optimal \(\zeta\) near 1, better to have static, time-independent
  - For **large communities**, the peak is shifted towards low \(\zeta\) values, better to have continually changing membership
Results
Results
Can we predict the evolution?

\[ w_{\text{out}}: \text{individual commitment to outside the community} \]
\[ w_{\text{in}}: \text{individual commitment inside the community} \]
\[ p: \text{probability to abandon the community} \]
Can we predict the evolution?

\( W_{\text{out}} \): total weight of links to nodes outside the community
\( W_{\text{in}} \): total weight of links inside the community
\( p \): probability of a community to disintegrate in the next step for co-authorship max lifetime at intermediate values
Conclusions

Significant *difference between smaller collaborative or friendship circles and institutions*.

- At the heart of **small communities** are **a few strong relationships**, and as long as these persist, the community around them is stable.

- The condition for stability of **large communities** is **continuous change**, so that after some time practically all members are exchanged.

- Loose, rapidly changing communities reminiscent of institutions, which can continue to exist even after all members have been replaced by new members (e.g., members of a school).
Basic References

Questions?