E6885 Network Science Lecture 3:  
*Network Partitioning, Clustering and Visualization*

Ching-Yung Lin, Dept. of Electrical Engineering, Columbia University  
September 24th, 2012
## Course Structure

<table>
<thead>
<tr>
<th>Class Date</th>
<th>Class Number</th>
<th>Topics Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/10/12</td>
<td>1</td>
<td>Overview of Network Science</td>
</tr>
<tr>
<td>09/17/12</td>
<td>2</td>
<td>Network Representations and Characteristics</td>
</tr>
<tr>
<td>09/24/12</td>
<td>3</td>
<td>Network Partitioning, Clustering and Visualization</td>
</tr>
<tr>
<td>10/01/12</td>
<td>4</td>
<td>Network Sampling and Estimation</td>
</tr>
<tr>
<td>10/08/12</td>
<td>5</td>
<td>Network Models</td>
</tr>
<tr>
<td>10/15/12</td>
<td>6</td>
<td>Network Topology Inference</td>
</tr>
<tr>
<td>10/22/12</td>
<td>7</td>
<td>Dynamic Networks - I</td>
</tr>
<tr>
<td>10/29/12</td>
<td>8</td>
<td>Dynamic Networks - II</td>
</tr>
<tr>
<td>11/12/12</td>
<td>9</td>
<td>Final Project Proposals</td>
</tr>
<tr>
<td>11/19/12</td>
<td>10</td>
<td>Analysis of Network Flow</td>
</tr>
<tr>
<td>11/26/12</td>
<td>11</td>
<td>Graphical Models and Bayesian Networks</td>
</tr>
<tr>
<td>12/03/12</td>
<td>12</td>
<td>Social and Economic Impact of Network Analysis</td>
</tr>
<tr>
<td>12/10/12</td>
<td>13</td>
<td>Large-Scale Network Processing System</td>
</tr>
<tr>
<td>12/17/12</td>
<td>14</td>
<td>Final Project Presentation</td>
</tr>
</tbody>
</table>
Graph Partitioning

- Many uses of graph partitioning:
  - E.g., community structure in social networks

- A cohesive subset of vertices generally is taken to refer to a subset of vertices that
  -(1) are well connected among themselves, and
  -(2) are relatively well separated from the remaining vertices

- Graph partitioning algorithms typically seek a partition of the vertex set of a graph
  in such a manner that the sets $E( C_k, C_{k'} )$ of edges connecting vertices in $C_k$
  to vertices in $C_{k'}$ are relatively small in size compared to the sets $E( C_k ) = E( C_k, C_{k'} )$
  of edges connecting vertices within $C_{k'}$. 
Example: Karate Club Network
Hierarchical Clustering Algorithms Types

- Primarily differ in [Jain et. al. 1999]:
  - (1) how they evaluate the quality of proposed clusters, and
  - (2) the algorithms by which they seek to optimize that quality.

- Agglomerative: successive coarsening of partitions through the process of merging.

- Divisive: successive refinement of partitions through the process of splitting.

- At each stage, the current candidate partition is modified in a way that minimizes a specific measure of cost.

- In agglomerative methods, the least costly merge of two previously existing partition elements is executed.

- In divisive methods, it is the least costly split of a single existing partition element into two that is executed.
Hierarchical Clustering

- The resulting hierarchy typically is represented in the form of a tree, called a *dendrogram*.

- The measure of cost incorporated into a hierarchical clustering method used in graph partitioning should reflect our sense of what defines a ‘cohesive’ subset of vertices.

- In agglomerative algorithms, given two sets of vertices $C_1$ and $C_2$, two standard approaches to assigning a similarity value to this pair of sets is to use the maximum (called single-linkage) or the minimum (called complete linkage) of the dissimilarity $x_{ij}$ over all pairs.

- Dissimilarities for subsets of vertices were calculated from the $x_{ij}$ using the extension of Ward (1963)’s method and the lengths of the branches in the dendrogram are in relative proportion to the changes in dissimilarity.

\[
x_{ij} = \frac{N_{v_i} \Delta N_{v_j}}{d(N_v) + d(N_v - 1)}
\]

- $x_{ij}$ is the “normalized” number of neighbors of $v_i$ and $v_j$ that are not shared.
- $N_v$ is the set of neighbors of a vertex.
- $\Delta$ is the symmetric difference of two sets which is the set of elements that are in one or the other but not both.

