Structure detection methods

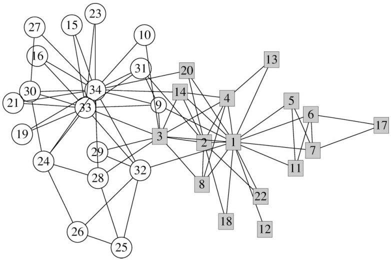
Complex Networks, CSYS/MATH 303, Spring, 2010

Prof. Peter Dodds

Department of Mathematics & Statistics
Center for Complex Systems
Vermont Advanced Computing Center
University of Vermont

Outline

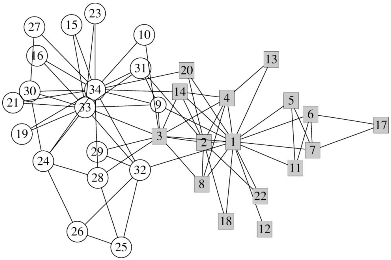
Structure detection



▲ Zachary's karate club [10, 7]

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

Structure detection

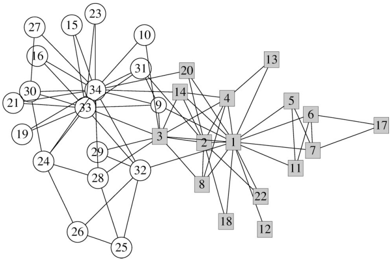


▲ Zachary's karate club [10, 7]

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

- Possible substructures:
  hierarchies, cliques, rings, . . .
- Plus:
  All combinations of substructures.
- Much focus on hierarchies...

Structure detection

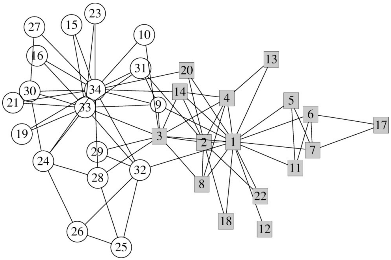


▲ Zachary's karate club [10, 7]

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

- Possible substructures:
  hierarchies, cliques, rings, . . .
- Plus:
  All combinations of substructures.
- Much focus on hierarchies...

Structure detection



▲ Zachary's karate club [10, 7]

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

- Possible substructures:
  hierarchies, cliques, rings, . . .
- Plus:
  All combinations of substructures.
- Much focus on hierarchies...

Outline

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

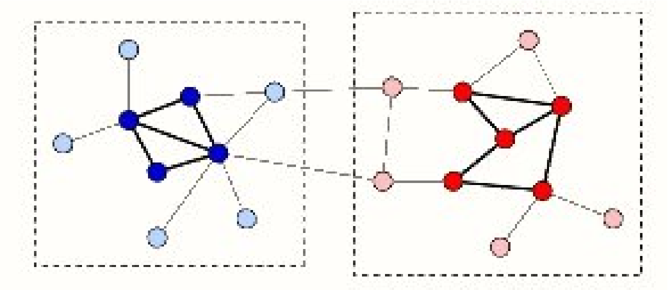
Bottom up problems:

- 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

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.
- 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.
- 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:

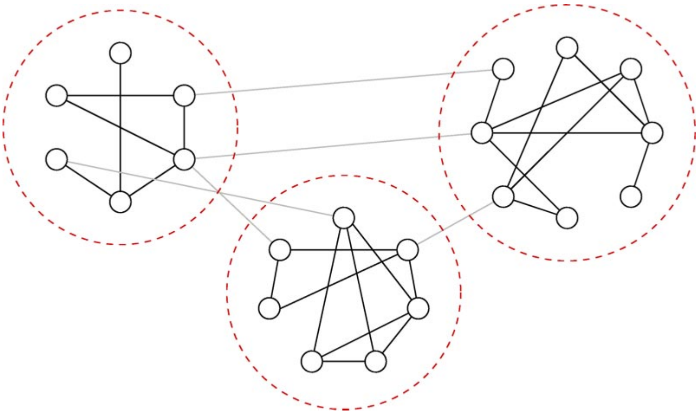
- Idea: Identify global structure first and recursively uncover more detailed structure.
- Basic objective: find dominant components that have significantly more links within than without, as compared to randomized version.
- We'll first work through "Finding and evaluating community structure in networks" 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.
- 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



- *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.

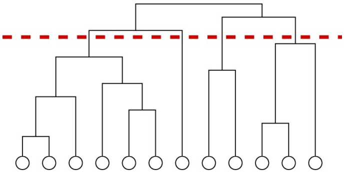
Hierarchy by division

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.

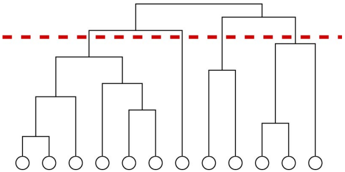
Hierarchy by division

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:

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

  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:

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

where e_{ij} is the fraction of edges between 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:

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

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:

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

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:

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

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:

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

  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:

$$\langle k \rangle_{\text{in}} = 6 \text{ and } \langle k \rangle_{\text{out}} = 2.$$

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:

$$\langle k \rangle_{\text{in}} = 6 \text{ and } \langle k \rangle_{\text{out}} = 2.$$

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:

$$\langle k \rangle_{\text{in}} = 6 \text{ and } \langle k \rangle_{\text{out}} = 2.$$

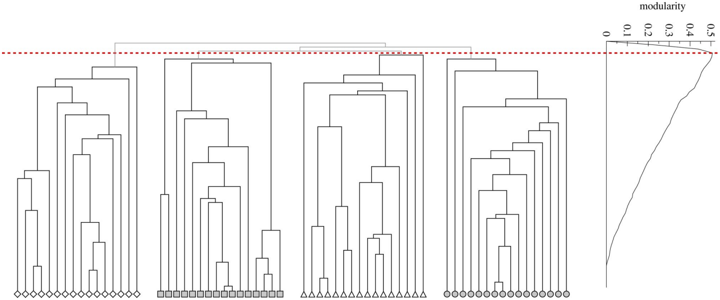
Hierarchy by division

Test case:

