Large-scale structure of complex networks (Part 2)

Snehal M. Shekatkar

Centre for modeling and simulation,
S.P. Pune University, Pune

2017-12-30

Large-scale structure of complex networks (Part 2)

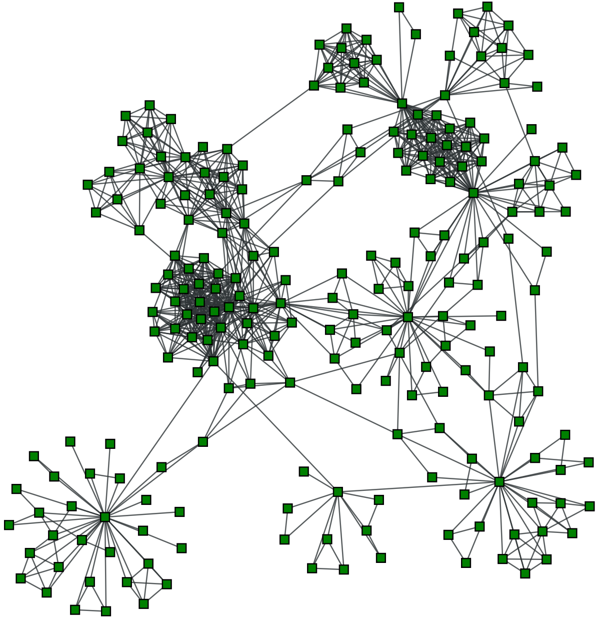
Large-scale structure of complex networks (Part 2)

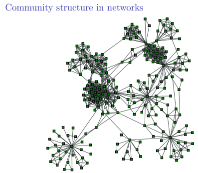
Snehal M. Shekatkar

Centre for modeling and simulation,
S.P. Pune University, Pune

Hello

Community structure in networks





Network of coauthorships in a university department

Community structure in networks

What are communities?

- **Traditional definition**: Groups of nodes with a high internal link density
- **Modern definition**: Nodes with similar connection probabilities to the rest of the network

Community structure in networks

What are communities?

- Traditional definition: Groups of nodes with a high internal link density
- Modern definition: Nodes with similar connection probabilities to the rest of the network

Communities in the real-world networks

- **Social networks**:
  - Friend-circles
  - Research communities
  - Co-workers
- **World Wide Web**:
  - Pages with similar contents
  - Webpages under the same domain (e.g. Wikipedia)
- **Biological networks**:
  - Proteins with similar roles in protein interaction networks
  - Chemicals together taking part in chemical reactions in metabolic networks
  - Communities in neuronal networks

Communities in the real-world networks

- **Social networks**:
  - Friend-circles
  - Research communities
  - Co-workers
- **World Wide Web**:
  - Pages with similar contents
  - Webpages under the same domain (e.g. Wikipedia)
- **Biological networks**:
  - Proteins with similar roles in protein interaction networks
  - Chemicals together taking part in chemical reactions in metabolic networks
  - Communities in neuronal networks

Community detection

Detecting communities is important!

- Communities are building blocks of networks
- Communities allow us to see "the big picture"
- Functional/Autonomous units
- Non-trivial effects on the processes on networks

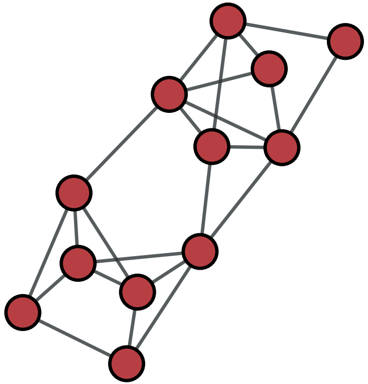
Community detection

Detecting communities is important!