Other dissimilarity measures

- There are various other common choices of dissimilarity measures, such as:

\[ x_{ij} = \sqrt{\sum_{k \neq i, j} (A_{ik} - A_{jk})^2} \]

- Hierarchical clustering algorithms based on dissimilarities of this sort are reasonably efficient, running in \( O(N_v^2 \log N_v) \) time.
Hierarchical Clustering Example

Fig. 4.7 Hierarchical clustering of the karate club network.
Spectral Clustering

- Finding the eigenvectors of the adjacency matrix
- Starting with the largest (absolute) eigenvalues, the entries of each eigenvector are sorted.
- The vertices corresponding to particularly large or negative entries, in conjunction with their immediate neighbors, are declared to be a cluster.

\[ Ax_i = \lambda_i x_i \quad \text{where} \quad \lambda_1 \leq \cdots \leq \lambda_{N_v} \]

- The motivation is the fact that, in the case of a graph consisting of two \(d\)-regular graphs joined to each other by just a handful of vertices, the two largest eigenvalues will be roughly equal to \(d\), and the remaining eigenvalues will be of only \(O(d^{1/2})\) in magnitude \(\Rightarrow\) a gap in the spectrum of eigenvalues – a spectral gap.
Spectral clustering – cont’d

- The two eigenvectors corresponding to the largest two eigenvalues can be expected to have large positive entries on vertices of one sub-network and large negative entries on vertices of the other sub-network.

- In the Karate Club Network example, the first eigenvector appears to suggest a separation of the 1st and 34th actors, and some of their nearest neighbors from the rest of the actors.

- The second eigenvector in turn provides evidence that these two actors, and certain of their neighbors, should themselves be separated.
Spectral Analysis Example

Fig. 4.8 Spectral analysis of the karate club network. Left: the (absolute) eigenvalues. Right: the first two eigenvectors, where points are labeled according to the number of their corresponding actors, and colored according to the sub-group of each actor, as displayed in Figure 1.2.
Spectral Partitioning involves iterative bisection

- The smallest eigenvalue of the Laplacian matrix can be shown to be identically equal to zero. If we suspect a graph \( G \) to consist of nearly 2 components, we should expect the second smallest eigenvalue \( \lambda_2 \) to be close to zero.

- Fiedler, 1973, suggested partitioning vertices by separating them according to the sign of their entries in the corresponding eigenvector \( x_2 \).

- The vector \( x_2 \) is hence often called the Fiedler vector, and the corresponding eigenvalue \( \lambda_2 \), the Fiedler value, which is also known as the algebraic connectivity of the graph.
Spectral Bisection Example based on Fielder Vector

**Fig. 4.9** Plot of the Fiedler vector, $x_2$, for the karate club network graph. The horizontal line at $y = 0$ indicates the clustering suggested by spectral bisection. Color and shape of the symbols for individual actors correspond to those of Figure 1.2.
Network Visualization
Methods of Graph Visualization

- Drawing Graphs with Special Structure:
  - Planar Graphs
    - Orthogonal paths for edges
    - K-sided convex polygons for each cycle of length k
  - Trees
    - Layering of Edges
    - Horizontal and vertical edges (hv-layout)
    - Radiate outward on concentric circles (radial layout)
- Drawing Graphs Using Analogies to Physical Systems
**Tips for Graph Drawing**

- **Minimize Edge Crossing**
  - Planar Graph and Planar Drawing
  - Graph Planarity Test
    - $O(n)$ algorithm at best
    - Intuitive method: test for subdivision of $K_5$ or $K_{3,3}$

- **Minimize and Uniform Edge Length**
  - Minimize edge length
    - Minimize the total edge length
    - Minimize the maximum of edge length
  - Uniform edge length (for unweighted graph)
    - Minimize the variance of edge length
    - For weighted graph: edge length proportional to the weight
Tips for Graph Drawing

- Minimize and Uniform Edge Bends
  - Minimize total edge bends
  - Minimize the variance of edge bends

- Maximize Angular Resolution
  - Maximize the smallest angle between two edges incident on the same node

- Minimize Aspect Ratio
  - Adapt to the real-world screens

- Graph Symmetry

Graph with and without edge bend

Graph with different angles (45° and 30°)

Symmetrical and non-symmetrical graph
Graph Drawings

- Graph drawing without modern techniques
  - Hand-made images
  - Drawing with cluster/factor analysis, multidimensional scaling

- Graph drawing approaches according to elaborately-defined aesthetics
  - Each approach follows a list of aesthetics with precedence since:
    - Aesthetics conflict with each other
    - Hard to design algorithm to fit all the aesthetics