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

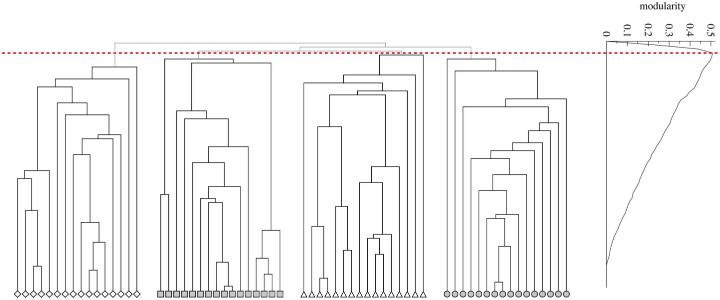
$$\langle k \rangle_{\rm in} = 6 \text{ and } \langle k \rangle_{\rm out} = 2.$$

Hierarchy by division



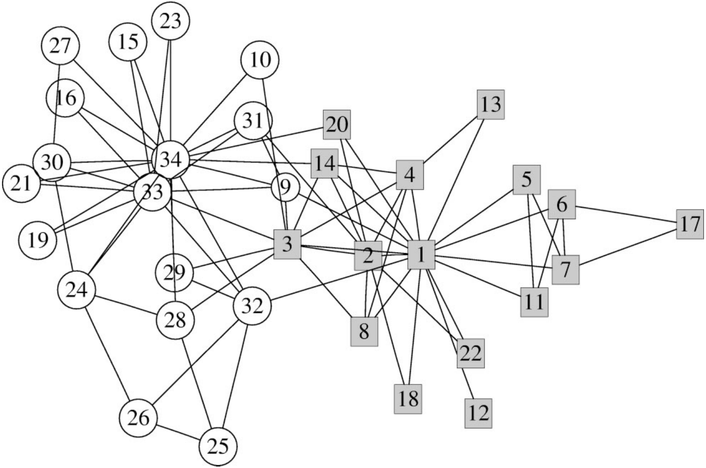
- Maximum modularity $Q \simeq 0.5$ obtained when four communities are uncovered.
- Further 'discovery' of internal structure is somewhat meaningless, as any communities arise accidentally.

Hierarchy by division



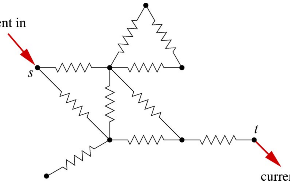
- Maximum modularity $Q \simeq 0.5$ obtained when four communities are uncovered.
- Further 'discovery' of internal structure is somewhat meaningless, as any communities arise accidentally.

Hierarchy by division



- Factions in Zachary's karate club network. [10]

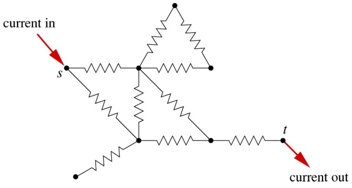
Betweenness for electrons:



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

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

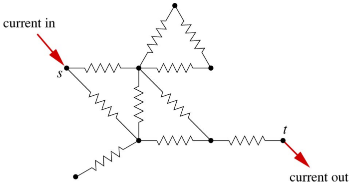
Betweenness for electrons:



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

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

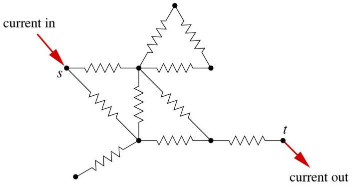
Betweenness for electrons:



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

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

Betweenness for electrons:

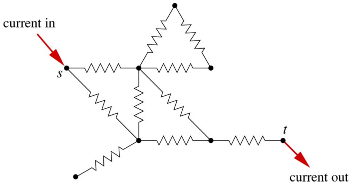


- Unit resistors on each edge.
- For every pair of nodes s (source) and t (sink), set up unit currents in at s and out at t.
- Measure absolute current along each edge ℓ, $|I_{\ell,st}|$.

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

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

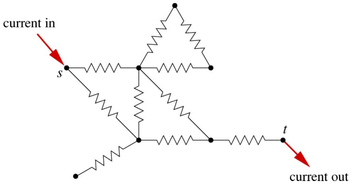
Betweenness for electrons:



- 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 ℓ, $|I_{\ell,st}|$.
- Sum $|I_{\ell,st}|$ over all pairs of nodes to obtain electronic betweenness for edge ℓ.
- (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 ℓ, $|I_{\ell,st}|$.
- Sum $|I_{\ell,st}|$ over all pairs of nodes to obtain electronic betweenness for edge ℓ.
- (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.
- 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: $\sum_j a_{ij}(V_i - V_j) = V_i \sum_j a_{ij} - \sum_j a_{ij} V_j$
$= V_i k_i - \sum_j a_{ij} V_j = k_i \delta_{ij} V_j - \sum_j a_{ij} V_j = [(\mathbf{K} - \mathbf{A})\vec{V}]_i$

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

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

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

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

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

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

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

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

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

Electronic betweenness

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

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

- $\mathbf{L} = \mathbf{K} - \mathbf{A}$ 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 $\mathbf{K} - \mathbf{A}$ is one dimensional.
- In fact, $\mathcal{N}(\mathbf{K} - \mathbf{A}) = \{c\vec{1}, c \in R\}$ since $(\mathbf{K} - \mathbf{A})\vec{1} = \vec{0}$.

Electronic betweenness

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

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

- $\mathbf{L} = \mathbf{K} - \mathbf{A}$ 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 $\mathbf{K} - \mathbf{A}$ is one dimensional.
- In fact, $\mathcal{N}(\mathbf{K} - \mathbf{A}) = \{c\vec{1}, c \in R\}$ since $(\mathbf{K} - \mathbf{A})\vec{1} = \vec{0}$.