- Communities are building blocks of networks
- Communities allow us to see "the big picture"
- Functional/Autonomous units
- Non-trivial effects on the processes on networks

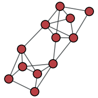
Graph partitioning

Problem of dividing a graph in a given number of groups of given sizes such that the number of links between the groups (**cut size**) is minimized



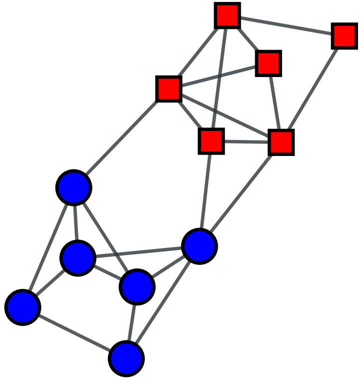
Graph partitioning

Problem of dividing a graph in a given number of groups of given sizes such that the number of links between the groups (**cut size**) is minimized

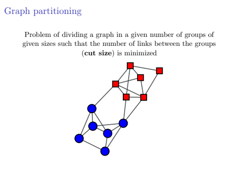


Graph partitioning

Problem of dividing a graph in a given number of groups of given sizes such that the number of links between the groups (**cut size**) is minimized



└Graph partitioning



Partitioning is hard!

- Graph with n vertices
- Find two groups with sizes n_1 and n_2 such that the cut size is minimum
- Number of ways: $\frac{n!}{n_1!n_2!} \approx \frac{2^{n+1}}{\sqrt{n}}$

Heuristics are needed!

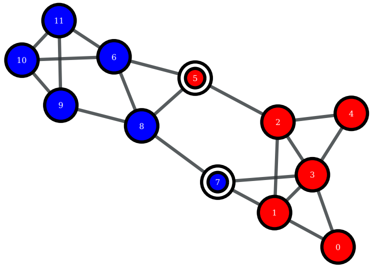
Partitioning is hard!

- Graph with n vertices
- Find two groups with sizes n_1 and n_2 such that the cut size is minimum
- Number of ways: $\frac{n!}{n_1!n_2!} \approx \frac{2^{n+1}}{\sqrt{n}}$

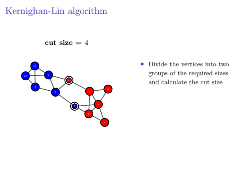
Heuristics are needed!

Kernighan-Lin algorithm

cut size $= 4$

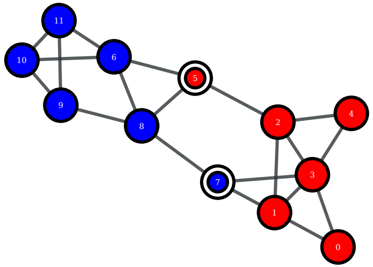


- Divide the vertices into two groups of the required sizes and calculate the cut size

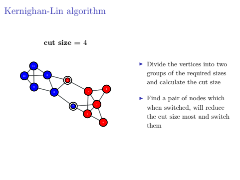


Kernighan-Lin algorithm

cut size $= 4$

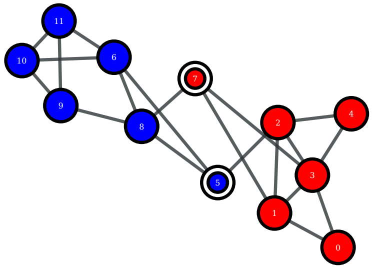


- Divide the vertices into two groups of the required sizes and calculate the cut size
- Find a pair of nodes which when switched, will reduce the cut size most and switch them



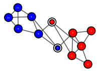
Kernighan-Lin algorithm

cut size $= 2$



- Divide the vertices into two groups of the required sizes
- Find a pair of nodes which when switched, will reduce the cut size most and switch them

