Structure detection methods
Complex Networks, Course 303A, Spring, 2009

Prof. Peter Dodds
Department of Mathematics & Statistics
University of Vermont

Licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License.
Outline

Overview

Methods
  Hierarchy by aggregation
  Hierarchy by division
  Hierarchy by shuffling
  Spectral methods
  Hierarchies & Missing Links
  General structure detection

References
Structure detection

▶ The issue: how do we elucidate the internal structure of large networks across many scales?

▲ Zachary’s karate club [10, 7]
Structure detection

The issue: how do we elucidate the internal structure of large networks across many scales?

Zachary's karate club [10, 7]

Possible substructures: hierarchies, cliques, rings, ...

Plus:
All combinations of substructures.

Much focus on hierarchies...
Structure detection

The issue: how do we elucidate the internal structure of large networks across many scales?

Zachary’s karate club \[10, 7\]

Possible substructures: hierarchies, cliques, rings, ...

Plus:
All combinations of substructures.

Much focus on hierarchies...
Structure detection

Zachary’s karate club

Possible substructures:
- hierarchies, cliques, rings, …

Plus:
- All combinations of substructures.
- Much focus on hierarchies...

The issue:
how do we elucidate the internal structure of large networks across many scales?

References
Outline

Overview

Methods

Hierarchy by aggregation
Hierarchy by division
Hierarchy by shuffling
Spectral methods
Hierarchies & Missing Links
General structure detection

References
Hierarchy by division

Bottom up:

- **Idea:** Extract hierarchical classification scheme for $N$ objects by an agglomeration process.
- Need a measure of distance between all pairs of objects.
- Note: evidently works for non-networked data.
- **Procedure:**
  1. Order pair-based distances.
  2. Sequentially add links between nodes based on closeness.
  3. Use additional criteria to determine when clusters are meaningful.
- Clusters gradually emerge, likely with clusters inside of clusters.
- Call above property **Modularity.**
Hierarchy by division

Bottom up:

- **Idea:** Extract hierarchical classification scheme for $N$ objects by an agglomeration process.
- Need a measure of distance between all pairs of objects.
- Note: evidently works for non-networked data.
- **Procedure:**
  1. Order pair-based distances.
  2. Sequentially add links between nodes based on closeness.
  3. Use additional criteria to determine when clusters are meaningful.
- Clusters gradually emerge, likely with clusters inside of clusters.
- Call above property *Modularity*. 
Hierarchy by division

Bottom up:

- **Idea:** Extract hierarchical classification scheme for $N$ objects by an agglomeration process.
- Need a measure of distance between all pairs of objects.
- **Note:** evidently works for non-networked data.
- **Procedure:**
  1. Order pair-based distances.
  2. Sequentially add links between nodes based on closeness.
  3. Use additional criteria to determine when clusters are meaningful.

- Clusters gradually emerge, likely with clusters inside of clusters.
- Call above property **Modularity**.
Hierarchy by division

Bottom up:

► **Idea:** Extract hierarchical classification scheme for $N$ objects by an agglomeration process.

► **Need a measure of distance between all pairs of objects.**

► **Note:** evidently works for non-networked data.

► **Procedure:**
  1. Order pair-based distances.
  2. Sequentially add links between nodes based on closeness.
  3. Use additional criteria to determine when clusters are meaningful.

► Clusters gradually emerge, likely with clusters inside of clusters.

► Call above property **Modularity**.
Hierarchy by division

Bottom up:

► **Idea:** Extract hierarchical classification scheme for $N$ objects by an agglomeration process.

► **Need a measure of distance between all pairs of objects.**

► **Note:** evidently works for non-networked data.

► **Procedure:**

1. Order pair-based distances.
2. Sequentially add links between nodes based on closeness.
3. Use additional criteria to determine when clusters are meaningful.

► Clusters gradually emerge, likely with clusters inside of clusters.

► Call above property **Modularity.**
Hierarchy by division

Bottom up:

► Idea: Extract hierarchical classification scheme for $N$ objects by an agglomeration process.
► Need a measure of distance between all pairs of objects.
► Note: evidently works for non-networked data.
► Procedure:
  1. Order pair-based distances.
  2. Sequentially add links between nodes based on closeness.
  3. Use additional criteria to determine when clusters are meaningful.
► Clusters gradually emerge, likely with clusters inside of clusters.
► Call above property Modularity.
Hierarchy by division

Bottom up:

- **Idea:** Extract hierarchical classification scheme for \( N \) objects by an agglomeration process.
- Need a measure of distance between all pairs of objects.
- Note: evidently works for non-networked data.
- **Procedure:**
  1. Order pair-based distances.
  2. Sequentially add links between nodes based on closeness.
  3. Use additional criteria to determine when clusters are meaningful.
- Clusters gradually emerge, likely with clusters inside of clusters.
- Call above property **Modularity**.
Hierarchy by division

Bottom up:

- **Idea:** Extract hierarchical classification scheme for $N$ objects by an agglomeration process.
- **Need a measure of distance between all pairs of objects.**
- **Note:** evidently works for non-networked data.
- **Procedure:**
  1. Order pair-based distances.
  2. Sequentially add links between nodes based on closeness.
  3. Use additional criteria to determine when clusters are meaningful.
- Clusters gradually emerge, likely with clusters inside of clusters.
- Call above property **Modularity.**
Hierarchy by division

Bottom up problems:

▶ Tend to plainly not work on data sets with known modular structures.
▶ Good at finding cores of well-connected (or similar) nodes... but fail to cope well with peripheral, in-between nodes.
Hierarchy by division

Bottom up problems:

- Tend to plainly not work on data sets with known modular structures.
- Good at finding cores of well-connected (or similar) nodes... but fail to cope well with peripheral, in-between nodes.
Outline

Overview

Methods

Hierarchy by aggregation
Hierarchy by division
Hierarchy by shuffling
Spectral methods
Hierarchies & Missing Links
General structure detection

References
Hierarchy by division

Top down:

- **Idea:** Identify global structure first and recursively uncover more detailed structure.

- **Basic objective:** Find dominant components that have significantly more links within than without, as compared to randomized version.

- We’ll first work through “Finding and evaluating community structure in networks” by Newman and Girvan (PRE, 2004). \[7\]

- See also
  1. “Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality” by Newman (PRE, 2001). \[5, 6\]
  2. “Community structure in social and biological networks” by Girvan and Newman (PNAS, 2002). \[3\]
Hierarchy by division

Top down:

- **Idea:** Identify global structure first and recursively uncover more detailed structure.
- **Basic objective:** find dominant components that have significantly more links within than without, as compared to randomized version.

- See also
Hierarchy by division

Top down:

- **Idea:** Identify **global structure first** and recursively uncover more detailed structure.

- **Basic objective:** find dominant components that have significantly more links within than without, as compared to randomized version.


- See also
Hierarchy by division

Top down:

- **Idea:** Identify global structure first and recursively uncover more detailed structure.
- **Basic objective:** find dominant components that have significantly more links within than without, as compared to randomized version.
- See also
Hierarchy by division

Top down:

- **Idea**: Identify global structure first and recursively uncover more detailed structure.

- **Basic objective**: find dominant components that have significantly more links within than without, as compared to randomized version.


- See also
Idea:
Edges that connect communities have higher betweenness than edges within communities.
Hierarchy by division

One class of structure-detection algorithms:

1. Compute edge betweenness for whole network.
2. Remove edge with highest betweenness.
3. Recompute edge betweenness.
4. Repeat steps 2 and 3 until all edges are removed.
5. Record when components appear as a function of # edges removed.
6. Generate dendogram revealing hierarchical structure.
Hierarchy by division

One class of structure-detection algorithms:

1. Compute edge betweenness for whole network.
2. **Remove** edge with highest betweenness.
3. Recompute edge betweenness
4. Repeat steps 2 and 3 until all edges are removed.

5. Record when components appear as a function of # edges removed.
6. Generate dendogram revealing hierarchical structure.
Hierarchy by division

One class of structure-detection algorithms:

1. Compute edge betweenness for whole network.
2. **Remove** edge with highest betweenness.
3. Recompute edge betweenness
4. Repeat steps 2 and 3 until all edges are removed.
5. Record when components appear as a function of # edges removed.
6. Generate **dendogram** revealing hierarchical structure.
Hierarchy by division

One class of structure-detection algorithms:

1. Compute edge betweenness for whole network.
2. **Remove** edge with highest betweenness.
3. Recompute edge betweenness
4. Repeat steps 2 and 3 until all edges are removed.

5. Record when components appear as a function of # edges removed.
6. Generate **dendogram** revealing hierarchical structure.
Hierarchy by division

One class of structure-detection algorithms:

1. Compute edge betweenness for whole network.
2. Remove edge with highest betweenness.
3. Recompute edge betweenness.
4. Repeat steps 2 and 3 until all edges are removed.