Electronic betweenness

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

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

- $\mathbf{L} = \mathbf{K} - \mathbf{A}$ 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 $\mathbf{K} - \mathbf{A}$ is one dimensional.
- In fact, $\mathcal{N}(\mathbf{K} - \mathbf{A}) = \{c\vec{1}, c \in R\}$ since $(\mathbf{K} - \mathbf{A})\vec{1} = \vec{0}$.

Electronic betweenness

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

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

- $\mathbf{L} = \mathbf{K} - \mathbf{A}$ 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 $\mathbf{K} - \mathbf{A}$ is one dimensional.
- In fact, $\mathcal{N}(\mathbf{K} - \mathbf{A}) = \{c\vec{1}, c \in R\}$ since $(\mathbf{K} - \mathbf{A})\vec{1} = \vec{0}$.

Electronic betweenness

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

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

- $\mathbf{L} = \mathbf{K} - \mathbf{A}$ 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 $\mathbf{K} - \mathbf{A}$ is one dimensional.
- In fact, $\mathcal{N}(\mathbf{K} - \mathbf{A}) = \{c\vec{1}, c \in R\}$ since $(\mathbf{K} - \mathbf{A})\vec{1} = \vec{0}$.

Electronic betweenness

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

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

- $\mathbf{L} = \mathbf{K} - \mathbf{A}$ 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 $\mathbf{K} - \mathbf{A}$ is one dimensional.
- In fact, $\mathcal{N}(\mathbf{K} - \mathbf{A}) = \{c\vec{1}, c \in R\}$ since $(\mathbf{K} - \mathbf{A})\vec{1} = \vec{0}$.

Electronic betweenness

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

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

- $\mathbf{L} = \mathbf{K} - \mathbf{A}$ 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 $\mathbf{K} - \mathbf{A}$ is one dimensional.
- In fact, $\mathcal{N}(\mathbf{K} - \mathbf{A}) = \{c\vec{1}, c \in R\}$ since $(\mathbf{K} - \mathbf{A})\vec{1} = \vec{0}$.

Electronic betweenness

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

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

- $\mathbf{L} = \mathbf{K} - \mathbf{A}$ 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 $\mathbf{K} - \mathbf{A}$ is one dimensional.
- In fact, $\mathcal{N}(\mathbf{K} - \mathbf{A}) = \{c\vec{1}, c \in R\}$ since $(\mathbf{K} - \mathbf{A})\vec{1} = \vec{0}$.

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:

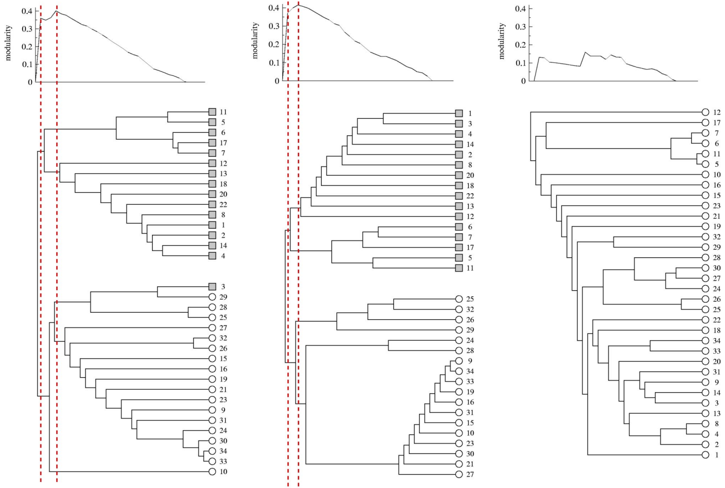
- Asking too much: Need full knowledge of network to travel along shortest paths.
- One of many alternatives: consider all random walks between pairs of nodes i and j.
- Walks starts at node i, traverses the network randomly, ending as soon as it reaches j.
- Record the number of times an edge is followed 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.