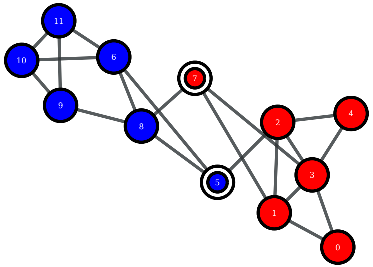
Kernighan-Lin algorithm
cut size = 2



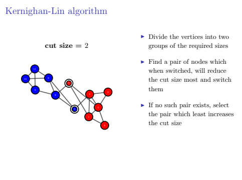
- Divide the vertices into two groups of the required sizes
- Find a pair of nodes which when switched, will reduce the cut size most and switch them

Kernighan-Lin algorithm

cut size = 2

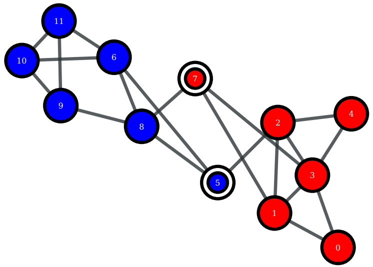


- Divide the vertices into two groups of the required sizes
- Find a pair of nodes which when switched, will reduce the cut size most and switch them
- If no such pair exists, select the pair which least increases the cut size

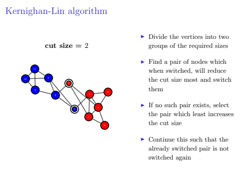


Kernighan-Lin algorithm

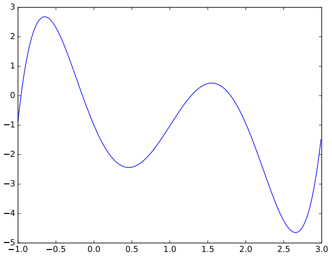
cut size = 2



- Divide the vertices into two groups of the required sizes
- Find a pair of nodes which when switched, will reduce the cut size most and switch them
- If no such pair exists, select the pair which least increases the cut size
- Continue this such that the already switched pair is not switched again

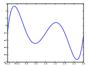


Kernighan-Lin algorithm



- Go through all the states and select the one with the least cut size
- Start with this state and repeat the whole procedure
- Continue till the cut size no longer becomes smaller
- Starting with many random initial conditions is better

Kernighan-Lin algorithm



- Go through all the states and select the one with the least cut size
- Start with this state and repeat the whole procedure
- Continue till the cut size no longer becomes smaller
- Starting with many random initial conditions is better

Group sizes remain constant

Spectral partitioning

- Faster algorithm than Kernighan-Lin
- Uses properties of the graph Laplacian
- More complex to implement than Kernighan-Lin

Spectral partitioning

- Faster algorithm than Kernighan-Lin
- Uses properties of the graph Laplacian
- More complex to implement than Kernighan-Lin

Spectral partitioning

Cut size:

$$R = \frac{1}{2} \sum_{\substack{i,j \text{ in} \\ \text{different} \\ \text{groups}}} A_{ij}$$

Define

$$s_i = \begin{cases} +1 & \text{if vertex } i \text{ belongs to group 1} \\ -1 & \text{if vertex } i \text{ belongs to group 2} \end{cases}$$

Then

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are in different groups,} \\ 0 & \text{if } i \text{ and } j \text{ are in the same group} \end{cases}$$

Spectral partitioning

Cut size:

$$R = \frac{1}{2} \sum_{\substack{i,j \text{ in} \\ \text{different} \\ \text{groups}}} A_{ij}$$

Define

$$s_i = \begin{cases} +1 & \text{if vertex } i \text{ belongs to group 1} \\ -1 & \text{if vertex } i \text{ belongs to group 2} \end{cases}$$

Then

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are in different groups,} \\ 0 & \text{if } i \text{ and } j \text{ are in the same group} \end{cases}$$