5. Record when components appear as a function of # edges removed.

6. Generate dendogram revealing hierarchical structure.
Hierarchy by division

One class of structure-detection algorithms:

1. Compute edge betweenness for whole network.
2. Remove edge with highest betweenness.
3. Recompute edge betweenness
4. Repeat steps 2 and 3 until all edges are removed.
5. Record when components appear as a function of # edges removed.
6. Generate dendogram revealing hierarchical structure.

Red line indicates appearance of four (4) components at a certain level.
Hierarchy by division

One class of structure-detection algorithms:

1. Compute edge betweenness for whole network.
2. Remove edge with highest betweenness.
3. Recompute edge betweenness
4. Repeat steps 2 and 3 until all edges are removed.
5. Record when components appear as a function of # edges removed.
6. Generate dendogram revealing hierarchical structure.

Red line indicates appearance of four (4) components at a certain level.
Hierarchy by division

Key element:

- Recomputing betweenness.
- **Reason:** Possible to have a low betweenness in links that connect large communities if other links carry majority of shortest paths.

When to stop?:

- How do we know which divisions are meaningful?
- **Modularity measure:** difference in fraction of within component nodes to that expected for randomized version:

$$Q = \sum_i \left[ e_{ij} - \left( \sum_j e_{ij} \right)^2 \right] = \text{Tr} E - ||E^2||_1,$$

where $e_{ij}$ is the fraction of edges between identified communities $i$ and $j$. 
Hierarchy by division

Key element:

- Recomputing betweenness.
- **Reason:** Possible to have a low betweenness in links that connect large communities if other links carry majority of shortest paths.

When to stop?:

- How do we know which divisions are meaningful?
- **Modularity measure:** difference in fraction of within component nodes to that expected for randomized version:

$$Q = \sum \left[ e_{ij} - (\sum e_{ij})^2 \right] = Tr E - ||E^2||,$$

where $e_{ij}$ is the fraction of edges between identified communities $i$ and $j$. 
Hierarchy by division

Key element:

- Recomputing betweenness.
- **Reason:** Possible to have a low betweenness in links that connect large communities if other links carry majority of shortest paths.

When to stop?:

- How do we know which divisions are meaningful?
- **Modularity measure:** difference in fraction of within component nodes to that expected for randomized version:

\[
Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] = \text{Tr}E - \|E^2\|_1,
\]

where \(e_{ij}\) is the fraction of edges between identified communities \(i\) and \(j\).
Hierarchy by division

Key element:
- Recomputing betweenness.
- **Reason**: Possible to have a low betweenness in links that connect large communities if other links carry majority of shortest paths.

When to stop?:
- How do we know which divisions are meaningful?
  - **Modularity measure**: difference in fraction of within component nodes to that expected for randomized version:

\[ Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] = \text{Tr}E - \|E^2\|_1, \]

where \( e_{ij} \) is the fraction of edges between identified communities \( i \) and \( j \).
Hierarchy by division

Key element:

- Recomputing betweenness.
- **Reason:** Possible to have a low betweenness in links that connect large communities if other links carry majority of shortest paths.

When to stop?:

- How do we know which divisions are meaningful?
- **Modularity measure:** difference in fraction of within component nodes to that expected for randomized version:

\[
Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] = \text{Tr}E - \|E^2\|_1,
\]

where \(e_{ij}\) is the fraction of edges between identified communities \(i\) and \(j\).
Hierarchy by division

Key element:

- Recomputing betweenness.
- **Reason:** Possible to have a low betweenness in links that connect large communities if other links carry majority of shortest paths.

When to stop?:

- How do we know which divisions are meaningful?
- **Modularity measure:** difference in fraction of within component nodes to that expected for randomized version:

\[ Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] = \text{Tr}E - \|E^2\|_1, \]

where \( e_{ij} \) is the fraction of edges between identified communities \( i \) and \( j \).
Hierarchy by division

Test case:

- Generate random community-based networks.
  - $N = 128$ with four communities of size 32.
  - Add edges randomly within and across communities.
  - Example:
    \[
    \langle k \rangle_{\text{in}} = 6 \quad \text{and} \quad \langle k \rangle_{\text{out}} = 2.
    \]
Hierarchy by division

Test case:

- Generate random community-based networks.
- \( N = 128 \) with four communities of size 32.
- Add edges randomly within and across communities.
- Example:

  \[ \langle k \rangle_{\text{in}} = 6 \quad \text{and} \quad \langle k \rangle_{\text{out}} = 2. \]
Hierarchy by division

Test case:
- Generate random community-based networks.
- $N = 128$ with four communities of size 32.
- Add edges randomly within and across communities.
- Example:
  \[
  \langle k \rangle_{\text{in}} = 6 \quad \text{and} \quad \langle k \rangle_{\text{out}} = 2.
  \]
Hierarchy by division

Test case:

- Generate random community-based networks.
- \( N = 128 \) with four communities of size 32.
- Add edges randomly within and across communities.
- Example:
  \[ \langle k \rangle_{\text{in}} = 6 \quad \text{and} \quad \langle k \rangle_{\text{out}} = 2. \]
A. Tests on computer-generated networks

First, as a controlled test of how well our algorithms perform, we have generated networks with known community structure, to see if the algorithms can recognize and extract this structure.

We have generated a large number of graphs with \( n/1005128 \) vertices, divided into four communities of 32 vertices each. Edges were placed independently at random between vertex pairs with probability \( p_{in} \) for an edge to fall between vertices in the same community and \( p_{out} \) to fall between vertices in different communities. The values of \( p_{in} \) and \( p_{out} \) were chosen to make the expected degree of each vertex equal to 16. In Fig. 6, we show a typical dendrogram from the analysis of such a graph using the shortest-path betweenness version of our algorithm.

In fact, for the sake of clarity, the figure is for a 64-node version of the graph. Results for the random-walk version are similar. At the right of the figure we also show the modularity, Eq. 5, for the same calculation, plotted as a function of position in the dendrogram. That is, the plot is aligned with the dendrogram so that one can read off modularity values for different divisions of the network directly. As we can see, the modularity has a single clear peak at the point where the network breaks into four communities, as we would expect. The peak value is around 0.5, which is typical.

In Fig. 7, we show the fraction of vertices in our computer-generated network sample classified correctly into the four communities by our algorithms, as a function of the mean number \( z_{out} \) of edges from each vertex to vertices in other communities. As the figure shows, both the shortest-path and random-walk versions of the algorithm perform excellently, with more than 90% of all vertices classified correctly from \( z_{out} \) all the way to around \( z_{out} \). Only for \( z_{out} \) does the classification begin to deteriorate markedly. In other words, our algorithm correctly identifies the community structure in the network almost all the way to the point \( z_{out} \) at which each vertex has on average the same number of connections to vertices outside its community as it does to those inside.

The shortest-path version of the algorithm does, however, perform noticeably better than the random-walk version, especially for the more difficult cases where \( z_{out} \) is large. Given that the random-walk algorithm is also more computationally demanding, there seems little reason to use it rather than the shortest-path algorithm, and hence, as discussed previously, we recommend the latter for most applications.
Hierarchy by division

- Maximum modularity $Q \approx 0.5$ obtained when four communities are uncovered.
- Further ‘discovery’ of internal structure is somewhat meaningless, as any communities arise accidentally.
Hierarchical by division

- Factions in Zachary's karate club network. [10]
Betweenness for electrons:

- Unit resistors on each edge.
- For every pair of nodes $s$ (source) and $t$ (sink), set up unit currents in at $s$ and out at $t$.
- Measure absolute current along each edge $\ell$, $|I_{\ell,st}|$.

- Sum $|I_{\ell,st}|$ over all pairs of nodes to obtain electronic betweenness for edge $\ell$.
- (Equivalent to random walk betweenness.)
- Electronic betweenness for edge between nodes $i$ and $j$:

$$B_{ij}^{\text{elec}} = a_{ij}|V_i - V_j|.$$
Betweenness for electrons:

- Unit resistors on each edge.
- For every pair of nodes $s$ (source) and $t$ (sink), set up unit currents in at $s$ and out at $t$.
- Measure absolute current along each edge $\ell$, $|I_{\ell,st}|$.
- Sum $|I_{\ell,st}|$ over all pairs of nodes to obtain electronic betweenness for edge $\ell$.
- (Equivalent to random walk betweenness.)
- Electronic betweenness for edge between nodes $i$ and $j$:

$$B_{ij}^{\text{elec}} = a_{ij}|V_i - V_j|.$$
Betweenness for electrons:

- Unit resistors on each edge.
- For every pair of nodes \( s \) (source) and \( t \) (sink), set up unit currents in at \( s \) and out at \( t \).
- Measure absolute current along each edge \( \ell \), \( |I_{\ell,st}| \).
- Sum \( |I_{\ell,st}| \) over all pairs of nodes to obtain electronic betweenness for edge \( \ell \).
- (Equivalent to random walk betweenness.)
- Electronic betweenness for edge between nodes \( i \) and \( j \):

\[
B_{ij}^\text{elec} = a_{ij} |V_i - V_j|.
\]
Betweenness for electrons:

- Unit resistors on each edge.
- For every pair of nodes $s$ (source) and $t$ (sink), set up unit currents in at $s$ and out at $t$.
- Measure absolute current along each edge $\ell$, $|I_{\ell,st}|$.