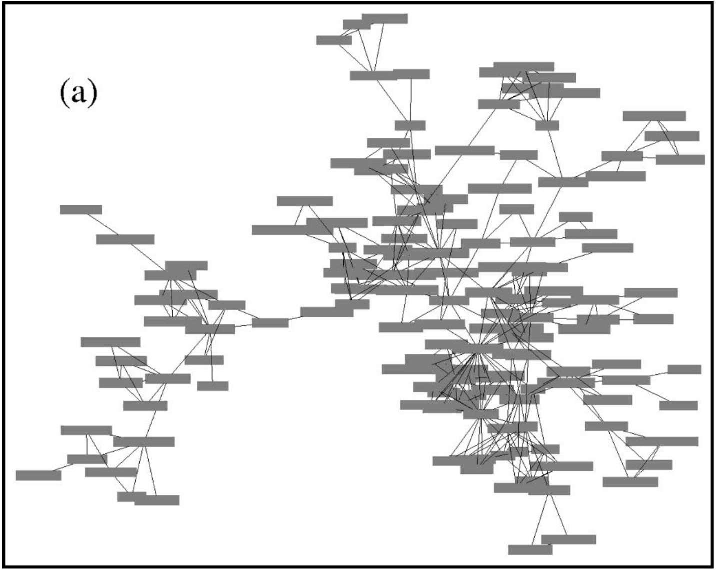
Hierarchy by division



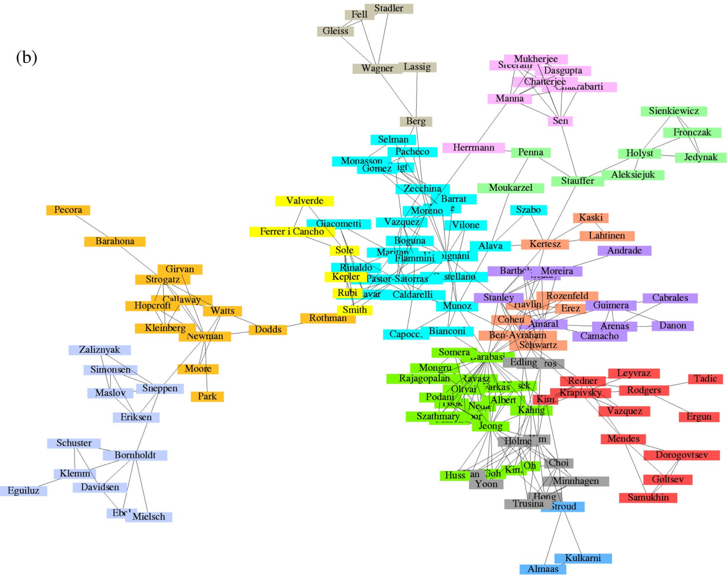
shortest path | random walk | shortest path without recalculation

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

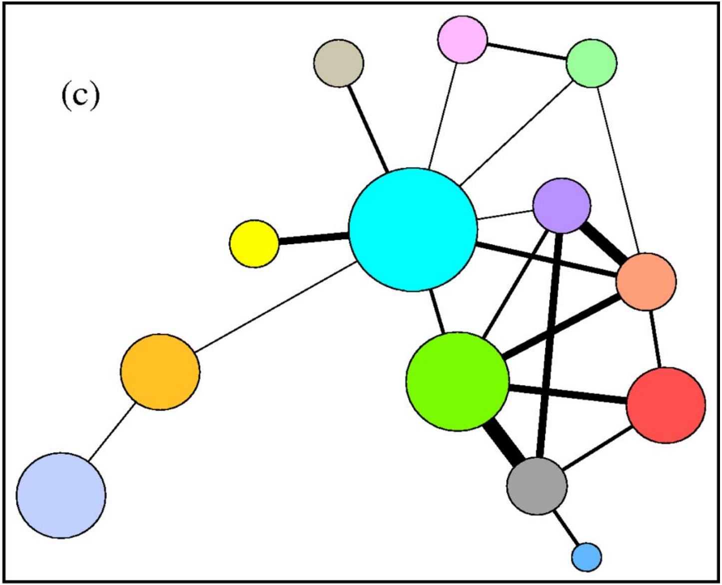
Scientists working on networks



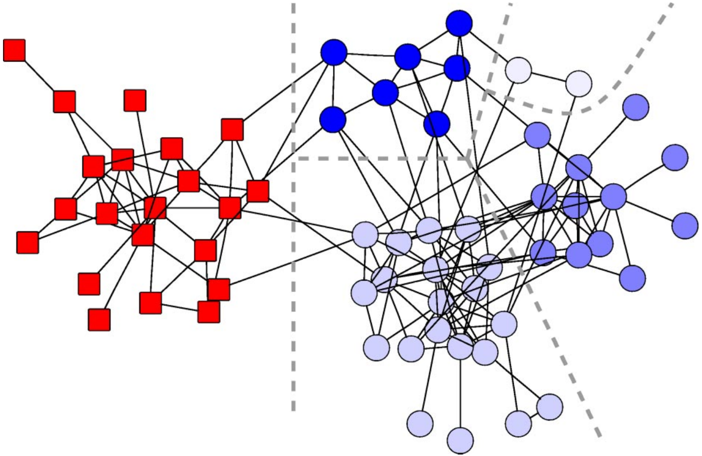
Scientists working on networks



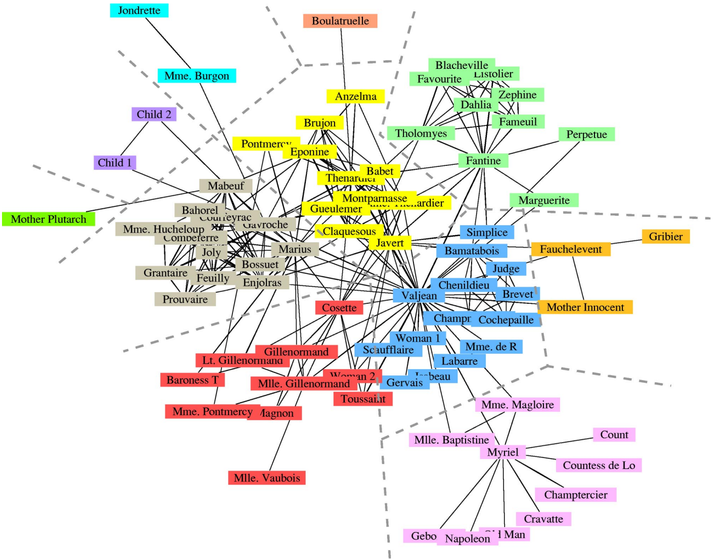
Scientists working on networks



Dolphins!



Les Miserables



Outline

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:

$$Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] = \mathrm{Tr}\mathbf{E} - ||\mathbf{E}^2||_1.$$

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:

$$Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] = \mathrm{Tr}\mathbf{E} - ||\mathbf{E}^2||_1.$$

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:

$$Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2] = \mathrm{Tr}\mathbf{E} - ||\mathbf{E}^2||_1 .$$

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** 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.

