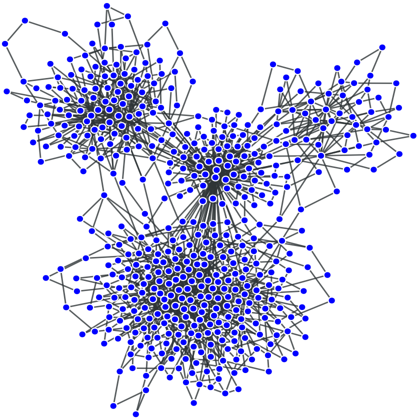
Large-scale structure of complex networks (Part 2)

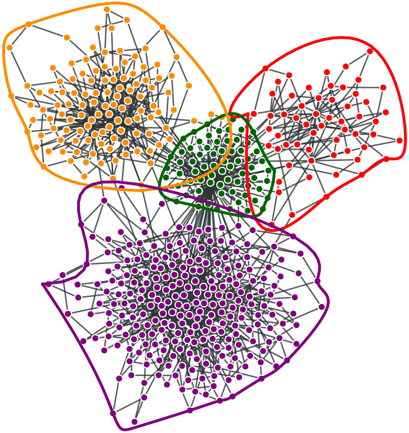
Snehal M. Shekatkar

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

Community structure in networks



Community structure in networks



Community structure in networks

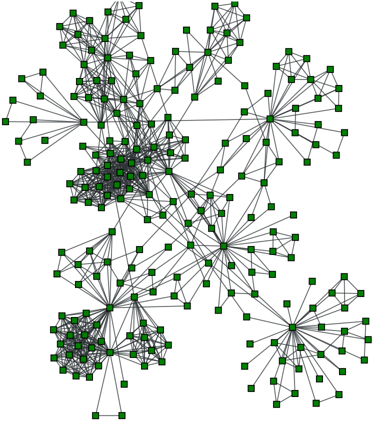
What are communities?

- **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

Community detection

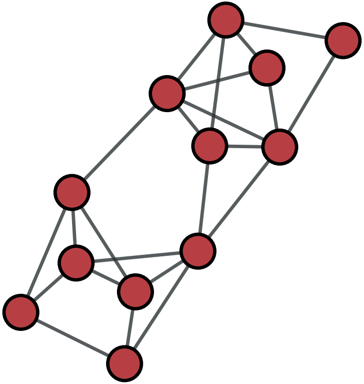


Detecting communities is important!

- 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

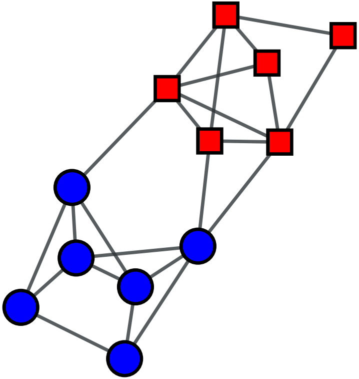
Graph partitioning

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



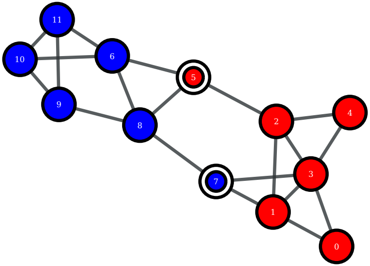
Partitioning is hard!

- 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

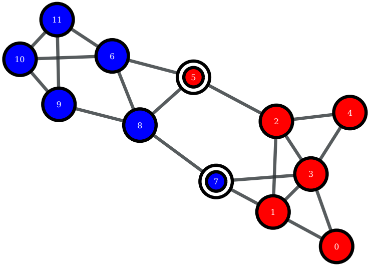
cut size = 4



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

Kernighan-Lin algorithm

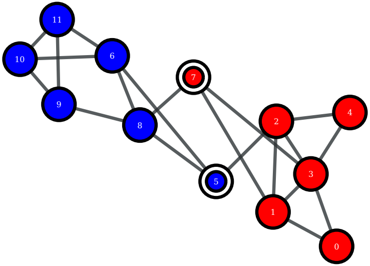
cut size $= 4$



- 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

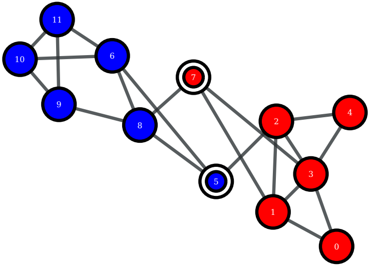
cut size $= 2$



- 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

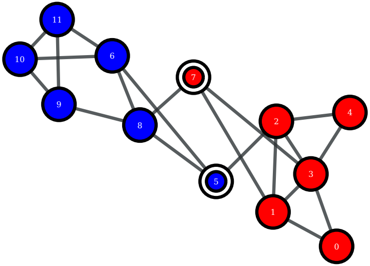
cut size = 2



- 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
- If no such pair exists, select the pair which least increases the cut size