Spectral partitioning

$$R = \frac{1}{4}\sum_{ij} A_{ij}(1 - s_i s_j)$$

First term,

$$\sum_{ij} A_{ij} = \sum_i k_i = \sum_i k_i s_i^2 = \sum_{ij} k_i \delta_{ij} s_i s_j$$

$$R = \frac{1}{4}\sum_{ij}(k_i\delta_{ij} - A_{ij})s_i s_j = \frac{1}{4}\sum_{ij} L_{ij} s_i s_j$$

$$R = \frac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s}$$

Spectral partitioning

$$R = \frac{1}{4}\sum_{ij} A_{ij}(1 - s_i s_j)$$

First term,

$$\sum_{ij} A_{ij} = \sum_i k_i = \sum_i k_i s_i^2 = \sum_{ij} k_i \delta_{ij} s_i s_j$$

$$R = \frac{1}{4}\sum_{ij}(k_i\delta_{ij} - A_{ij})s_i s_j = \frac{1}{4}\sum_{ij} L_{ij} s_i s_j$$

$$R = \frac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s}$$

L is so imp that we have a name for it! Laplacian

.

s is a columnvector

.

L: structure, s: division

.

find s that minimizes R

.

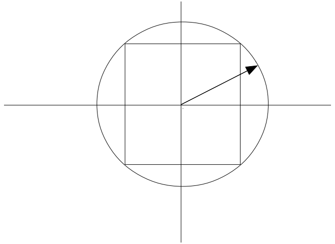
Problem is hard, s takes only integer values

.

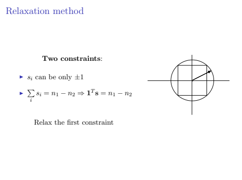
Relaxation method

Two constraints:

- s_i can be only ± 1
- $\sum_i s_i = n_1 - n_2 \Rightarrow \mathbf{1}^T \mathbf{s} = n_1 - n_2$



Relax the first constraint


Relaxation method
Two constraints:
s_i can be only ± 1
$\sum_i s_i = n_1 - n_2 \Rightarrow \mathbf{1}^T \mathbf{s} = n_1 - n_2$
Relax the first constraint

hypercube

.

continuous s, differentiate

Spectral partitioning

Minimization with constraints ⇒ Lagrange multipliers

$$\frac{\partial}{\partial s_i}\left[\sum_{jk} L_{jk} s_j s_k + \lambda\left(n - \sum_j s_j^2\right) + 2\mu\left((n_1 - n_2) - \sum_j s_j\right)\right] = 0$$

Spectral partitioning

Minimization with constraints ⇒ Lagrange multipliers

$$\frac{\partial}{\partial s_i}\left[\sum_{jk} L_{jk} s_j s_k + \lambda\left(n - \sum_j s_j^2\right) + 2\mu\left((n_1 - n_2) - \sum_j s_j\right)\right] = 0$$

Spectral partitioning

Minimization with constraints ⇒ Lagrange multipliers

$$\frac{\partial}{\partial s_i}\left[\sum_{jk} L_{jk} s_j s_k + \lambda\left(n - \sum_j s_j^2\right) + 2\mu\left((n_1 - n_2) - \sum_j s_j\right)\right] = 0$$

$$\sum_j L_{ij} s_j = \lambda s_i + \mu$$

Spectral partitioning

Minimization with constraints ⇒ Lagrange multipliers

$$\frac{\partial}{\partial s_i}\left[\sum_{jk} L_{jk} s_j s_k + \lambda\left(n - \sum_j s_j^2\right) + 2\mu\left((n_1 - n_2) - \sum_j s_j\right)\right] = 0$$

$$\sum_j L_{ij} s_j = \lambda s_i + \mu$$

Spectral partitioning

Minimization with constraints ⇒ Lagrange multipliers

$$\frac{\partial}{\partial s_i}\left[\sum_{jk} L_{jk} s_j s_k + \lambda\left(n - \sum_j s_j^2\right) + 2\mu\left((n_1 - n_2) - \sum_j s_j\right)\right] = 0$$

$$\sum_j L_{ij} s_j = \lambda s_i + \mu$$

$$\mathbf{L}\mathbf{s} = \lambda \mathbf{s} + \mu \mathbf{1} = \lambda\left(\mathbf{s} + \frac{\mu}{\lambda}\mathbf{1}\right)$$

$$\mathbf{L}\left(\mathbf{s} + \frac{\mu}{\lambda}\mathbf{1}\right) = \lambda\left(\mathbf{s} + \frac{\mu}{\lambda}\mathbf{1}\right)$$