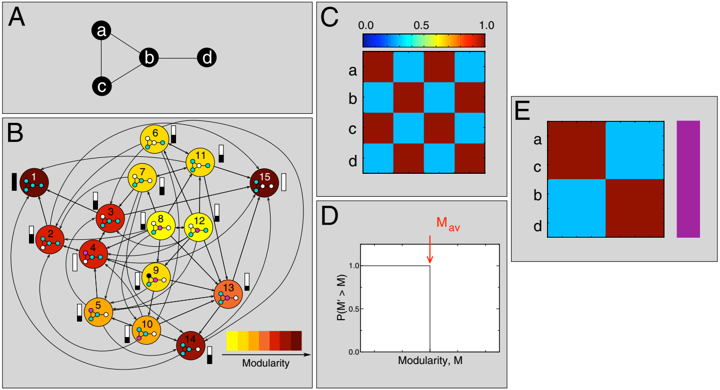
Shuffling for structure

- 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** 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

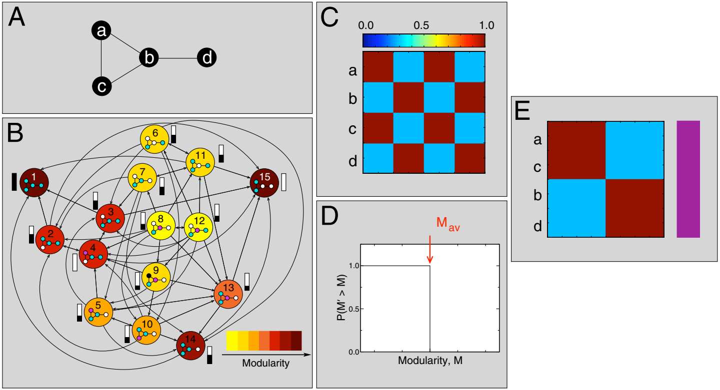
- Consider partition network, i.e., the network of all possible partitions.
- Defn: Two partitions are connected if they differ only by the reassignment of a single node.
- Look for local maxima in partition network.
- Construct an affinity matrix with entries A_{ij}.
- A_{ij} = **Pr** random walker 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



- **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, ...

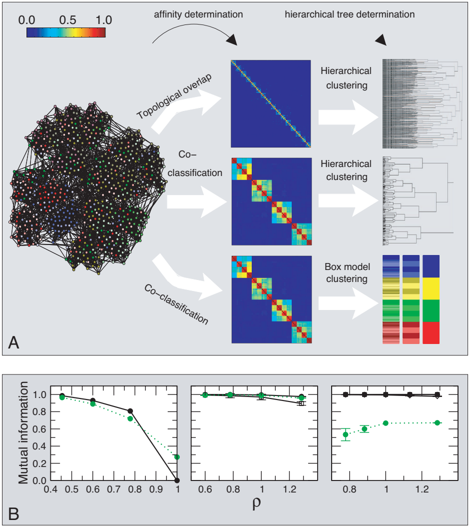
Shuffling for structure

- 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 $\mathbf{T}$ with entries $T_{ij} = f(|i-j|)$.
- Weird observation: if $T_{ij} = (i-j)^2$ then $\mathbf{T}$ 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

$$(\vec{v}_1)_i = \left(i - \frac{n+1}{2}\right),$$

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

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

- Remarkably,

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

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

Shuffling for structure

- Eigenvectors

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

- Remarkably,

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

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

Shuffling for structure

- Eigenvectors

$$(\vec{v}_1)_i = \left(i - \frac{n+1}{2}\right),$$

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

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

- Remarkably,

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

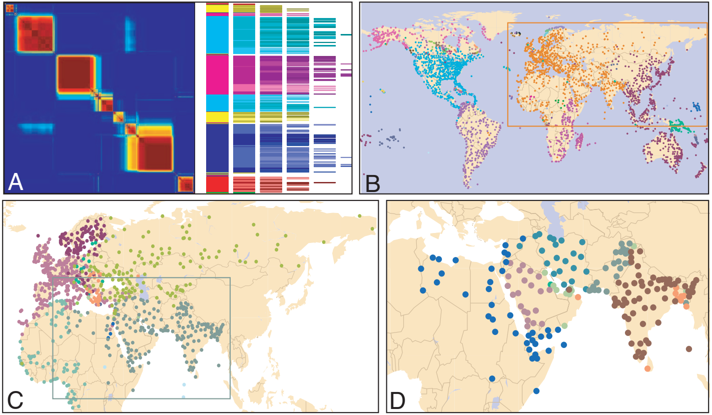
- The next step: figure out how to capitalize on this...

Shuffling for structure

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

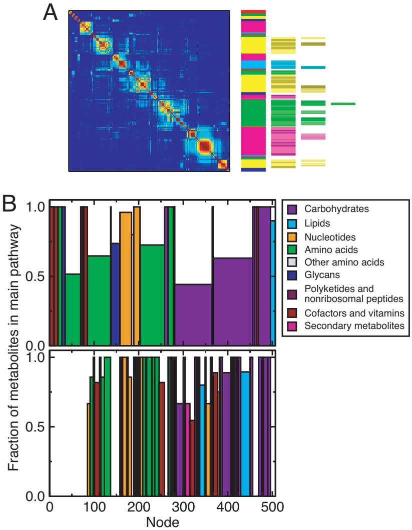
Network	Nodes	Edges	Modules	Main modules
Air transportation	3,618	28,284	57	8
E-mail	1,133	10,902	41	8
Electronic circuit	516	686	18	11
Escherichia coli KEGG	739	1,369	39	13
E. coli UCSD	507	947	28	17

Shuffling for structure



- Modules found match up with geopolitical units.