- Sum $|I_{\ell,st}|$ over all pairs of nodes to obtain electronic betweenness for edge $\ell$.

(Equivalent to random walk betweenness.)

Electronic betweenness for edge between nodes $i$ and $j$:

$$B_{ij}^{\text{elec}} = a_{ij}|V_i - V_j|.$$

References

- Methods
  - Hierarchy by aggregation
  - Hierarchy by division
  - Hierarchy by shuffling
  - Spectral methods
  - Hierarchies & Missing Links
  - General structure detection
Betweenness for electrons:

- Unit resistors on each edge.
- For every pair of nodes \( s \) (source) and \( t \) (sink), set up unit currents in at \( s \) and out at \( t \).
- Measure absolute current along each edge \( \ell \), \( |I_{\ell,st}| \).

- Sum \( |I_{\ell,st}| \) over all pairs of nodes to obtain electronic betweenness for edge \( \ell \).
- (Equivalent to random walk betweenness.)
- Electronic betweenness for edge between nodes \( i \) and \( j \):

\[
B_{ij}^{\text{elec}} = a_{ij}|V_i - V_j|.
\]
Betweenness for electrons:

- Unit resistors on each edge.
- For every pair of nodes \( s \) (source) and \( t \) (sink), set up unit currents in at \( s \) and out at \( t \).
- Measure absolute current along each edge \( \ell \), \( |I_{\ell,st}| \).

- Sum \( |I_{\ell,st}| \) over all pairs of nodes to obtain electronic betweenness for edge \( \ell \).
- (Equivalent to random walk betweenness.)
- Electronic betweenness for edge between nodes \( i \) and \( j \):
  \[
  B_{ij}^{\text{elec}} = a_{ij} |V_i - V_j|.
  \]
Electronic betweenness

- Define some arbitrary voltage reference.
  - Kirchhoff’s laws: current flowing out of node $i$ must balance:
    \[ \sum_{j=1}^{N} \frac{1}{R_{ij}} (V_j - V_i) = \delta_{is} - \delta_{it}. \]
  - Between connected nodes, $R_{ij} = 1 = a_{ij} = 1/a_{ij}$.
  - Between unconnected nodes, $R_{ij} = \infty = 1/a_{ij}$.
  - We can therefore write:
    \[ \sum_{j=1}^{N} a_{ij} (V_i - V_j) = \delta_{is} - \delta_{it}. \]
  - Some gentle jiggery pokery on the left hand side:
    \[ \sum_j a_{ij} (V_i - V_j) = V_i \sum_j a_{ij} - \sum_j a_{ij} V_j \\
    = V_i k_i - \sum_j a_{ij} V_j = k_i \delta_{ij} V_j - \sum_j a_{ij} V_j = [(K - A) \vec{V}]_i. \]
Electronic betweenness

- Define some arbitrary voltage reference.
- Kirchoff’s laws: current flowing out of node $i$ must balance:

$$\sum_{j=1}^{N} \frac{1}{R_{ij}} (V_j - V_i) = \delta_{is} - \delta_{it}.$$ 

- Between connected nodes, $R_{ij} = 1 = a_{ij} = 1/a_{ij}$.
- Between unconnected nodes, $R_{ij} = \infty = 1/a_{ij}$.
- We can therefore write:

$$\sum_{j=1}^{N} a_{ij} (V_i - V_j) = \delta_{is} - \delta_{it}.$$ 

- Some gentle jiggery pokery on the left hand side:

$$\sum_j a_{ij} (V_i - V_j) = V_i \sum_j a_{ij} - \sum_j a_{ij} V_j$$
$$= V_i k_i - \sum_j a_{ij} V_j = k_i \delta_{ij} V_j - \sum_j a_{ij} V_j = [(K - A) \vec{V}]_i$$.
Electronic betweenness

- Define some arbitrary voltage reference.
- Kirchoff’s laws: current flowing out of node $i$ must balance:

$$\sum_{j=1}^{N} \frac{1}{R_{ij}}(V_j - V_i) = \delta_{is} - \delta_{it}.$$ 

- Between connected nodes, $R_{ij} = 1 = a_{ij} = 1/a_{ij}$.
- Between unconnected nodes, $R_{ij} = \infty = 1/a_{ij}$.
- We can therefore write:

$$\sum_{j=1}^{N} a_{ij}(V_i - V_j) = \delta_{is} - \delta_{it}.$$ 

- Some gentle jiggery pokery on the left hand side:

$$\sum_{j} a_{ij}(V_i - V_j) = V_i \sum_{j} a_{ij} - \sum_{j} a_{ij} V_j$$

$$= V_i k_i - \sum_{j} a_{ij} V_j = k_i \delta_{ij} V_j - \sum_{j} a_{ij} V_j = [(K - A) \vec{V}]_i$$
Electronic betweenness

- Define some arbitrary voltage reference.
- Kirchoff’s laws: current flowing out of node $i$ must balance:
  \[ \sum_{j=1}^{N} \frac{1}{R_{ij}} (V_j - V_i) = \delta_{is} - \delta_{it}. \]

- Between connected nodes, $R_{ij} = 1 = a_{ij} = 1/a_{ij}$.
- Between unconnected nodes, $R_{ij} = \infty = 1/a_{ij}$.
- We can therefore write:
  \[ \sum_{j=1}^{N} a_{ij} (V_i - V_j) = \delta_{is} - \delta_{it}. \]

- Some gentle jiggery pokery on the left hand side:
  \[ \sum_j a_{ij} (V_i - V_j) = V_i \sum_j a_{ij} - \sum_j a_{ij} V_j = V_i k_i - \sum_j a_{ij} V_j = k_i \delta_{ij} V_j - \sum_j a_{ij} V_j = [(K - A) \vec{V}]_i. \]
Electronic betweenness

- Define some arbitrary voltage reference.
- Kirchoff’s laws: current flowing out of node \( i \) must balance:

\[
\sum_{j=1}^{N} \frac{1}{R_{ij}} (V_j - V_i) = \delta_{is} - \delta_{it}.
\]

- Between connected nodes, \( R_{ij} = 1 = a_{ij} = 1/a_{ij} \).
- Between unconnected nodes, \( R_{ij} = \infty = 1/a_{ij} \).
- We can therefore write:

\[
\sum_{j=1}^{N} a_{ij} (V_i - V_j) = \delta_{is} - \delta_{it}.
\]

- Some gentle jiggery pokery on the left hand side:

\[
\sum_j a_{ij} (V_i - V_j) = V_i \sum_j a_{ij} - \sum_j a_{ij} V_j
= V_i k_i - \sum_j a_{ij} V_j = k_i \delta_{ij} V_j - \sum_j a_{ij} V_j = [(K - A) \vec{V}]_i
\]
Electronic betweenness

- Define some arbitrary voltage reference.
- Kirchoff’s laws: current flowing out of node $i$ must balance:
  \[ \sum_{j=1}^{N} \frac{1}{R_{ij}} (V_j - V_i) = \delta_{is} - \delta_{it}. \]

- Between connected nodes, $R_{ij} = 1 = a_{ij} = 1/a_{ij}$.
- Between unconnected nodes, $R_{ij} = \infty = 1/a_{ij}$.
- We can therefore write:
  \[ \sum_{j=1}^{N} a_{ij} (V_i - V_j) = \delta_{is} - \delta_{it}. \]

- Some gentle jiggery pokery on the left hand side:
  \[ \sum_j a_{ij} (V_i - V_j) = V_i \sum_j a_{ij} - \sum_j a_{ij} V_j = V_i k_i - \sum_j a_{ij} V_j = k_i \delta_{ij} V_j - \sum_j a_{ij} V_j = [(K - A) \vec{V}]_i \]
Electronic betweenness

- Define some arbitrary voltage reference.
- Kirchoff’s laws: current flowing out of node $i$ must balance:

$$\sum_{j=1}^{N} \frac{1}{R_{ij}} (V_j - V_i) = \delta_{is} - \delta_{it}.$$ 

- Between connected nodes, $R_{ij} = 1 = a_{ij} = 1/a_{ij}$.
- Between unconnected nodes, $R_{ij} = \infty = 1/a_{ij}$.
- We can therefore write:

$$\sum_{j=1}^{N} a_{ij} (V_i - V_j) = \delta_{is} - \delta_{it}.$$ 

- Some gentle jiggery pokery on the left hand side:

$$\sum_j a_{ij} (V_i - V_j) = V_i \sum_j a_{ij} - \sum_j a_{ij} V_j = V_i k_i - \sum_j a_{ij} V_j = k_i \delta_{ij} V_j - \sum_j a_{ij} V_j = [(K - A) \tilde{V}]_i$$
Electronic betweenness