$\mathbf{1}$ is an eigenvector of the Laplacian with eigenvalue 0

$$\mathbf{L}\mathbf{x} = \lambda\mathbf{x}$$

Spectral partitioning

Minimization with constraints ⇒ Lagrange multipliers

$$\frac{\partial}{\partial s_i}\left[\sum_{jk} L_{jk} s_j s_k + \lambda\left(n - \sum_j s_j^2\right) + 2\mu\left((n_1 - n_2) - \sum_j s_j\right)\right] = 0$$

$$\sum_j L_{ij} s_j = \lambda s_i + \mu$$

$$\mathbf{L}\mathbf{s} = \lambda \mathbf{s} + \mu \mathbf{1} = \lambda\left(\mathbf{s} + \frac{\mu}{\lambda}\mathbf{1}\right)$$

$$\mathbf{L}\left(\mathbf{s} + \frac{\mu}{\lambda}\mathbf{1}\right) = \lambda\left(\mathbf{s} + \frac{\mu}{\lambda}\mathbf{1}\right)$$

$\mathbf{1}$ is an eigenvector of the Laplacian with eigenvalue 0

$$\mathbf{L}\mathbf{x} = \lambda\mathbf{x}$$

Spectral partitioning

$\mathbf{x}$ is an eigenvector of the Laplacian with eigenvalue λ

Which eigenvector to choose?

$$\mathbf{x} \text{ cannot be the eigenvector } \mathbf{1} = \begin{pmatrix} 1 \\ 1 \\ . \\ . \\ 1 \end{pmatrix}$$

$$\mathbf{1}^T\mathbf{x} = \mathbf{1}\left(\mathbf{s} + \frac{\mu}{\lambda}\mathbf{1}\right) = (n_1 - n_2) + \frac{\mu}{\lambda}n = 0$$

Spectral partitioning

$\mathbf{x}$ is an eigenvector of the Laplacian with eigenvalue λ

Which eigenvector to choose?

$$\mathbf{x} \text{ cannot be the eigenvector } \mathbf{1} = \begin{pmatrix} 1 \\ 1 \\ . \\ . \\ 1 \end{pmatrix}$$

$$\mathbf{1}^T\mathbf{x} = \mathbf{1}\left(\mathbf{s} + \frac{\mu}{\lambda}\mathbf{1}\right) = (n_1 - n_2) + \frac{\mu}{\lambda}n = 0$$

x is orthogonal to 1

.

x is eigenvector but not 1

Spectral partitioning

Which eigenvector to choose?

$$R = \frac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s} = \frac{1}{4}\mathbf{x}^T\mathbf{x} = \frac{n_1 n_2}{n}\lambda$$

Choose the eigenvector with smallest possible eigenvalue!

Eigenvalues of the Laplacian are non-negative and smallest is always 0

$\mathbf{v}_1 = \mathbf{1}$ is ruled out already. So choose $\mathbf{v}_2$ with the smallest positive eigenvalue

Spectral partitioning

Which eigenvector to choose?

$$R = \frac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s} = \frac{1}{4}\mathbf{x}^T\mathbf{x} = \frac{n_1 n_2}{n}\lambda$$

Choose the eigenvector with smallest possible eigenvalue!