Shuffling for structure



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

Outline

General structure detection

- "Detecting communities in large networks" Capocci *et al.*(2005) [1]
- Consider normal matrix $\mathbf{K}^{-1}A$, random walk matrix $A^{\mathrm{T}}\mathbf{K}^{-1}$, Laplacian $\mathbf{K} - \mathbf{A}$, 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 $\mathbf{K}^{-1}A$, random walk matrix $A^{\mathrm{T}}\mathbf{K}^{-1}$, Laplacian $\mathbf{K} - \mathbf{A}$, 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

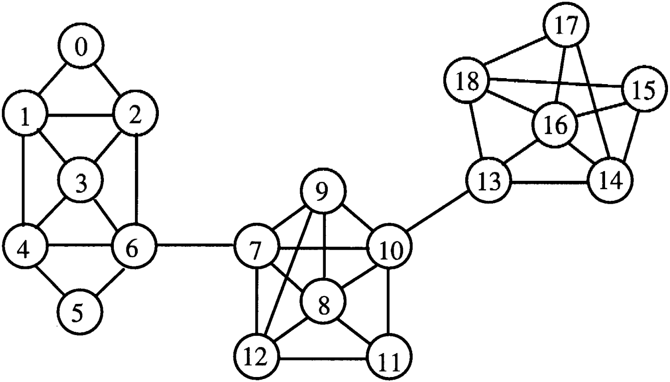
- "Detecting communities in large networks" Capocci *et al.*(2005) [1]
- Consider normal matrix $\mathbf{K}^{-1}A$, random walk matrix $A^{\mathrm{T}}\mathbf{K}^{-1}$, Laplacian $\mathbf{K} - \mathbf{A}$, and AA^{T}.
- Basic observation is that eigenvectors associated with secondary eigenvalues reveal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

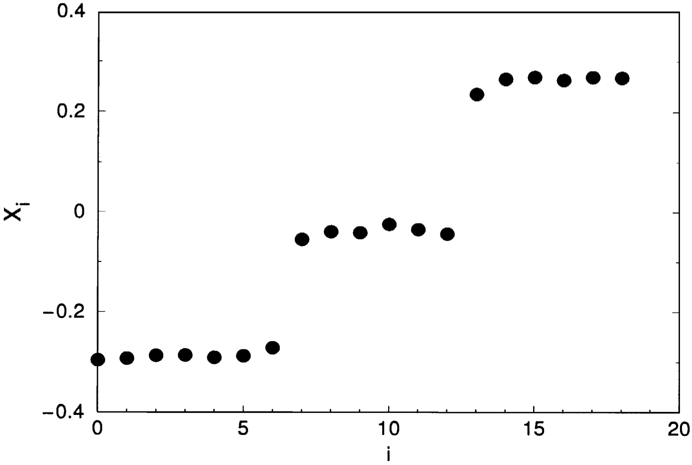
General structure detection

- Example network:



General structure detection

- Second eigenvector's components:



General structure detection

- Network of word associations for 10616 words.
- Average in-degree of 7.
- Using 2nd to 11th evectors of a modified version of $\mathbf{AA}^{\mathrm{T}}$:

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

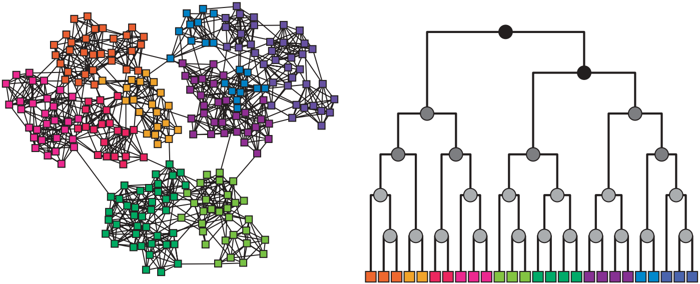
Science	1	Literature	1	Piano	1
Scientific	0.994	Dictionary	0.994	Cello	0.993
Chemistry	0.990	Editorial	0.990	Fiddle	0.992
Physics	0.988	Synopsis	0.988	Viola	0.990
Concentrate	0.973	Words	0.987	Banjo	0.988
Thinking	0.973	Grammar	0.986	Saxophone	0.985
Test	0.973	Adjective	0.983	Director	0.984
Lab	0.969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

Values indicate the correlation.

Outline

Hierarchies and missing links

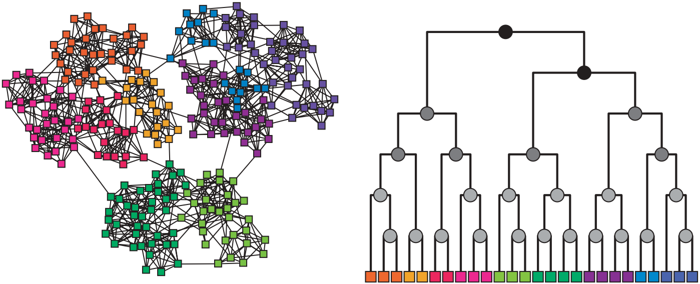
Clauset *et al.*, Nature (2008) [2]



- 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

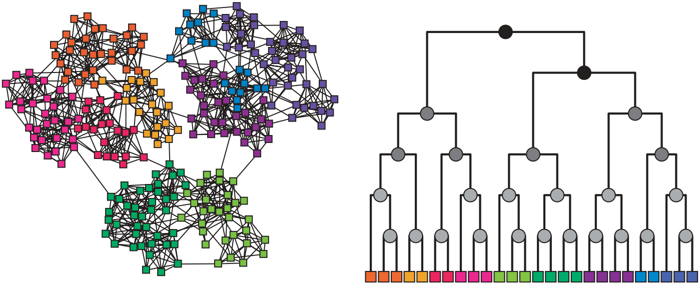
Clauset *et al.*, Nature (2008) [2]



- 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

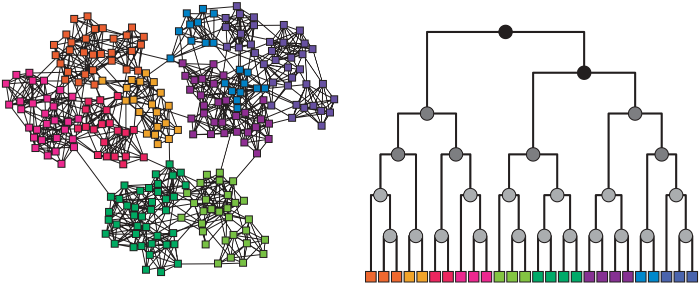
Clauset *et al.*, Nature (2008) [2]



- 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

Clauset *et al.*, Nature (2008) [2]