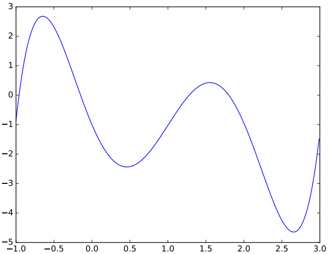
Kernighan-Lin algorithm

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

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

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

First term,

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

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

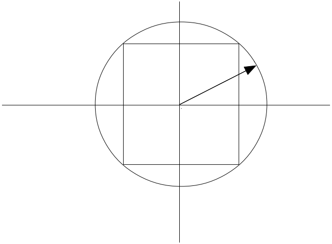
$$R = \frac{1}{4} \mathbf{s}^T \mathbf{L} \mathbf{s}$$

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



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

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

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

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

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

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

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

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

- 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

Many definitions.. many algorithms!

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

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?
  - Trivial partition
  - Needs ad hoc specification of sizes

“The” simplest community detection problem

- Bisecting a graph with n nodes
- Group sizes are not fixed
- Minimum cut size?
  - Trivial partition
  - Needs ad hoc specification of sizes

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

Quantification of community structure

- Fewer than expected edges between the groups

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
- 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
- Look for divisions with high modularity

Modularity

How to find the division which maximizes the modularity?

Modularity

How to find the division which maximizes the modularity?

- Check the value of Q for all possible divisions and choose the best one

Modularity

How to find the division which maximizes the modularity?

- Check the value of Q for all possible divisions and choose the best one
- Consider, $N = 100$, $n_1 = n_2 = 50$

Modularity

How to find the division which maximizes the modularity?

- Check the value of Q for all possible divisions and choose the best one
- Consider, $N = 100$, $n_1 = n_2 = 50$
- Total possible divisions $= {}^{100}C_{50} > 10^{29}$

Modularity

How to find the division which maximizes the modularity?

- Check the value of Q for all possible divisions and choose the best one
- Consider, $N = 100$, $n_1 = n_2 = 50$
- Total possible divisions $= {}^{100}C_{50} > 10^{29}$
- With a fast computer which checks 100 billion divisions per second: 3×10^{10} years!

Modularity

How to find the division which maximizes the modularity?

- Check the value of Q for all possible divisions and choose the best one
- Consider, $N = 100$, $n_1 = n_2 = 50$
- Total possible divisions $= {}^{100}C_{50} > 10^{29}$
- With a fast computer which checks 100 billion divisions per second: 3×10^{10} years!
- Clever heuristics are required

Kernighan-Lin-Newman algorithm

Kernighan-Lin-Newman algorithm

- Start with a random division of the nodes

Kernighan-Lin-Newman algorithm

- Start with a random division of the nodes
- Change in modularity for shifting each vertex to the other group

Kernighan-Lin-Newman algorithm

- Start with a random division of the nodes
- Change in modularity for shifting each vertex to the other group
- Choose vertex whose shift makes maximum modularity change

Kernighan-Lin-Newman algorithm

- Start with a random division of the nodes
- Change in modularity for shifting each vertex to the other group
- Choose vertex whose shift makes maximum modularity change
- If no such vertex exists, choose the one resulting in the least decrease in the modularity

Kernighan-Lin-Newman algorithm

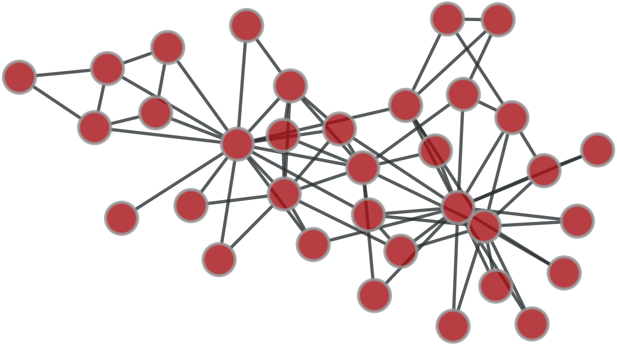
- Start with a random division of the nodes
- Change in modularity for shifting each vertex to the other group
- Choose vertex whose shift makes maximum modularity change
- If no such vertex exists, choose the one resulting in the least decrease in the modularity
- Repeat so that the vertex once moved is not moved again