- Define some arbitrary voltage reference.
- Kirchoff’s laws: current flowing out of node $i$ must balance:

$$\sum_{j=1}^{N} \frac{1}{R_{ij}} (V_j - V_i) = \delta_{is} - \delta_{it}.$$ 

- Between connected nodes, $R_{ij} = 1 = a_{ij} = 1/a_{ij}$.
- Between unconnected nodes, $R_{ij} = \infty = 1/a_{ij}$.
- We can therefore write:

$$\sum_{j=1}^{N} a_{ij} (V_i - V_j) = \delta_{is} - \delta_{it}.$$ 

- Some gentle jiggery pokery on the left hand side:

$$\sum_j a_{ij} (V_i - V_j) = V_i \sum_j a_{ij} - \sum_j a_{ij} V_j$$

$$= V_i k_i - \sum_j a_{ij} V_j = k_i \delta_{ij} V_j - \sum_j a_{ij} V_j = [(K - A) \vec{V}]_i.$$
Electronic betweenness

- Define some arbitrary voltage reference.
- Kirchoff’s laws: current flowing out of node $i$ must balance:
  \[ \sum_{j=1}^{N} \frac{1}{R_{ij}} (V_j - V_i) = \delta_{is} - \delta_{it}. \]

- Between connected nodes, $R_{ij} = 1 = a_{ij} = 1/a_{ij}$.
- Between unconnected nodes, $R_{ij} = \infty = 1/a_{ij}$.
- We can therefore write:
  \[ \sum_{j=1}^{N} a_{ij} (V_i - V_j) = \delta_{is} - \delta_{it}. \]

- Some gentle jiggery pokery on the left hand side:
  \[ \sum_j a_{ij} (V_i - V_j) = V_i \sum_j a_{ij} - \sum_j a_{ij} V_j \]
  \[ = V_i k_i - \sum_j a_{ij} V_j = k_i \delta_{ij} V_j - \sum_j a_{ij} V_j = [(K - A) \vec{V}]_i \]
Electronic betweenness

- Define some arbitrary voltage reference.
- Kirchoff’s laws: current flowing out of node $i$ must balance:
  \[ \sum_{j=1}^{N} \frac{1}{R_{ij}} (V_j - V_i) = \delta_{is} - \delta_{it}. \]
- Between connected nodes, $R_{ij} = 1 = a_{ij} = 1/a_{ij}$.
- Between unconnected nodes, $R_{ij} = \infty = 1/a_{ij}$.
- We can therefore write:
  \[ \sum_{j=1}^{N} a_{ij} (V_i - V_j) = \delta_{is} - \delta_{it}. \]
- Some gentle jiggery pokery on the left hand side:
  \[ \sum_j a_{ij} (V_i - V_j) = V_i \sum_j a_{ij} - \sum_j a_{ij} V_j = V_i k_i - \sum_j a_{ij} V_j = k_i \delta_{ij} V_j - \sum_j a_{ij} V_j = [(K - A) \vec{V}]_i. \]
Electronic betweenness

- Write right hand side as $[I^\text{ext}]_i = \delta_{is} - \delta_{it}$, where $I^\text{ext}$ holds external source and sink currents.

- Matrixingly then:

$$ (K - A) \vec{V} = I^\text{ext}. $$

- $L = K - A$ is a beast of some utility—known as the Laplacian.

- Solve for voltage vector $\vec{V}$ by LU decomposition (Gaussian elimination).

- Do not compute an inverse!

- Note: voltage offset is arbitrary so no unique solution.

- Presuming network has one component, null space of $K - A$ is one dimensional.

- In fact, $\mathcal{N}(K - A) = \{c\vec{1}, c \in \mathbb{R}\}$ since $(K - A)\vec{1} = \vec{0}$. 
Electronic betweenness

- Write right hand side as \([I^\text{ext}]_i = \delta_{is} - \delta_{it}\), where \(I^\text{ext}\) holds external source and sink currents.
- Matrixingly then:

\[
(K - A) \vec{V} = I^\text{ext}.
\]

- \(L = K - A\) is a beast of some utility—known as the Laplacian.
- Solve for voltage vector \(\vec{V}\) by \(LU\) decomposition (Gaussian elimination).
- Do not compute an inverse!
- Note: voltage offset is arbitrary so no unique solution.
- Presuming network has one component, null space of \(K - A\) is one dimensional.
- In fact, \(\mathcal{N}(K - A) = \{c \vec{1}, c \in R\}\) since \((K - A) \vec{1} = \vec{0}\).
Electronic betweenness

- Write right hand side as $[I^{\text{ext}}]_i = \delta_{is} - \delta_{it}$, where $I^{\text{ext}}$ holds external source and sink currents.
- Matrixically then:

$$\mathbf{(K - A)\bar{V}} = I^{\text{ext}}.$$

- $\mathbf{L = K - A}$ is a beast of some utility—known as the Laplacian.
- Solve for voltage vector $\mathbf{\bar{V}}$ by LU decomposition (Gaussian elimination).
- Do not compute an inverse!
- Note: voltage offset is arbitrary so no unique solution.
- Presuming network has one component, null space of $\mathbf{K - A}$ is one dimensional.
- In fact, $\mathcal{N}(\mathbf{K - A}) = \{c\mathbf{1}, c \in \mathbb{R}\}$ since $(\mathbf{K - A})\mathbf{1} = \mathbf{0}$. 
Electronic betweenness

- Write right hand side as \([I^\text{ext}]_i = \delta_{is} - \delta_{it}\), where \(I^\text{ext}\) holds external source and sink currents.
- Matrixingly then:

\[(K - A)\vec{V} = I^\text{ext}.\]

- \(L = K - A\) is a beast of some utility—known as the Laplacian.
- Solve for voltage vector \(\vec{V}\) by \textbf{LU decomposition} (Gaussian elimination).
- Do not compute an inverse!
- \textbf{Note:} voltage offset is arbitrary so no unique solution.
- Presuming network has one component, null space of \(K - A\) is one dimensional.
- In fact, \(\mathcal{N}(K - A) = \{c\vec{1}, c \in R\}\) since \((K - A)\vec{1} = \vec{0}\).
Electronic betweenness

- Write right hand side as \([I^{\text{ext}}]_i = \delta_{is} - \delta_{it}\), where \(I^{\text{ext}}\) holds external source and sink currents.

- Matrixingly then:

\[
(K - A) \vec{V} = I^{\text{ext}}.
\]

- \(L = K - A\) is a beast of some utility—known as the Laplacian.

- Solve for voltage vector \(\vec{V}\) by \(\text{LU decomposition}\) (Gaussian elimination).

- Do not compute an inverse!

- Note: voltage offset is arbitrary so no unique solution.

- Presuming network has one component, null space of \(K - A\) is one dimensional.

- In fact, \(\mathcal{N}(K - A) = \{c\vec{1}, c \in \mathbb{R}\}\) since \((K - A)\vec{1} = \vec{0}\).
Electronic betweenness

- Write right hand side as $[I^{\text{ext}}]_i = \delta_{is} - \delta_{it}$, where $I^{\text{ext}}$ holds external source and sink currents.
- Matrixingly then:

$$\begin{align*}
(K - A) \tilde{V} &= I^{\text{ext}}.
\end{align*}$$

- $L = K - A$ is a beast of some utility—known as the Laplacian.
- Solve for voltage vector $\tilde{V}$ by LU decomposition (Gaussian elimination).
- Do not compute an inverse!
- Note: voltage offset is arbitrary so no unique solution.
- Presuming network has one component, null space of $K - A$ is one dimensional.
- In fact, $\mathcal{N}(K - A) = \{c\tilde{1}, c \in \mathbb{R}\}$ since $(K - A)\tilde{1} = \tilde{0}$. 
Electronic betweenness

- Write right hand side as $[I^{\text{ext}}]_i = \delta_{is} - \delta_{it}$, where $I^{\text{ext}}$ holds external source and sink currents.
- Matrixingly then:
  
  $$(K - A)\vec{V} = I^{\text{ext}}.$$ 

- $L = K - A$ is a beast of some utility—known as the Laplacian.
- Solve for voltage vector $\vec{V}$ by LU decomposition (Gaussian elimination).
- Do not compute an inverse!
- Note: voltage offset is arbitrary so no unique solution.
- Presuming network has one component, null space of $K - A$ is one dimensional.
- In fact, $\mathcal{N}(K - A) = \{c\vec{1}, c \in \mathbb{R}\}$ since $(K - A)\vec{1} = \vec{0}$. 
Electronic betweenness

- Write right hand side as \([I^{\text{ext}}]_i = \delta_{is} - \delta_{it}\), where \(I^{\text{ext}}\) holds external source and sink currents.
- Matrixingly then:
  \[
  (K - A) \vec{V} = I^{\text{ext}}.
  \]
- \(L = K - A\) is a beast of some utility—known as the Laplacian.
- Solve for voltage vector \(\vec{V}\) by **LU decomposition** (Gaussian elimination).
- Do not compute an inverse!
- **Note**: voltage offset is arbitrary so no unique solution.
- Presuming network has one component, null space of \(K - A\) is one dimensional.
- In fact, \(\mathcal{N}(K - A) = \{c\vec{1}, c \in R\}\) since \((K - A)\vec{1} = \vec{0}\).
Alternate betweenness measures:

Random walk betweenness:

- **Asking too much:** Need full knowledge of network to travel along shortest paths.
- One of many alternatives: consider all random walks between pairs of nodes $i$ and $j$.
- Walks starts at node $i$, traverses the network randomly, ending as soon as it reaches $j$.
- Record the number of times an edge is followed by a walk.
- Consider all pairs of nodes.
- Random walk betweenness of an edge = absolute difference in probability a random walk travels one way versus the other along the edge.
- Equivalent to electronic betweenness.
Alternate betweenness measures:

Random walk betweenness:

- **Asking too much:** Need full knowledge of network to travel along shortest paths.
- One of many alternatives: consider all random walks between pairs of nodes \( i \) and \( j \).
  - Walks starts at node \( i \), traverses the network randomly, ending as soon as it reaches \( j \).
  - Record the number of times an edge is followed by a walk.
  - Consider all pairs of nodes.
  - Random walk betweenness of an edge = absolute difference in probability a random walk travels one way versus the other along the edge.
  - Equivalent to electronic betweenness.
Alternate betweenness measures:

Random walk betweenness:

- **Asking too much**: Need full knowledge of network to travel along shortest paths.
- One of many alternatives: consider all random walks between pairs of nodes $i$ and $j$.
- Walks starts at node $i$, traverses the network randomly, ending as soon as it reaches $j$.
- Record the number of times an edge is followed by a walk.
- Consider all pairs of nodes.
- Random walk betweenness of an edge = absolute difference in probability a random walk travels one way versus the other along the edge.
- Equivalent to electronic betweenness.
Alternate betweenness measures:

Random walk betweenness:

- Asking too much: Need full knowledge of network to travel along shortest paths.
- One of many alternatives: consider all random walks between pairs of nodes $i$ and $j$.
- Walks starts at node $i$, traverses the network randomly, ending as soon as it reaches $j$.
- Record the number of times an edge is followed by a walk.
- Consider all pairs of nodes.
- Random walk betweenness of an edge = absolute difference in probability a random walk travels one way versus the other along the edge.
- Equivalent to electronic betweenness.
Alternate betweenness measures:

Random walk betweenness:

- **Asking too much:** Need full knowledge of network to travel along shortest paths.

- One of many alternatives: consider all random walks between pairs of nodes $i$ and $j$.

- Walks starts at node $i$, traverses the network randomly, ending as soon as it reaches $j$.

- Record the number of times an edge is followed by a walk.

- Consider all pairs of nodes.

  - Random walk betweenness of an edge = absolute difference in probability a random walk travels one way versus the other along the edge.

  - Equivalent to electronic betweenness.
Alternate betweenness measures:

Random walk betweenness:

- **Asking too much**: Need full knowledge of network to travel along shortest paths.
- One of many alternatives: consider all random walks between pairs of nodes \( i \) and \( j \).
- Walks starts at node \( i \), traverses the network randomly, ending as soon as it reaches \( j \).
- Record the number of times an edge is followed by a walk.
- Consider all pairs of nodes.
- Random walk betweenness of an edge = absolute difference in probability a random walk travels one way versus the other along the edge.
- Equivalent to electronic betweenness.
Alternate betweenness measures:

Random walk betweenness:

- Asking too much: Need full knowledge of network to travel along shortest paths.
- One of many alternatives: consider all random walks between pairs of nodes $i$ and $j$.
- Walks starts at node $i$, traverses the network randomly, ending as soon as it reaches $j$.
- Record the number of times an edge is followed by a walk.
- Consider all pairs of nodes.
- Random walk betweenness of an edge = absolute difference in probability a random walk travels one way versus the other along the edge.
- Equivalent to electronic betweenness.
Hierarchy by division

- Third column shows what happens if we don’t recompute betweenness after each edge removal.
Scientists working on networks

FIG. 10. Illustration of the use of the community-structure algorithm to make sense of a complex network. The initial network is a network of coauthorships between physicists who have published on topics related to networks. The figure shows only the largest component of the network, which contains 145 scientists. There are 90 more scientists in smaller components, which are not shown.

Application of the shortest-path betweenness version of the community-structure algorithm produces the communities indicated by the shades of the vertices.

A coarse-graining of the network in which each community is represented by a single node, with edges representing collaborations between communities. The thickness of the edges is proportional to the number of pairs of collaborators between communities. Clearly panel (c) reveals much that is not easily seen in the original network of panel (a).
Scientists working on networks

Overview

Methods
- Hierarchy by aggregation
- Hierarchy by division
- Hierarchy by shuffling
- Spectral methods
- Hierarchies & Missing Links
- General structure detection

References
Scientists working on networks

FIG. 10. Illustration of the use of the community-structure algorithm to make sense of a complex network.

The initial network is a network of coauthorships between physicists who have published on topics related to networks. The figure shows only the largest component of the network, which contains 145 scientists. There are 90 more scientists in smaller components, which are not shown.

Application of the shortest-path betweenness version of the community-structure algorithm produces the communities indicated by the shades of the vertices.

A coarse-graining of the network in which each community is represented by a single node, with edges representing collaborations between communities. The thickness of the edges is proportional to the number of pairs of collaborators between communities. Clearly panel (c) reveals much that is not easily seen in the original network of panel (a).
The split into two groups appears to correspond to a known division of the dolphin community. Lusseau reports that for a period of about two years during observation of the dolphins they separated into two groups along the lines found by our analysis, apparently because of the disappearance of individuals on the boundary between the groups. When some of these individuals later reappeared, the two halves of the network joined together once more. As Lusseau points out, developments of this kind illustrate that the dolphin network is not merely a scientific curiosity but, like human social networks, is closely tied to the evolution of the community. The subgroupings within the larger half of the network also seem to correspond to real divisions among the animals: the largest subgroup consists almost entirely of females and the others almost entirely of males, and it is conjectured that the split between the male groups is governed by matrilineage.

Figure 12 shows the community structure of the network of interactions between major characters in Victor Hugo’s sprawling novel of crime and redemption in post-restoration France. The greatest modularity achieved in the shortest-path version of our algorithm is $Q = 0.54$ and corresponds to the 11 communities shown.
Les Miserables

Frame 24/54
Outline

Overview

Methods
- Hierarchy by aggregation
- Hierarchy by division
- Hierarchy by shuffling
- Spectral methods
- Hierarchies & Missing Links
- General structure detection

References
Shuffling for structure

“Extracting the hierarchical organization of complex systems”
Sales-Pardo et al., PNAS (2007) [8, 9]

Consider all partitions of networks into $m$ groups
As for Newman and Girvan approach, aim is to find partitions with maximum modularity:

$$Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] = \text{Tr}E - \|E^2\|_1.$$
Shuffling for structure

- “Extracting the hierarchical organization of complex systems”
  Sales-Pardo et al., PNAS (2007) [8, 9]
- Consider all partitions of networks into $m$ groups
- As for Newman and Girvan approach, aim is to find partitions with maximum modularity:

$$Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] = \text{Tr}E - \|E^2\|_1.$$
Shuffling for structure

- “Extracting the hierarchical organization of complex systems”
  Sales-Pardo et al., PNAS (2007) [8, 9]

- Consider all partitions of networks into $m$ groups

- As for Newman and Girvan approach, aim is to find partitions with maximum modularity:

$$Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] = \text{Tr}E - ||E^2||_1.$$
Shuffling for structure

- Consider **partition network**, i.e., the network of all possible partitions.
  - **Defn:** Two partitions are connected if they differ only by the reassignment of a single node.
  - Look for local maxima in partition network.
  - Construct an **affinity matrix** with entries $A_{ij}$.
  - $A_{ij} = \Pr$ random walker on modularity network ends up at a partition with $i$ and $j$ in the same group.
  - C.f. **topological overlap** between $i$ and $j = \#$ matching neighbors for $i$ and $j$ divided by maximum of $k_i$ and $k_j$. 
Shuffling for structure

- Consider **partition network**, i.e., the network of all possible partitions.
- **Defn:** Two partitions are connected if they differ only by the reassignment of a single node.
- Look for local maxima in partition network.
- Construct an **affinity matrix** with entries $A_{ij}$.
  \[ A_{ij} = \Pr \text{ random walker on modularity network ends up at a partition with } i \text{ and } j \text{ in the same group.} \]
- C.f. **topological overlap** between $i$ and $j = \# \text{ matching neighbors for } i \text{ and } j \text{ divided by maximum of } k_i \text{ and } k_j.$
Shuffling for structure

- Consider partition network, i.e., the network of all possible partitions.
- **Defn:** Two partitions are connected if they differ only by the reassignment of a single node.
- Look for local maxima in partition network.
- Construct an affinity matrix with entries $A_{ij}$.
- $A_{ij} = \Pr$ random walker on modularity network ends up at a partition with $i$ and $j$ in the same group.
- C.f. topological overlap between $i$ and $j = \#$ matching neighbors for $i$ and $j$ divided by maximum of $k_i$ and $k_j$. 
Shuffling for structure