Hand-made drawing of friendships

Various modern drawing approaches
Examples of Tree Drawing

• Layering of Edges
• Horizontal and vertical edges (hv-layout)
• Radiate outward on concentric circles (radial layout)
Hierarchical Graph Drawing

- For dependency relationships abstracted as acyclic digraph
- Sugiyama-style 3-step layout approach

Hierarchical drawing of a citation network
Tree Drawing

- Orthogonal Graph Drawing
  - For circuit placement in VLSI and PCB board layout
  - Topology-Shape-Metrics approach

- Radial Graph Drawing
  - For rooted tree drawing
  - Foldable for hierarchical graphs

Radial Drawing

Orthogonal Drawing

$V = \{1, 2, 3, 4, 5, 6\}$
$E = \{(1, 4), (1, 5), (1, 6), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6)\}$
Force-Directed Graph Drawing

- Force-directed Graph Drawing
  - For undirected straight-line drawing
  - Simulate physical systems with forces, the final graph layout reaches the local system equilibrium (i.e. a local minimization of energy)
  - Generate graphs with good node uniformity and symmetric
  - Two types of force-directed algorithms
    - Spring Embedder by Eades and the seminar work by Fructerman and Reingold
    - Kamada and Kawai algorithm, trying to maintain graph theoretic distances between nodes, the recent solution by stress majorization is popular
  - Multi-Scale force-directed algorithms for large graphs

Force-directed layout with spring embedder
Algorithm

- Assign a measure of ‘force’ to each vertex.
- Iteratively adjust the position \( p_v = (x_v, y_v)^T \) until the forces on each have converged.
- A standard setup defines the force \( F = (F_x, F_y)^T \) on \( v \) in the \( x \) direction to be:

\[
F_x(v) = \sum_{\{u,v\} \in E} k_{uv}^{(1)} (p_u - p_v) \frac{x_v - x_u}{|p_u - p_v|^2} + \sum_{(u,v) \in V^2} k_{uv}^{(2)} \frac{x_v - x_u}{|p_u - p_v|^2},
\]

where

- \( k_{uv}^{(1)} \) is a spring stiffness
- \( k_{uv}^{(2)} \) is s strength of electric repulsion stiffness
- is Euclidean distance
- \( len_{uv} \) is a parameter for desired dist.
Social Network Visualization -- Metaphors

- Basic Metaphors with Graph
  - Nodes ~ People
    - Visual attribute mapping: hue, transparency, icon types, labels, etc.
  - Edges ~ Relationships
    - Visual attribute mapping: length, thickness, color, direction, labels, etc.

- Advanced Metaphors with Graph
  - Shading ~ Community (cluster)

- Other Non-Graph Metaphors
  - Adjacency Matrix
Social Network Visualization -- Matrix

- Adjacency Matrix as an Alternative of Node-Link Graph
  - Good for huge, scale-free graph
  - Hybrid visualization
    - MatrixExplorer: synchronized view of both adjacency matrix and node-link graph
    - NodeTrix: combined visualization for social network with communities

- User Studies on Performance
  - Matrix is better than node-link graph in several major tasks when node number is over 20
  - Negatives: the mass people (information consumer) can only perceive simple visualization such as maps, line-charts and graphs
Social Network Visualization – Huge Structural Network

- Social Networks could be quite large, posing challenge to visualization
  - Traditional layout algorithms do not scale, with $O(n^2) \sim O(n^3)$ computation complexity
  - With multi-scale fast layout algorithm, huge graph can be drawn with degraded quality, but raises severe visual clutter
- Social Networks are intrinsically clustered, with well-known feature of scale-free, small-world
  - Visualization should reveal this structural information

A Huge Graph Visualization with Multi-Scale Layout Algorithm (by AT&T)
Social Network Visualization — Huge Structural Network

- Graph Skeleton Visualization by Edge Pruning
  - Filter edges by edge betweenness centrality from low to high

Edge pruning with betweenness centrality
Social Network Visualization — Huge Structural Network

- Graph Skeleton Visualization by Edge Pruning
  - Construct a minimal spanning tree (MST) or path-finder network from the original huge graph

  - Graph skeleton visualization highlights the topology and intrinsic structure of huge graph by reducing visual clutter

  Skeleton extraction by constructing MST
Case study – Mapping ‘Science’

Fig 3.5 in Statistical Analysis of Network Data. Original source: Kevin Boyack.
Case study – Mapping ‘Science’

Fig 3.6 in Statistical Analysis of Network Data. Original source: Kevin Boyack.
Social Network Visualization – Huge Structural Network