Kernighan-Lin-Newman algorithm

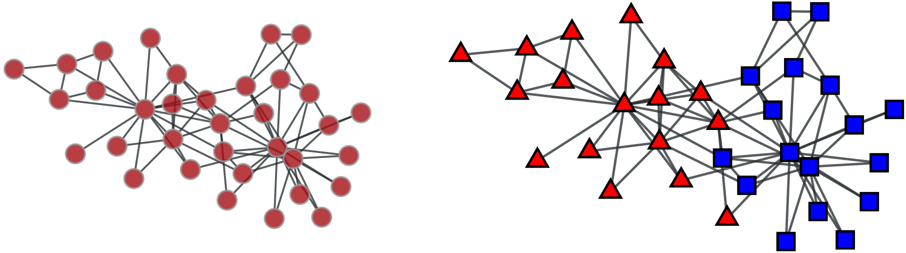
- Start with a random division of the nodes
- Change in modularity for shifting each vertex to the other group
- Choose vertex whose shift makes maximum modularity change
- If no such vertex exists, choose the one resulting in the least decrease in the modularity
- Repeat so that the vertex once moved is not moved again
- Select a state with the highest modularity

Kernighan-Lin-Newman algorithm

- Start with a random division of the nodes
- Change in modularity for shifting each vertex to the other group
- Choose vertex whose shift makes maximum modularity change
- If no such vertex exists, choose the one resulting in the least decrease in the modularity
- Repeat so that the vertex once moved is not moved again
- Select a state with the highest modularity
- Repeat the whole process starting with this state till the modularity stabilizes

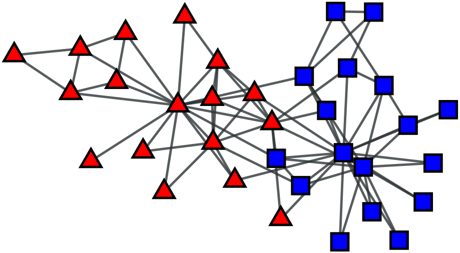


Zachry karate club network



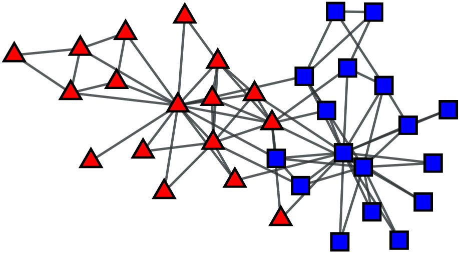
Application to Zachry karate club

Actual division

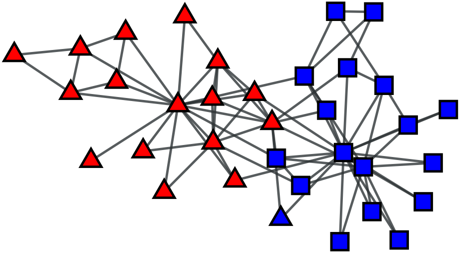


Application to Zachry karate club

Actual division



Division by KLN algorithm



Spectral modularity maximization

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) = \frac{1}{2m} \sum_{ij} B_{ij} \delta(c_i, c_j)$$

Note that:

$$\sum_j B_j = \sum_j A_{ij} - \frac{k_i}{2m} \sum_j k_j = k_i - \frac{k_i}{2m} 2m = 0$$

Spectral modularity maximization

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

Spectral modularity maximization

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

$$\frac{1}{2}(1 + s_i s_j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ belong to the same group} \\ 0 & \text{Otherwise} \end{cases}$$

Spectral modularity maximization

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

$$\frac{1}{2}(1 + s_i s_j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ belong to the same group} \\ 0 & \text{Otherwise} \end{cases}$$

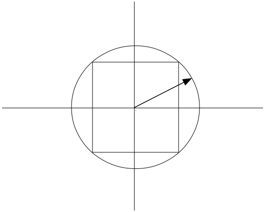
$$B = \frac{1}{2m} \sum_{ij} B_{ij} \delta(c_i, c_j) = \frac{1}{4m} \sum_{ij} B_{ij} (1 + s_i s_j) = \frac{1}{4m} \sum_{ij} B_{ij} s_i s_j$$

$$Q = \frac{1}{4m} \mathbf{s}^T \mathbf{B} \mathbf{s}$$

Spectral modularity maximization

Relaxation method

- Numbers of elements with values $+1$ and -1 are not fixed
- Only constraint: $\mathbf{s}^T\mathbf{s} = \sum_i s_i^2 = n$



$$\frac{\partial}{\partial s_i}\left[\sum_{ij} B_{jk} s_j s_k + \beta\left(n - \sum_j s_j^2\right)\right] = 0$$

$$\sum_j B_{ij} s_j = \beta s_i$$

$$\mathbf{B}\mathbf{s} = \beta\mathbf{s}$$

Spectral modularity maximization

$$Q = \frac{1}{4m}\beta \mathbf{s}^T \mathbf{B}\mathbf{s} = \frac{1}{4m}\beta \mathbf{s}^T \mathbf{s} = \frac{n}{4m}\beta$$