- Consider **partition network**, i.e., the network of all possible partitions.
- **Defn:** Two partitions are connected if they differ only by the reassignment of a single node.
- Look for local maxima in partition network.
- Construct an **affinity matrix** with entries $A_{ij}$.
  - $A_{ij} = \Pr$ random walker on modularity network ends up at a partition with $i$ and $j$ in the same group.
  - C.f. **topological overlap** between $i$ and $j = \#$ matching neighbors for $i$ and $j$ divided by maximum of $k_i$ and $k_j$. 
Shuffling for structure

- Consider **partition network**, i.e., the network of all possible partitions.
- **Defn:** Two partitions are connected if they differ only by the reassignment of a single node.
- Look for local maxima in partition network.
- Construct an **affinity matrix** with entries $A_{ij}$.
  \[ A_{ij} = \Pr \text{ random walker on modularity network ends up at a partition with } i \text{ and } j \text{ in the same group.} \]
- C.f. **topological overlap** between $i$ and $j = \# \text{ matching neighbors for } i \text{ and } j \text{ divided by maximum of } k_i \text{ and } k_j.$
Shuffling for structure

A: Base network; B: Partition network; C: Coclassification matrix; D: Comparison to random networks (all the same!); E: Ordered coclassification matrix;
**Shuffling for structure**

- **A:** Base network; **B:** Partition network; **C:** Coclassification matrix; **D:** Comparison to random networks (all the same!); **E:** Ordered coclassification matrix; Conclusion: no structure...
Shuffling for structure

- Method obtains a distribution of classification hierarchies.
- Note: the hierarchy with the highest modularity score isn’t chosen.
- Idea is to weight possible hierarchies according to their basin of attraction’s size in the partition network.
- Next step: Given affinities, now need to sort nodes into modules, submodules, and so on.

Idea: permute nodes to minimize following cost

\[ C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} |i - j|. \]

- Use simulated annealing (slow).
- Observation: should achieve same results for more general cost function: 
  \[ C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} f(|i - j|) \]
  where \( f \) is a strictly monotonically increasing function of 0, 1, 2, ...
Shuffling for structure

- Method obtains a distribution of classification hierarchies.
- Note: the hierarchy with the highest modularity score isn’t chosen.
- Idea is to weight possible hierarchies according to their basin of attraction’s size in the partition network.
- Next step: Given affinities, now need to sort nodes into modules, submodules, and so on.
- Idea: permute nodes to minimize following cost

\[
C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} |i - j|.
\]

- Use simulated annealing (slow).
- Observation: should achieve same results for more general cost function: 

\[
C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} f(|i - j|)
\]

where \( f \) is a strictly monotonically increasing function of 0, 1, 2, ...
Shuffling for structure

- Method obtains a distribution of classification hierarchies.
- Note: the hierarchy with the highest modularity score isn’t chosen.
- Idea is to weight possible hierarchies according to their basin of attraction’s size in the partition network.
- Next step: Given affinities, now need to sort nodes into modules, submodules, and so on.
- Idea: permute nodes to minimize following cost

\[ C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} |i - j|. \]

- Use simulated annealing (slow).
- Observation: should achieve same results for more general cost function: 
  \[ C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} f(|i - j|) \]
  where \( f \) is a strictly monotonically increasing function of 0, 1, 2, ...
Shuffling for structure

- Method obtains a distribution of classification hierarchies.
- Note: the hierarchy with the highest modularity score isn’t chosen.
- Idea is to weight possible hierarchies according to their basin of attraction’s size in the partition network.
- **Next step:** Given affinities, now need to sort nodes into modules, submodules, and so on.
- Idea: permute nodes to minimize following cost

\[
C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} |i - j|.
\]

- Use simulated annealing (slow).
- **Observation:** should achieve same results for more general cost function:

\[
C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} f(|i - j|)
\]

where \( f \) is a strictly monotonically increasing function of 0, 1, 2, ...
Shuffling for structure

- Method obtains a distribution of classification hierarchies.
- Note: the hierarchy with the highest modularity score isn’t chosen.
- Idea is to weight possible hierarchies according to their basin of attraction’s size in the partition network.
- Next step: Given affinities, now need to sort nodes into modules, submodules, and so on.
- Idea: permute nodes to minimize following cost

\[ C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} |i - j|. \]

- Use simulated annealing (slow).
- Observation: should achieve same results for more general cost function: \( C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} f(|i - j|) \) where \( f \) is a strictly monotonically increasing function of 0, 1, 2, ...
Shuffling for structure

- Method obtains a distribution of classification hierarchies.
- Note: the hierarchy with the highest modularity score isn’t chosen.
- Idea is to weight possible hierarchies according to their basin of attraction’s size in the partition network.
- Next step: Given affinities, now need to sort nodes into modules, submodules, and so on.
- Idea: permute nodes to minimize following cost

\[ C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} |i - j|. \]

- Use simulated annealing (slow).
- Observation: should achieve same results for more general cost function: \( C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} f(|i - j|) \) where \( f \) is a strictly monotonically increasing function of 0, 1, 2, ...
Shuffling for structure

- Method obtains a distribution of classification hierarchies.
- Note: the hierarchy with the highest modularity score isn’t chosen.
- Idea is to weight possible hierarchies according to their basin of attraction’s size in the partition network.
- Next step: Given affinities, now need to sort nodes into modules, submodules, and so on.
- Idea: permute nodes to minimize following cost

\[ C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} |i - j|. \]

- Use simulated annealing (slow).
- Observation: should achieve same results for more general cost function: \( C = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} f(|i - j|) \) where \( f \) is a strictly monotonically increasing function of 0, 1, 2, ...
Shuffling for structure

- $N = 640$,
- $\langle k \rangle = 16$,
- 3 tiered hierarchy.
Shuffling for structure

- Define cost matrix as $T$ with entries $T_{ij} = f(|i - j|)$.
- Weird observation: if $T_{ij} = (i - j)^2$ then $T$ is of rank 3, independent of $N$.
- Discovered by numerical inspection...
- The eigenvalues are

$$
\lambda_1 = -\frac{1}{6}n(n^2 - 1),
$$

$$
\lambda_2 = +\sqrt{nS_{n,4} + S_{n,2}}, \text{ and}
$$

$$
\lambda_3 = -\sqrt{nS_{n,4} + S_{n,2}}.
$$

where

$$
S_{n,2} = \frac{1}{12}n(n^2 - 1), \text{ and}
$$

$$
S_{n,4} = \frac{1}{240}n(n^2 - 1)(3n^2 - 7).
$$
Shuffling for structure

- Define cost matrix as \( T \) with entries \( T_{ij} = f(|i - j|) \).
- Weird observation: if \( T_{ij} = (i - j)^2 \) then \( T \) is of rank 3, independent of \( N \).
- Discovered by numerical inspection...
- The eigenvalues are

\[
\lambda_1 = -\frac{1}{6}n(n^2 - 1),
\]

\[
\lambda_2 = +\sqrt{nS_{n,4} + S_{n,2}}, \text{ and}
\]

\[
\lambda_3 = -\sqrt{nS_{n,4} + S_{n,2}}.
\]

where

\[
S_{n,2} = \frac{1}{12}n(n^2 - 1), \text{ and}
\]

\[
S_{n,4} = \frac{1}{240}n(n^2 - 1)(3n^2 - 7).
\]
Shuffling for structure

- Define **cost matrix** as $T$ with entries $T_{ij} = f(|i - j|)$.
- Weird observation: if $T_{ij} = (i - j)^2$ then $T$ is of **rank 3**, independent of $N$.
- Discovered by numerical inspection...