- 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

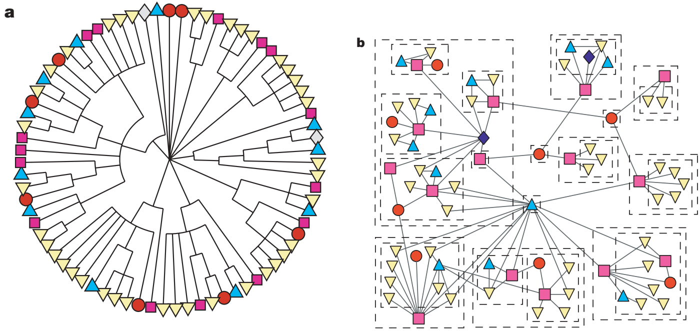
- Model also predicts reasonably well
  1. average degree,
  2. clustering,
  3. and average shortest path length.

Table 1 | Comparison of original and resampled networks

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

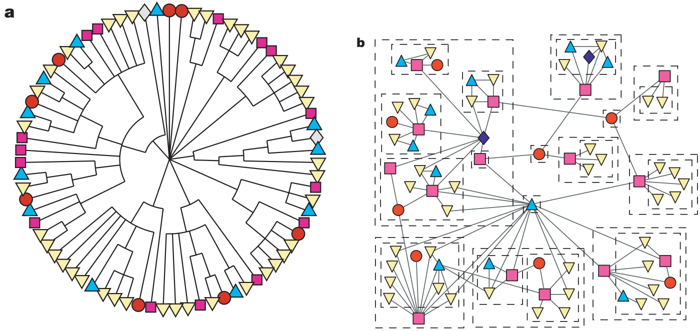
Statistics are shown for the three example networks studied and for new networks generated by resampling from our hierarchical model. The generated networks closely match the average degree $\langle k \rangle$, clustering coefficient C and average vertex–vertex distance d in each case, suggesting that 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

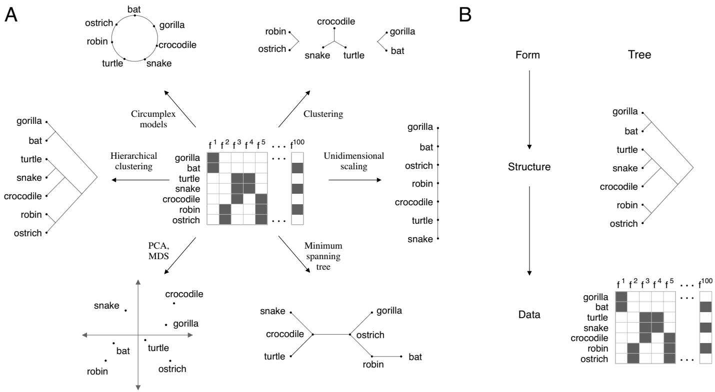


- Consensus dendogram for grassland species.
- Copes with disassortative and assortative communities.

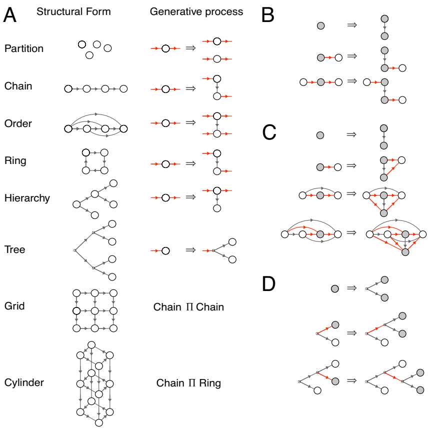
Outline

General structure detection

- “The discovery of structural form”
  Kemp and Tenenbaum, PNAS (2008) [4]

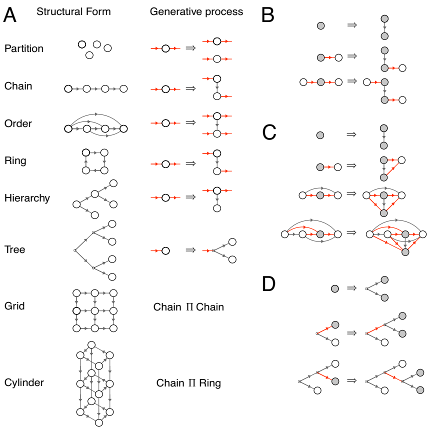


General structure detection



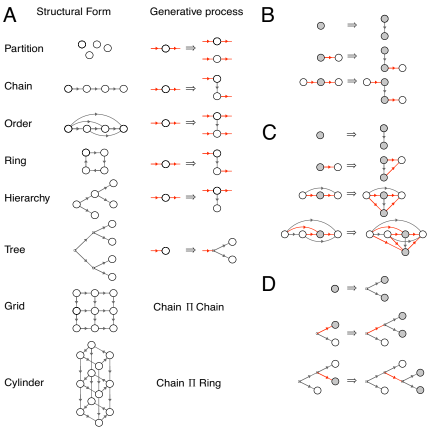
- Top down description of form.
- Node replacement graph grammar: parent node becomes two child nodes.
- B-D: Growing chains, orders, and trees.

General structure detection



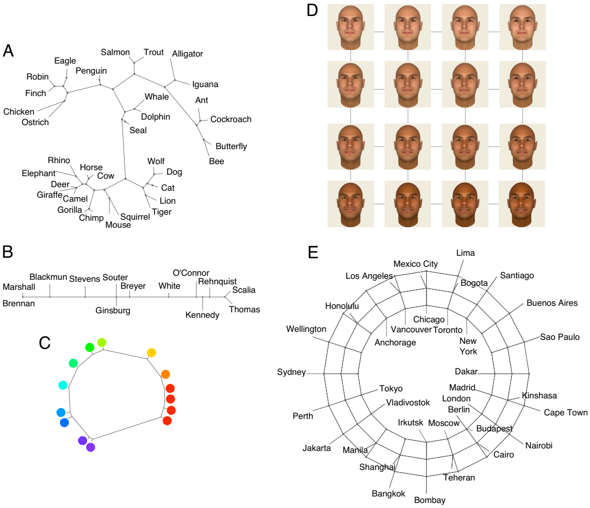
- Top down description of form.
- Node replacement graph grammar: parent node becomes two child nodes.
- B-D: Growing chains, orders, and trees.