Thus, choose $\mathbf{s}$ to be the eigenvector $\mathbf{u}_1$ corresponding to the largest eigenvalue of the modularity matrix

Maximize:

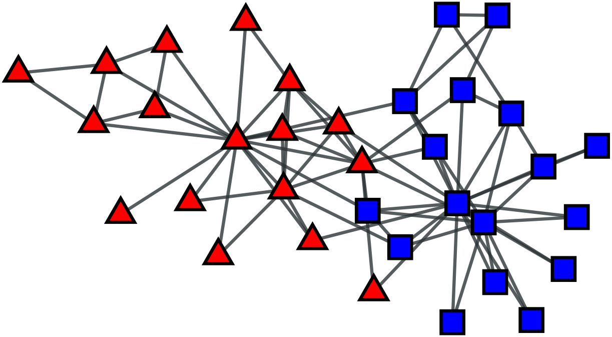
$$\mathbf{s}^T \mathbf{u}_1 = \sum_i s_i u_{1i}$$

Maximum is achieved when each term is non-negative ⇒ Use signs of u_{1i}!

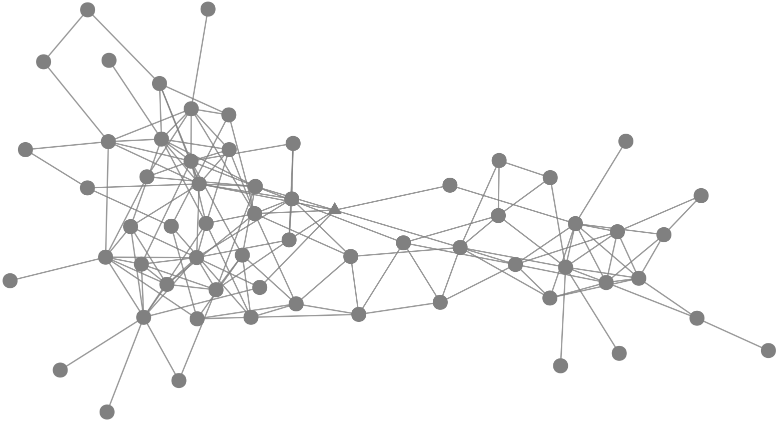
Spectral modularity maximization

- Calculate the modularity matrix
- Calculate its eigenvector corresponding to the largest eigenvalue
- Assign nodes to communities based on the signs of elements

Application to karate club network



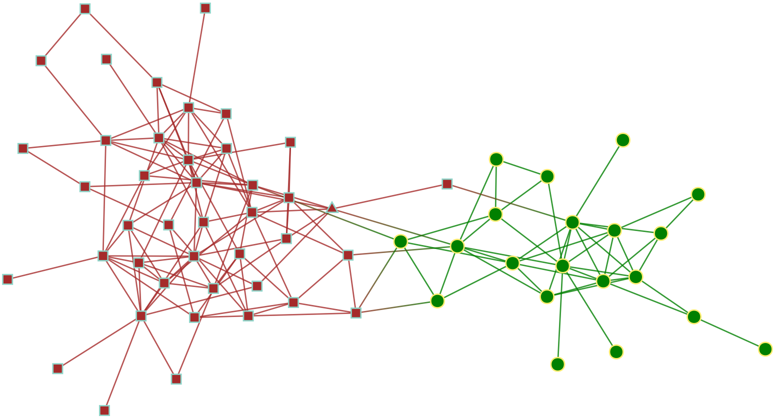
Bottlenose dolphins



[1]

[1] Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM (2003) Behav Ecol Sociobiol 54:396405

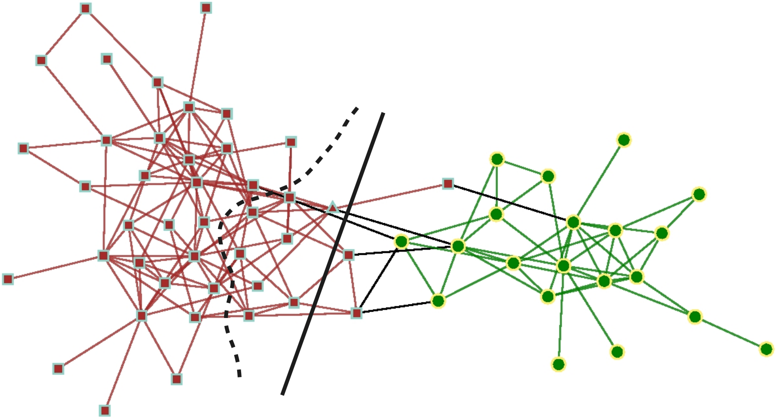
Bottlenose dolphins



2

[2] Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM (2003) Behav Ecol Sociobiol 54:396405

Bottlenose dolphins



[3]