- Graph Visual Clutter Reduction through Edge Bundling
  - Edge bundling steps
    - Generate control mesh
    - Cluster the edges
    - Let the clustered edge traverse through the same set of control points

The effectiveness of edge bundling
Social Network Visualization – Huge Structural Network

- Graph Visual Clutter Reduction through Edge Bundling
  - Hierarchical edge bundling
    - For hierarchically clustered graph
    - Maintain high-level topology information of edges

Hierarchical Edge Bundling
Social Network Visualization — Huge Structural Network

- Huge Graph Visualization through Advanced Interactions
  - Focus+Context: to highlight part of the network while leaving context
    - Topology Fisheye
    - Hyperbolic Tree

Focus + Context

![Topology Fisheye](image1)

![Hyperbolic Tree](image2)
Social Network Visualization — Huge Structural Network

- Huge Graph Visualization through Advanced Interactions
  - Overview + Detail
    - Overview graph with lowered quality
    - Overview graph show hierarchy information
  - Traditional Zoom & Pan

Overview + Detail
Social Network Visualization — Huge Structural Network

- Avoid huge graph with user-interest centric visualization
  - Ego-centric network visualization
    - Filter the huge graph by hop from the central node
    - Allow expansion with interactions
Social Network Visualization — Huge Structural Network

- Avoid huge graph with user-interest centric visualization
  - Spring-board by Frank Van Ham
    - The “search, show context, expand on Demand” paradigm
    - Degree of Interest (DOI) function combining graph topology (distance to the central node) and user interest (search context)
    - Allow further navigation by expanding a subgraph

Springboard: visualize huge graph by user-interest
Social Network Visualization — Temporal Network

- Dynamic Network Visualization with “Network Movie”
  - A Key Guideline: preserving user’s mental map during the visualization
    - Balance the static graph layout aesthetic and the graph stability between time frame
  - Use Staged Animation to Help User Perceive Complex Changes
    - Animation planning according to cognitive studies
    - Intermediate layout result had better satisfy graph aesthetics rather than interpolations
  - Some Negative Comments from User Studies
    - Animation leads to many participant errors, though participants find it enjoyable
    - Animation is the least effective form for analysis, even worse than static depictions
Social Network Visualization — Temporal Network

- Static Depiction of Network Evolutions (Still with Stable Layout)
Example – Wiki Revert Graph

- Understanding Social Dynamics in Wikipedia with Revert Graph Visualization
  - Revert Graph: user as the node and revert relationship as the link
  - Revert Graph Layout
    - Opposite to the traditional spring embedder
    - Links impose pushing forces and node attracts each other

Revert Graph for a Wiki Page
Example – Wiki Revert Graph

- Understanding Social Dynamics in Wikipedia with Revert Graph Visualization
  - This tool helps in discover:
    - Formation of opinion group
    - Pattern of mediation
    - Fighting of vandalism
    - Major controversial user and topic

Administrators (in the bottom group) are fighting vandalisms (in the upper group)
CAIDA router-level Internet data

- K-core decomposition

Fig 3.8 in Statistical Analysis of Network Data. Original source: Ignacio Alvarez-Hamelin
Case Study — SmallBlue

Overview

- SmallBlue is an IBM social search engine that enables IBMers to locate knowledgeable colleagues, communities, and knowledge networks within IBM.
- IBMers use their social networks everyday to help them in their daily work, SmallBlue automates such practice.
- SmallBlue is made up of:
  - A Client – installed by users who want to contribute their social network data to SmallBlue
  - A web-based tool suite - containing 4 tools: Ego, Find, Net, Reach
  - Back-end processing / search engine
Case Study – SmallBlue Net

- SmallBlue gauges the efficiency / health of enterprise network
  - Network by business unit / geography
  - Bridges – people who span / bridge clusters
  - Hubs – most connected people
  - Ability to identify (and correct) issues in collaboration, connectedness and knowledge flow across specific parts of the organisation

- Usage Scenario Includes
  - Visualize community membership
  - Succession planning – implication of loosing people
Case Study – SmallBlue Ego

- SmallBlue Ego helps IBMer understand and manage personal social network
  - Distance from central = closeness
  - Diversity of network = color coded for IBM business units
  - Location of network = pictures overlay on map
  - Shows relative social value of a person (how many people they could introduce you to)
  - Shows changes and trends
Acknowledgment

- Thanks to Dr. Lei Shi of IBM China Research Labs for providing several visualization slides in this lecture.
Questions?