General structure detection



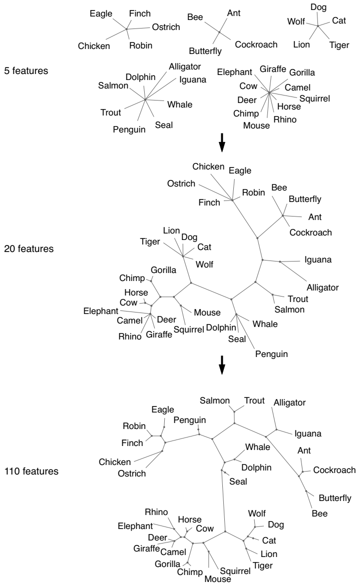
- Top down description of form.
- Node replacement graph grammar: parent node becomes two child nodes.
- B-D: Growing chains, orders, and trees.

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

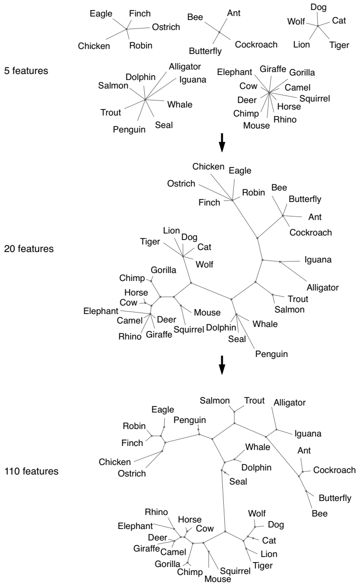
⇓

simple tree

⇓

complex tree

General structure detection



- Effect of adding features on detected form.

Straight partition

⇓

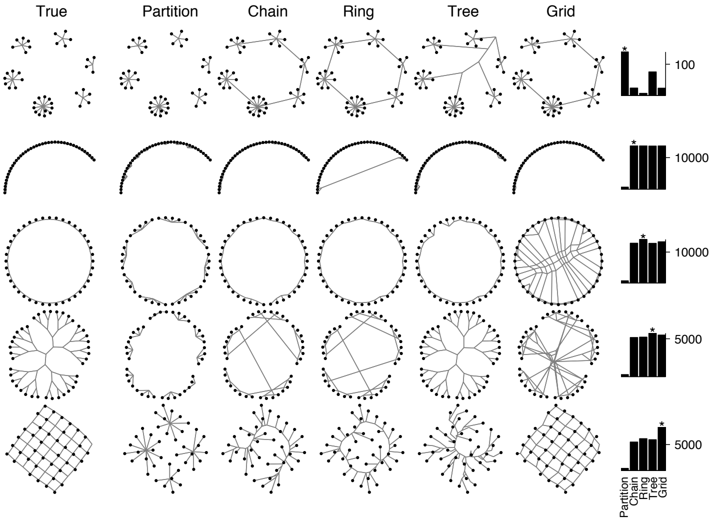
simple tree

⇓

complex tree

General structure detection

- Performance for test networks.



Final words:

Modern science in three steps:

1. Find interesting/meaningful/important phenomena involving spectacular amounts of data.
2. Describe what you see.
3. Explain it.

Final words:

Modern science in three steps:

1. Find interesting/meaningful/important phenomena involving spectacular amounts of data.
2. Describe what you see.
3. Explain it.

Final words:

Modern science in three steps:

1. Find interesting/meaningful/important phenomena involving spectacular amounts of data.
2. Describe what you see.
3. Explain it.

References I

[1] A. Capocci, V. Servedio, G. Caldarelli, and F. Colaiori.
Detecting communities in large networks.
Physica A: Statistical Mechanics and its Applications, 352:669–676, 2005. pdf (⊞)

[2] A. Clauset, C. Moore, and M. E. J. Newman.
Hierarchical structure and the prediction of missing links in networks.
Nature, 453:98–101, 2008. pdf (⊞)

[3] M. Girvan and M. E. J. Newman.
Community structure in social and biological networks.
Proc. Natl. Acad. Sci., 99:7821–7826, 2002. pdf (⊞)

References II

[4] C. Kemp and J. B. Tenenbaum.
The discovery of structural form.
Proc. Natl. Acad. Sci., 105:10687–10692, 2008.
pdf (⊞)

[5] M. E. J. Newman.
Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality.
Phys. Rev. E, 64(1):016132, 2001. pdf (⊞)

[6] M. E. J. Newman.
Erratum: Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality [Phys. Rev. E 64, 016132 (2001)].
Phys. Rev. E, 73:039906(E), 2006. pdf (⊞)

References III

[7] M. E. J. Newman and M. Girvan.
Finding and evaluating community structure in networks.
Phys. Rev. E, 69(2):026113, 2004. pdf (⊞)

[8] M. Sales-Pardo, R. Guimerà, A. A. Moreira, and L. A. N. Amaral.
Extracting the hierarchical organization of complex systems.
Proc. Natl. Acad. Sci., 104:15224–15229, 2007. pdf (⊞)

[9] M. Sales-Pardo, R. Guimerà, A. A. Moreira, and L. A. N. Amaral.
Extracting the hierarchical organization of complex systems: Correction.
Proc. Natl. Acad. Sci., 104:18874, 2007. pdf (⊞)

References IV

[10] W. W. Zachary.
An information flow model for conflict and fission in small groups.
J. Anthropol. Res., 33:452–473, 1977.