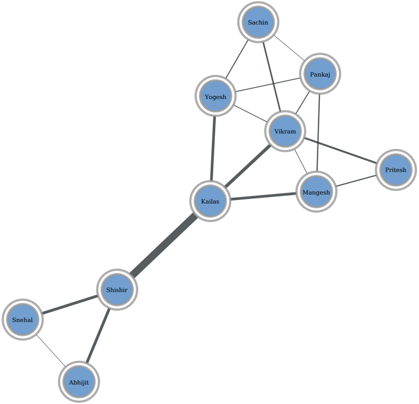
[3] Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM (2003) Behav Ecol Sociobiol 54:396405

Newman-Girvan algorithm

- Look for edges between the communities
- Edge betweenness

Edge betweenness

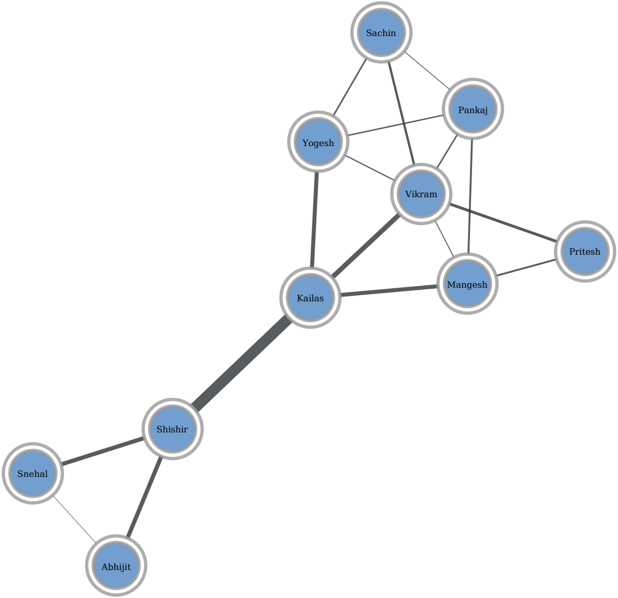
- Path between two nodes
- Shortest path between two nodes
- Number of shortest paths that go through a given edge