The eigenvalues are

$$
\lambda_1 = -\frac{1}{6} n(n^2 - 1), \\
\lambda_2 = +\sqrt{nS_{n,4} + S_{n,2}}, \text{ and} \\
\lambda_3 = -\sqrt{nS_{n,4} + S_{n,2}}.
$$

where

$$
S_{n,2} = \frac{1}{12} n(n^2 - 1), \text{ and} \\
S_{n,4} = \frac{1}{240} n(n^2 - 1)(3n^2 - 7).
$$
Shuffling for structure

- Define cost matrix as $T$ with entries $T_{ij} = f(|i - j|)$.
- Weird observation: if $T_{ij} = (i - j)^2$ then $T$ is of rank 3, independent of $N$.
- Discovered by numerical inspection...
- The eigenvalues are

$$
\lambda_1 = -\frac{1}{6}n(n^2 - 1),
$$
$$
\lambda_2 = +\sqrt{nS_{n,4} + S_{n,2}}, \text{ and}
$$
$$
\lambda_3 = -\sqrt{nS_{n,4} + S_{n,2}}.
$$

where

$$
S_{n,2} = \frac{1}{12}n(n^2 - 1), \text{ and}
$$
$$
S_{n,4} = \frac{1}{240}n(n^2 - 1)(3n^2 - 7).
$$
Shuffling for structure

- Eigenvectors

\[
\begin{align*}
(\vec{v}_1)_i &= \left(i - \frac{n+1}{2}\right), \\
(\vec{v}_2)_i &= \left(i - \frac{n+1}{2}\right)^2 + \sqrt{S_{n,4}/n}, \text{ and} \\
(\vec{v}_3)_i &= \left(i - \frac{n+1}{2}\right)^2 - \sqrt{S_{n,4}/n}.
\end{align*}
\]

- Remarkably,

\[
T = \lambda_1 \hat{v}_1 \hat{v}_1^T + \lambda_2 \hat{v}_2 \hat{v}_2^T + \lambda_3 \hat{v}_3 \hat{v}_3^T.
\]

- The next step: figure out how to capitalize on this...
Shuffling for structure

- **Eigenvectors**

\[
(\hat{v}_1)_i = \left( i - \frac{n + 1}{2} \right),
\]
\[
(\hat{v}_2)_i = \left( i - \frac{n + 1}{2} \right)^2 + \sqrt{\frac{S_{n,4}}{n}}, \text{ and}
\]
\[
(\hat{v}_3)_i = \left( i - \frac{n + 1}{2} \right)^2 - \sqrt{\frac{S_{n,4}}{n}}.
\]

- **Remarkably,**

\[
T = \lambda_1 \hat{v}_1 \hat{v}_1^T + \lambda_2 \hat{v}_2 \hat{v}_2^T + \lambda_3 \hat{v}_3 \hat{v}_3^T.
\]

- **The next step:** figure out how to capitalize on this...
Shuffling for structure

- **Eigenvectors**

\[
(\hat{v}_1)_i = \left( i - \frac{n+1}{2} \right),
\]

\[
(\hat{v}_2)_i = \left( i - \frac{n+1}{2} \right)^2 + \sqrt{S_{n,4}/n}, \text{ and}
\]

\[
(\hat{v}_3)_i = \left( i - \frac{n+1}{2} \right)^2 - \sqrt{S_{n,4}/n}.
\]

- **Remarkably,**

\[
T = \lambda_1 \hat{v}_1 \hat{v}_1^T + \lambda_2 \hat{v}_2 \hat{v}_2^T + \lambda_3 \hat{v}_3 \hat{v}_3^T.
\]

- **The next step:** figure out how to capitalize on this...
Shuffling for structure

Table 1. Top-level structure of real-world networks

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Edges</th>
<th>Modules</th>
<th>Main modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air transportation</td>
<td>3,618</td>
<td>28,284</td>
<td>57</td>
<td>8</td>
</tr>
<tr>
<td>E-mail</td>
<td>1,133</td>
<td>10,902</td>
<td>41</td>
<td>8</td>
</tr>
<tr>
<td>Electronic circuit</td>
<td>516</td>
<td>686</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td><em>Escherichia coli</em> KEGG</td>
<td>739</td>
<td>1,369</td>
<td>39</td>
<td>13</td>
</tr>
<tr>
<td><em>E. coli</em> UCSD</td>
<td>507</td>
<td>947</td>
<td>28</td>
<td>17</td>
</tr>
</tbody>
</table>
Shuffling for structure

- Modules found match up with geopolitical units.
Shuffling for structure

For the second level, we find that for most of the modules, all of the pathways associated with a single top level module. Again, the width of each submodule is proportional to the number of pathways in the module. For each module, we first identify the pathway classifications of the pathways in the module (see SI Fig. 11). The overall organization of the network is similar and independent of the reconstruction used to build the network (43). The steps of the method would remain unchanged; one of the key differences is that in the real-world network, the J. S. McDonnell Foundation, and a National Institutes of Health Roadmap Initiative on Systems Biology. L.A.N. gratefully acknowledges the support of the Keck Foundation, the National Science Foundation, the National Institutes of Health, the National Institute of General Medical Sciences, the Environmental Protection Agency, the National Institute of Standards and Technology, the National Institute of Forestry and Wildlife, and the National Institute of Energy and Environment. M.S.-P. and R.G. thank the National Science Foundation for support.

Our analysis of model and real-world networks demonstrates that our algorithm can be easily generalized to other classes of graphs (e.g., citation networks, food webs, or gene-regulatory networks). The steps of the method would remain unchanged; one of the key differences is that in the real-world network, the J. S. McDonnell Foundation, and a National Institutes of Health Roadmap Initiative on Systems Biology. L.A.N. gratefully acknowledges the support of the Keck Foundation, the National Science Foundation, the National Institutes of Health, the National Institute of General Medical Sciences, the Environmental Protection Agency, the National Institute of Standards and Technology, the National Institute of Forestry and Wildlife, and the National Institute of Energy and Environment. M.S.-P. and R.G. thank the National Science Foundation for support.

Our analysis of model and real-world networks demonstrates that our algorithm can be easily generalized to other classes of graphs (e.g., citation networks, food webs, or gene-regulatory networks). The steps of the method would remain unchanged; one of the key differences is that in the real-world network, the J. S. McDonnell Foundation, and a National Institutes of Health Roadmap Initiative on Systems Biology. L.A.N. gratefully acknowledges the support of the Keck Foundation, the National Science Foundation, the National Institutes of Health, the National Institute of General Medical Sciences, the Environmental Protection Agency, the National Institute of Standards and Technology, the National Institute of Forestry and Wildlife, and the National Institute of Energy and Environment. M.S.-P. and R.G. thank the National Science Foundation for support.

Modularity structure for metabolic network of E. coli (UCSD reconstruction).
Outline

Overview

Methods

- Hierarchy by aggregation
- Hierarchy by division
- Hierarchy by shuffling

Spectral methods

Hierarchies & Missing Links

General structure detection

References
General structure detection

- “Detecting communities in large networks” Capocci et al. (2005) [1]
  - Consider normal matrix $K^{-1}A$, random walk matrix $A^TK^{-1}$, Laplacian $K - A$, and $AA^T$.
  - Basic observation is that eigenvectors associated with secondary eigenvalues reveal evidence of structure.
  - Build on Kleinberg’s HITS algorithm.
General structure detection

▶ “Detecting communities in large networks” Capocci et al. (2005) [1]

▶ Consider normal matrix $K^{-1}A$, random walk matrix $A^T K^{-1}$, Laplacian $K - A$, and $AA^T$.

▶ Basic observation is that eigenvectors associated with secondary eigenvalues reveal evidence of structure.

▶ Build on Kleinberg’s HITS algorithm.
General structure detection

- “Detecting communities in large networks” Capocci et al. (2005) [1]
- Consider normal matrix $K^{-1}A$, random walk matrix $A^TK^{-1}$, Laplacian $K - A$, and $AA^T$.
- Basic observation is that eigenvectors associated with secondary eigenvalues reveal evidence of structure.
- Build on Kleinberg’s HITS algorithm.
General structure detection

- “Detecting communities in large networks” Capocci et al. (2005) [1]
- Consider normal matrix $K^{-1}A$, random walk matrix $A^TK^{-1}$, Laplacian $K - A$, and $AA^T$.
- Basic observation is that eigenvectors associated with secondary eigenvalues reveal evidence of structure.
- Build on Kleinberg’s HITS algorithm.
General structure detection

- Example network:

![Example network graph](image)
When dealing with a directed network, links do not correspond to any equivalence relation. Rather, pointing to common neighbors is a significant relation, as suggested in the sociologists' literature where this quantity measures the so-called structural equivalence of nodes [18]. Accordingly, in a directed network, clusters should be composed by nodes pointing to a high number of common neighbors, no matter their direct linkage. For directed networks, we thus modify our method in the streamline of the HITS algorithm [17]. The HITS algorithm was proposed on empirical bases to find the main communities in large oriented networks. It assumes that the largest components (in the absolute value) of eigenvectors of the matrices $AA^T$ and $A^T A$ correspond to highly clustered nodes belonging to a single community. Such algorithm efficiently detects the main communities, even when these are not sharply defined. However, it becomes computationally heavy when one is interested in minor communities, which correspond to smaller eigenvalues. As explained in the undirected case, we tackle this issue by combining information from the first few eigenvectors of the normal matrix and extracting the community structure from correlations between the same components in different eigenvectors.

To detect the community structure in a directed network, we therefore replace, in the previous analysis, the matrix $W$ with a matrix $Y = WW^T$. This corresponds to replacing the directed network with an undirected weighted network, where nodes pointing to common neighbors are connected by a link, whose intensity is proportional to the total sum of the weights of the links pointing from the two original nodes to the common neighbors. Then, one performs the analysis on the undirected network as described previously. Thus, the function to minimize in this case is

$$y(x) = \sum_{ij} x_i/C0 x_j^2 w_{il} w_{jl}.$$ (5)
General structure detection

- Network of word associations for 10616 words.
- Average in-degree of 7.
- Using 2nd to 11th eigenvectors of a modified version of $AA^T$.