Eigenvalues of the Laplacian are non-negative and smallest is always 0

$\mathbf{v}_1 = \mathbf{1}$ is ruled out already. So choose $\mathbf{v}_2$ with the smallest positive eigenvalue

Spectral partitioning

$$\mathbf{s} = x + \frac{n_1 - n_2}{n}\mathbf{1}$$

OR

$$s_i = x_i + \frac{n_1 - n_2}{n}$$

But s_i can be only ± 1

Thus, we want $\mathbf{x}$ to be as close as possible to $\mathbf{s}$

Spectral partitioning

$$\mathbf{s} = x + \frac{n_1 - n_2}{n}\mathbf{1}$$

OR

$$s_i = x_i + \frac{n_1 - n_2}{n}$$

But s_i can be only ± 1

Thus, we want $\mathbf{x}$ to be as close as possible to $\mathbf{s}$

Spectral partitioning

Maximize:

$$\mathbf{s}^T \left(\mathbf{x} + \frac{n_1 - n_2}{n} \mathbf{1} \right) = \sum_i s_i \left(x_i + \frac{n_1 - n_2}{n} \right)$$

Equivalently, maximize:

$$\sum_i s_i x_i$$

Simply put the n_1 vertices with most positive elements in group 1 and the rest ones in group 2

Group assignments are arbitrary

Spectral partitioning

Maximize:

$$\mathbf{s}^T \left(\mathbf{x} + \frac{n_1 - n_2}{n} \mathbf{1} \right) = \sum_i s_i \left(x_i + \frac{n_1 - n_2}{n} \right)$$

Equivalently, maximize:

$$\sum_i s_i x_i$$

Simply put the n_1 vertices with most positive elements in group 1 and the rest ones in group 2

Group assignments are arbitrary

Choose the assignment with the smaller cut size

Spectral partitioning

- Calculate $\mathbf{v}_2$ of the Laplacian
- Put vertices corresponding to largest n_1 elements in group 1 and others in group 2. Calculate the cut size
- Put vertices corresponding to smallest n_1 elements in group 1 and others in group 2. Calculate the cut size
- Choose the division with the smallest cut size among the two

Spectral partitioning

- Calculate $\mathbf{v}_2$ of the Laplacian
- Put vertices corresponding to largest n_1 elements in group 1 and others in group 2. Calculate the cut size
- Put vertices corresponding to smallest n_1 elements in group 1 and others in group 2. Calculate the cut size
- Choose the division with the smallest cut size among the two

Community detection is harder!

- **Graph partitioning**
  - well defined
  - Number of groups is fixed
  - Sizes of the groups are fixed
  - Divide even if no good division exists
- **Community detection**
  - ill-defined
  - Number of groups depends on the structure of the network
  - Sizes of the groups depend on the structure of the network
  - Discover natural fault lines

Community detection is harder!

- **Graph partitioning**
  - well defined
  - Number of groups is fixed
  - Sizes of the groups are fixed
  - Divide even if no good division exists
- **Community detection**
  - ill-defined
  - Number of groups depends on the structure of the network
  - Sizes of the groups depend on the structure of the network
  - Discover natural fault lines

Many definitions.. many algorithms!

- Girvan-Newman algorithm
- Kernighan-Lin-Newman algorithm
- Spectral decomposition
- Clique-percolation
- Radom walk methods
- Statistical inference
- Label propagation
- Hierarchical clustering

Many definitions.. many algorithms!

- Girvan-Newman algorithm
- Kernighan-Lin-Newman algorithm
- Spectral decomposition
- Clique-percolation
- Radom walk methods
- Statistical inference
- Label propagation
- Hierarchical clustering

I can go on.. These algorithms use different definitions/views of communities

Broad classification

- **Agglomerative algorithms:**
  - Hierarchical clustering
  - Louvain method
  - CNM algorithm
- **Divisive algorithms:**
  - Girvan-Newman algorithm
  - Radichhi algorithm
- **Assignment algorithms:**
  - Label propagation
  - Spectral partitioning
  - Kernighan-Lin-Newman algorithm

Broad classification
- **Agglomerative algorithms:**
  - Hierarchical clustering
  - Louvain method
  - CNM algorithm
- **Divisive algorithms:**
  - Girvan-Newman algorithm
  - Radichhi algorithm
- **Assignment algorithms:**
  - Label propagation
  - Spectral partitioning
  - Kernighan-Lin-Newman algorithm

"The" simplest community detection problem

- Bisecting a graph with n nodes
- Group sizes are not fixed
- Minimum cut size?