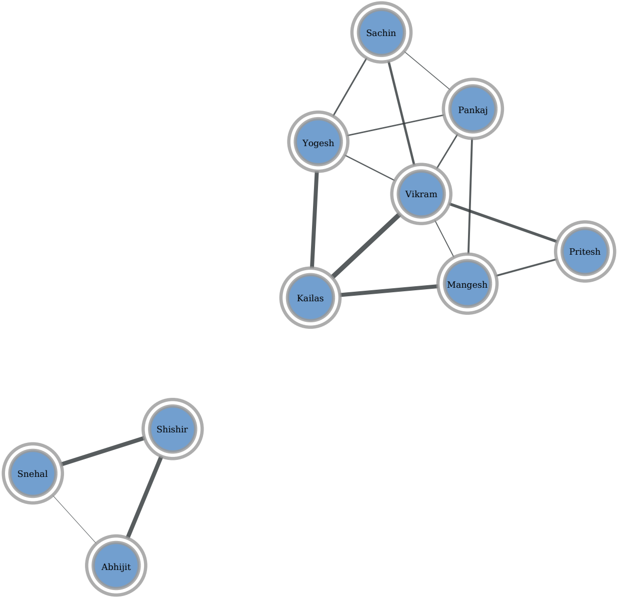
The algorithm

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

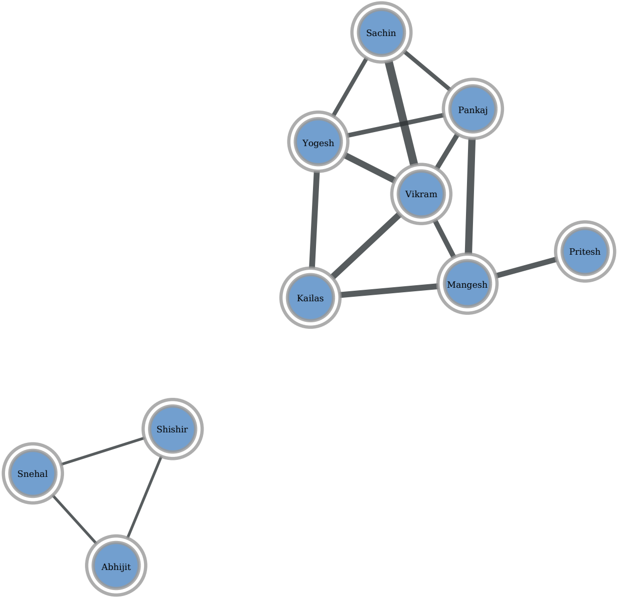
Girvan-Newman algorithm



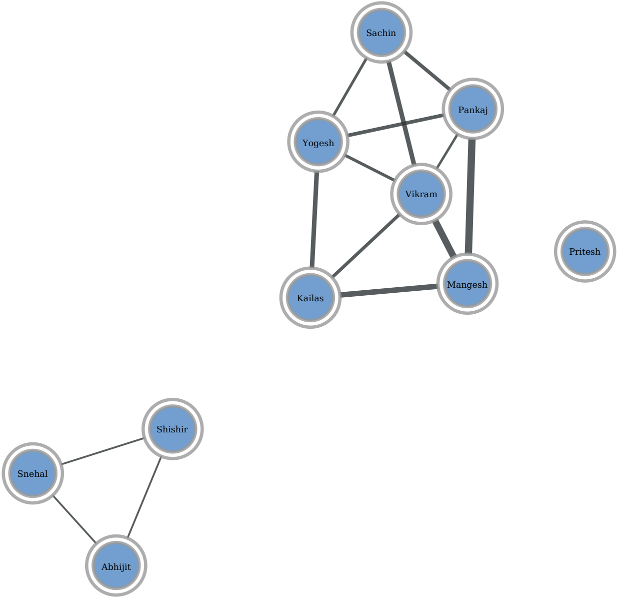
Girvan-Newman algorithm



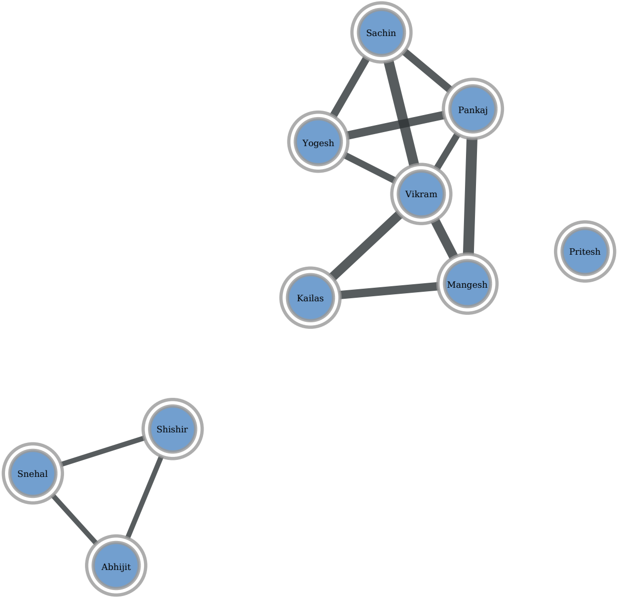
Girvan-Newman algorithm