Table 1
Words most correlated to science, literature and piano in the eigenvectors of $Q^{-1}WW^T$

<table>
<thead>
<tr>
<th>Science</th>
<th>1</th>
<th>Literature</th>
<th>1</th>
<th>Piano</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scientific</td>
<td>0.994</td>
<td>Dictionary</td>
<td>0.994</td>
<td>Cello</td>
<td>0.993</td>
</tr>
<tr>
<td>Chemistry</td>
<td>0.990</td>
<td>Editorial</td>
<td>0.990</td>
<td>Fiddle</td>
<td>0.992</td>
</tr>
<tr>
<td>Physics</td>
<td>0.988</td>
<td>Synopsis</td>
<td>0.988</td>
<td>Viola</td>
<td>0.990</td>
</tr>
<tr>
<td>Concentrate</td>
<td>0.973</td>
<td>Words</td>
<td>0.987</td>
<td>Banjo</td>
<td>0.988</td>
</tr>
<tr>
<td>Thinking</td>
<td>0.973</td>
<td>Grammar</td>
<td>0.986</td>
<td>Saxophone</td>
<td>0.985</td>
</tr>
<tr>
<td>Test</td>
<td>0.973</td>
<td>Adjective</td>
<td>0.983</td>
<td>Director</td>
<td>0.984</td>
</tr>
<tr>
<td>Lab</td>
<td>0.969</td>
<td>Chapter</td>
<td>0.982</td>
<td>Violin</td>
<td>0.983</td>
</tr>
<tr>
<td>Brain</td>
<td>0.965</td>
<td>Prose</td>
<td>0.979</td>
<td>Clarinet</td>
<td>0.983</td>
</tr>
<tr>
<td>Equation</td>
<td>0.963</td>
<td>Topic</td>
<td>0.976</td>
<td>Oboe</td>
<td>0.983</td>
</tr>
<tr>
<td>Examine</td>
<td>0.962</td>
<td>English</td>
<td>0.975</td>
<td>Theater</td>
<td>0.982</td>
</tr>
</tbody>
</table>

Values indicate the correlation.
Outline

Overview

Methods
Hierarchy by aggregation
Hierarchy by division
Hierarchy by shuffling
Spectral methods

Hierarchies & Missing Links

General structure detection

References
Hierarchies and missing links

- Idea: Shades indicate probability that nodes in left and right subtrees of dendogram are connected.
- Handle: Hierarchical random graph models.
- Plan: Infer consensus dendogram for a given real network.
- Obtain probability that links are missing (big problem...).
Hierarchies and missing links

- Idea: Shades indicate probability that nodes in left and right subtrees of dendogram are connected.
- Handle: Hierarchical random graph models.
- Plan: Infer consensus dendogram for a given real network.
- Obtain probability that links are missing (big problem...).
Hierarchies and missing links

- Idea: Shades indicate probability that nodes in left and right subtrees of dendogram are connected.
- Handle: Hierarchical random graph models.
- Plan: Infer consensus dendogram for a given real network.
- Obtain probability that links are missing (big problem...).
Hierarchies and missing links

- Idea: Shades indicate probability that nodes in left and right subtrees of dendrogram are connected.
- Handle: Hierarchical random graph models.
- Plan: Infer consensus dendogram for a given real network.
- Obtain probability that links are missing (big problem...).
Hierarchies and missing links

- Model also predicts reasonably well
  1. average degree,
  2. clustering,
  3. and average shortest path length.

Table 1 | Comparison of original and resampled networks

<table>
<thead>
<tr>
<th>Network</th>
<th>$\langle k \rangle_{\text{real}}$</th>
<th>$\langle k \rangle_{\text{samp}}$</th>
<th>$C_{\text{real}}$</th>
<th>$C_{\text{samp}}$</th>
<th>$d_{\text{real}}$</th>
<th>$d_{\text{samp}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T. pallidum</td>
<td>4.8</td>
<td>3.7(1)</td>
<td>0.0625</td>
<td>0.0444(2)</td>
<td>3.690</td>
<td>3.940(6)</td>
</tr>
<tr>
<td>Terrorists</td>
<td>4.9</td>
<td>5.1(2)</td>
<td>0.361</td>
<td>0.352(1)</td>
<td>2.575</td>
<td>2.794(7)</td>
</tr>
<tr>
<td>Grassland</td>
<td>3.0</td>
<td>2.9(1)</td>
<td>0.174</td>
<td>0.168(1)</td>
<td>3.29</td>
<td>3.69(2)</td>
</tr>
</tbody>
</table>

Statistics are shown for the three example networks studied and for new networks generated by resampling from our hierarchical model. The generated networks closely match the average degree $\langle k \rangle$, clustering coefficient $C$ and average vertex–vertex distance $d$ in each case, suggesting that they capture much of the structure of the real networks. Parenthetical values indicate standard errors on the final digits.
Hierarchies and missing links

- Consensus dendogram for grassland species.

- Copes with disassortative and assortative communities.
Hierarchies and missing links

- Consensus dendogram for grassland species.
- Copes with disassortative and assortative communities.
Structure detection methods

Overview

Methods
- Hierarchy by aggregation
- Hierarchy by division
- Hierarchy by shuffling
- Spectral methods
- Hierarchies & Missing Links

General structure detection

References
“The discovery of structural form”
General structure detection

A | Structural Form | Generative process |
---|----------------|-------------------|
Partition | | |
Chain | | |
Order | | |
Ring | | |
Hierarchy | | |
Tree | | |
Grid | Chain II Chain |
Cylinder | Chain II Ring |

- Top down description of form.
- Node replacement graph grammar: parent node becomes two child nodes.
- B-D: Growing chains, orders, and trees.
General structure detection

- **Top down description of form.**
- **Node replacement graph grammar:** parent node becomes two child nodes.
- **B-D:** Growing chains, orders, and trees.

A | Structural Form | Generative process | B | C | D | E |
--- | --- | --- | --- | --- | --- | --- |
Partition | ![Image of partition structure](image) | ![Image of generative process for partition](image) | ![Image of generative process for partition](image) | ![Image of generative process for partition](image) | ![Image of generative process for partition](image) | ![Image of generative process for partition](image) |
Chain | ![Image of chain structure](image) | ![Image of generative process for chain](image) | ![Image of generative process for chain](image) | ![Image of generative process for chain](image) | ![Image of generative process for chain](image) | ![Image of generative process for chain](image) |
Order | ![Image of order structure](image) | ![Image of generative process for order](image) | ![Image of generative process for order](image) | ![Image of generative process for order](image) | ![Image of generative process for order](image) | ![Image of generative process for order](image) |
Ring | ![Image of ring structure](image) | ![Image of generative process for ring](image) | ![Image of generative process for ring](image) | ![Image of generative process for ring](image) | ![Image of generative process for ring](image) | ![Image of generative process for ring](image) |
Hierarchy | ![Image of hierarchy structure](image) | ![Image of generative process for hierarchy](image) | ![Image of generative process for hierarchy](image) | ![Image of generative process for hierarchy](image) | ![Image of generative process for hierarchy](image) | ![Image of generative process for hierarchy](image) |
Tree | ![Image of tree structure](image) | ![Image of generative process for tree](image) | ![Image of generative process for tree](image) | ![Image of generative process for tree](image) | ![Image of generative process for tree](image) | ![Image of generative process for tree](image) |
Grid | ![Image of grid structure](image) | ![Image of generative process for grid](image) | ![Image of generative process for grid](image) | ![Image of generative process for grid](image) | ![Image of generative process for grid](image) | ![Image of generative process for grid](image) |
Cylinder | ![Image of cylinder structure](image) | ![Image of generative process for cylinder](image) | ![Image of generative process for cylinder](image) | ![Image of generative process for cylinder](image) | ![Image of generative process for cylinder](image) | ![Image of generative process for cylinder](image) |
General structure detection

- Top down description of form.
- Node replacement graph grammar: parent node becomes two child nodes.
- B-D: Growing chains, orders, and trees.

![Diagram of structural forms and generative processes]

- Structural Form:
  - Partition
  - Chain
  - Order
  - Ring
  - Hierarchy
  - Tree
  - Grid
  - Cylinder

- Generative process:
  - Chain → Tree
  - Chain → Ring
  - Ring → Chain II Ring
  - Partition → Chain
  - Hierarchy → Tree

References
Example learned structures:

- Biological features; Supreme Court votes; perceived color differences; face differences; & distances between cities.
General structure detection

Effect of adding features on detected form.

Straight partition

simple tree

complex tree
General structure detection

Effect of adding features on detected form.

- Straight partition
  - simple tree
  - complex tree
General structure detection

Performance for test networks.

True Partition Chain Ring Tree Grid

- Frame 50/54
References I

Detecting communities in large networks. 

Hierarchical structure and the prediction of missing links in networks. 

Community structure in social and biological networks. 
References II


References III


An information flow model for conflict and fission in small groups. 