"The" simplest community detection problem

- Bisecting a graph with n nodes
- Group sizes are not fixed
- Minimum cut size?

Empty group

"The" simplest community detection problem

- Bisecting a graph with n nodes
- Group sizes are not fixed
- Minimum cut size?

A different measure of the quality of division is required..

"The" simplest community detection problem

- Bisecting a graph with n nodes
- Group sizes are not fixed
- Minimum cut size?

A different measure of the quality of division is required..

Different measure

Quantification of community structure

- Fewer than expected edges between the groups

Quantification of community structure

- Fewer than expected edges between the groups

few edges = expected edges = not a good division

Quantification of community structure

- Fewer than expected edges between the groups
- Equivalently, more than expected edges inside the groups

Quantification of community structure
- Fewer than expected edges between the groups
- Equivalently, more than expected edges inside the groups

Remember assortativity

Quantification of community structure

- Fewer than expected edges between the groups
- Equivalently, more than expected edges inside the groups
- Assortativity mixing and modularity

Quantification of community structure

- Fewer than expected edges between the groups
- Equivalently, more than expected edges inside the groups
- Assortativity mixing and modularity

Divide network using modularity

Quantification of community structure

- Fewer than expected edges between the groups
- Equivalently, more than expected edges inside the groups
- Assortativity mixing and modularity
- Look for divisions with high modularity

Quantification of community structure
- Fewer than expected edges between the groups
- Equivalently, more than expected edges inside the groups
- Assortativity mixing and modularity
- Look for divisions with high modularity

Heuristics are needed

Quantification of community structure

- Fewer than expected edges between the groups
- Equivalently, more than expected edges inside the groups
- Assortativity mixing and modularity
- Look for divisions with high modularity
- Modularity maximization is hard

Quantification of community structure

- Fewer than expected edges between the groups
- Equivalently, more than expected edges inside the groups
- Assortativity mixing and modularity
- Look for divisions with high modularity
- Modularity maximization is hard

Newman-Girvan algorithm

- Look for edges between the communities
- Edge betweenness

2017-12-30

Large-scale structure of complex networks (Part 2)

└─Newman-Girvan algorithm

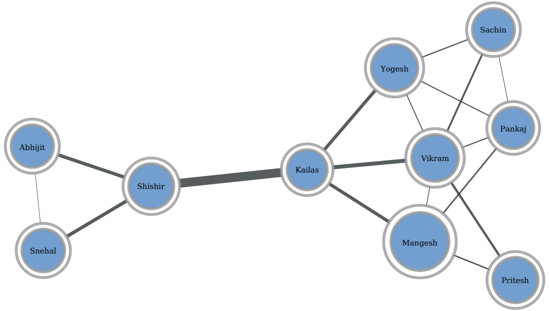
Newman-Girvan algorithm

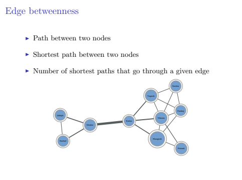
- Look for edges between the communities
- Edge betweenness

Let's have a look at the edge betweenness

Edge betweenness

- Path between two nodes
- Shortest path between two nodes
- Number of shortest paths that go through a given edge





The algorithm

- Calculate betweenness for all edges
- Remove the edge with the highest betweenness
- Recalculate betweenness for all edges
- Repeat

The algorithm

- Calculate betweenness for all edges
- Remove the edge with the highest betweenness
- Recalculate betweenness for all edges
- Repeat