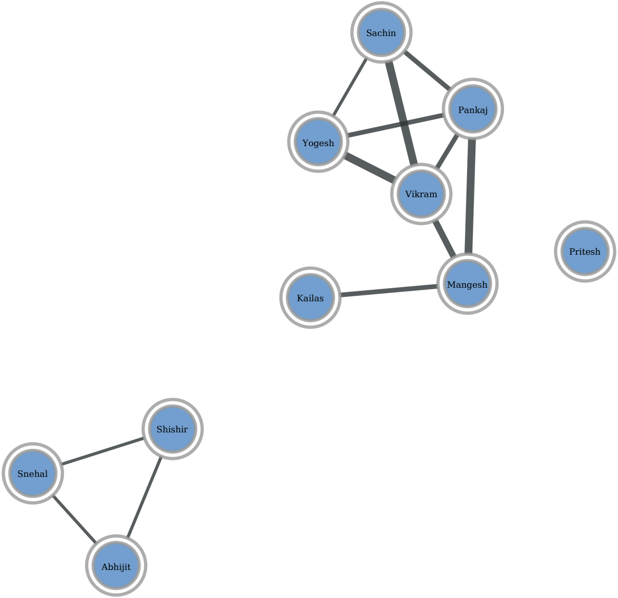
Girvan-Newman algorithm



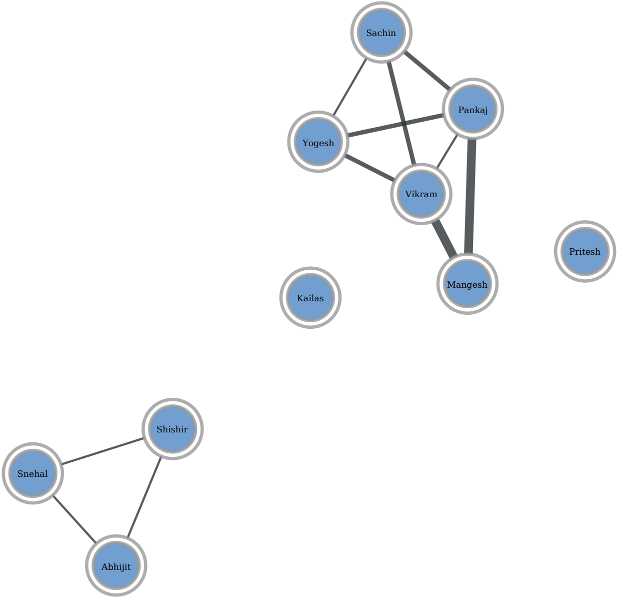
Girvan-Newman algorithm



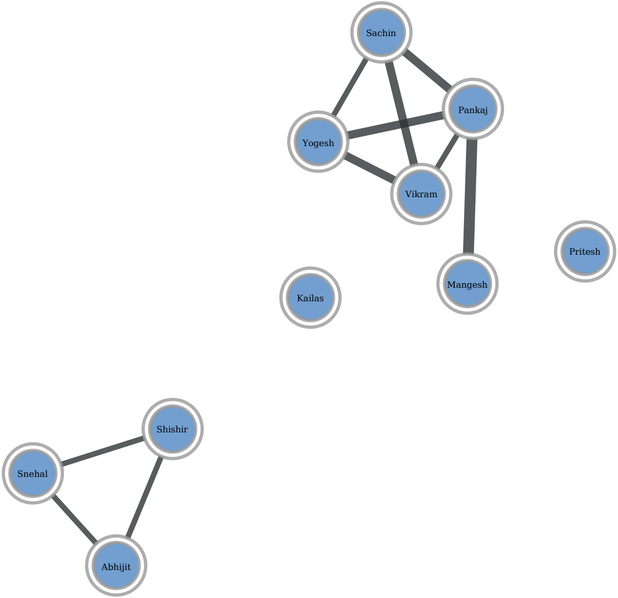
Girvan-Newman algorithm



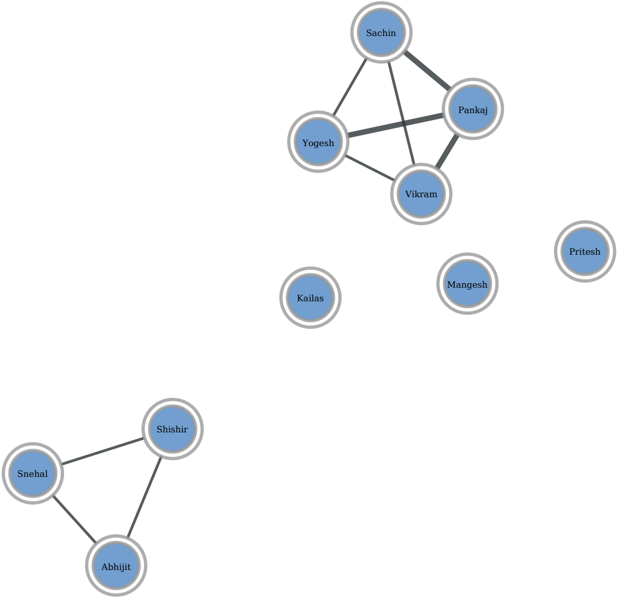
Girvan-Newman algorithm



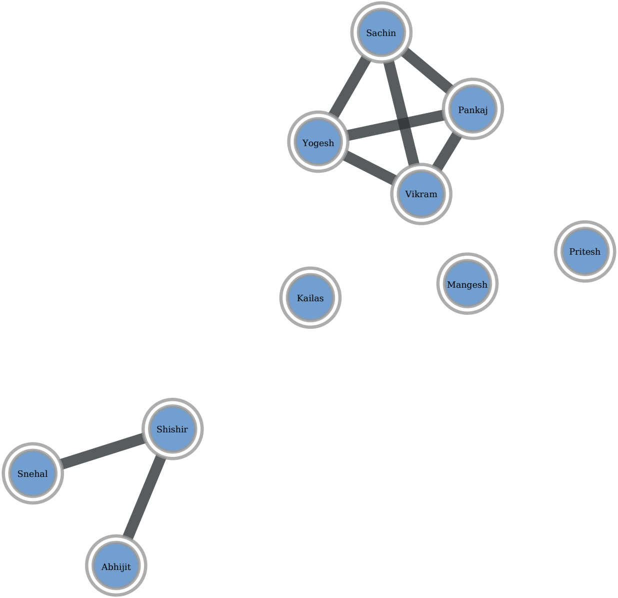
Girvan-Newman algorithm



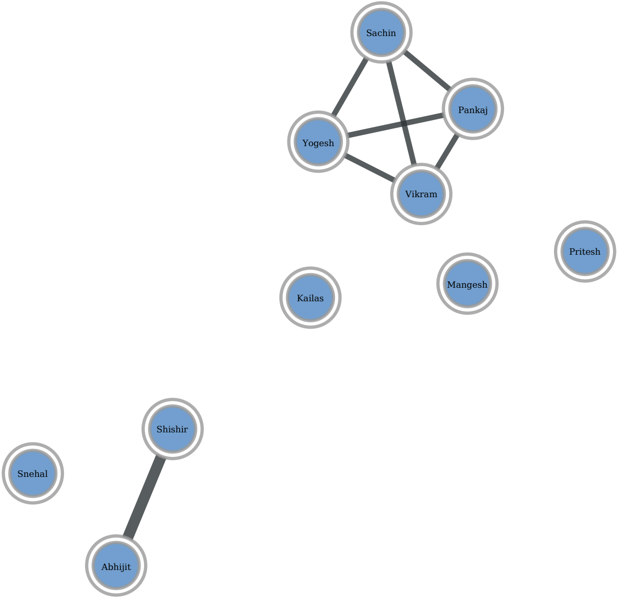
Girvan-Newman algorithm



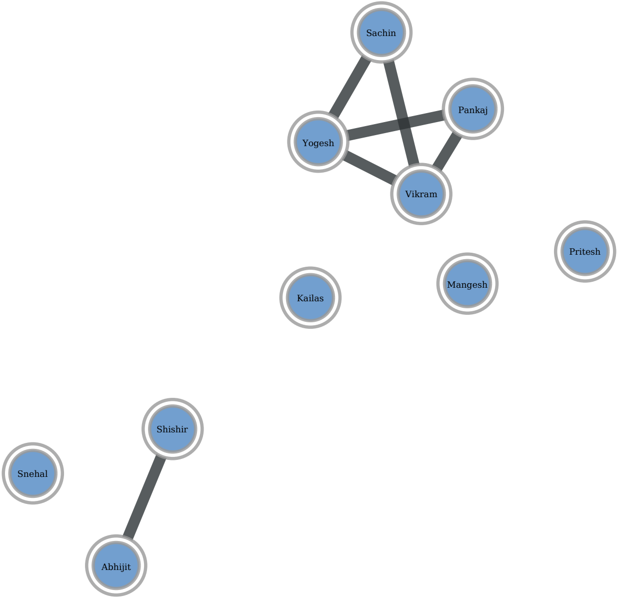
Girvan-Newman algorithm



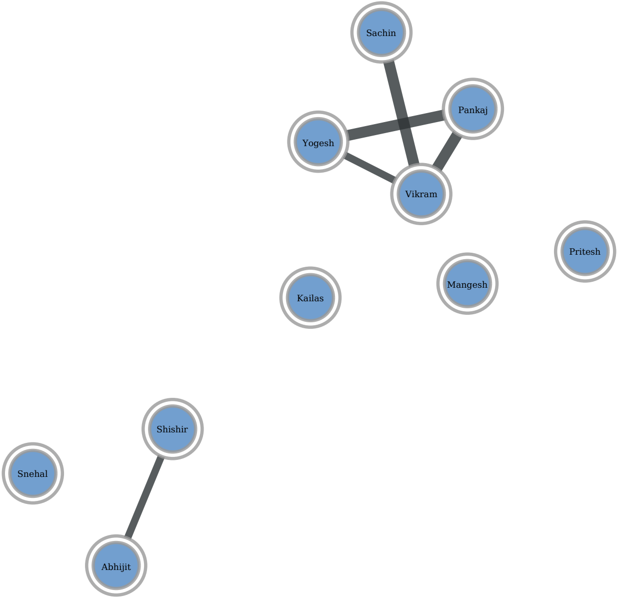
Girvan-Newman algorithm



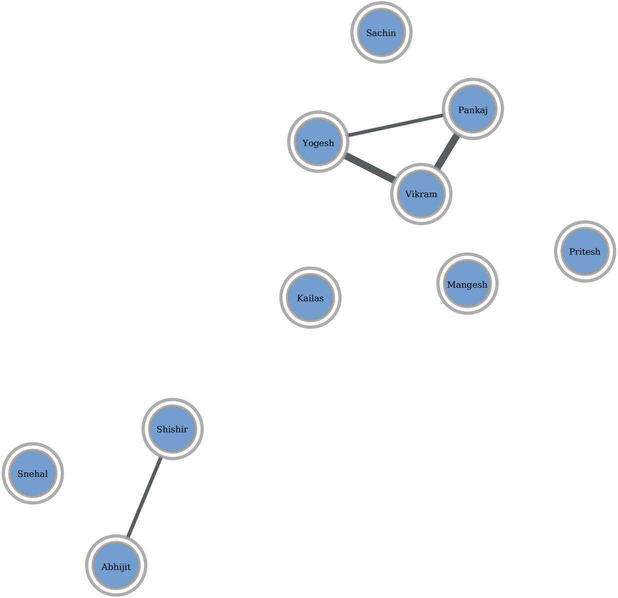
Girvan-Newman algorithm



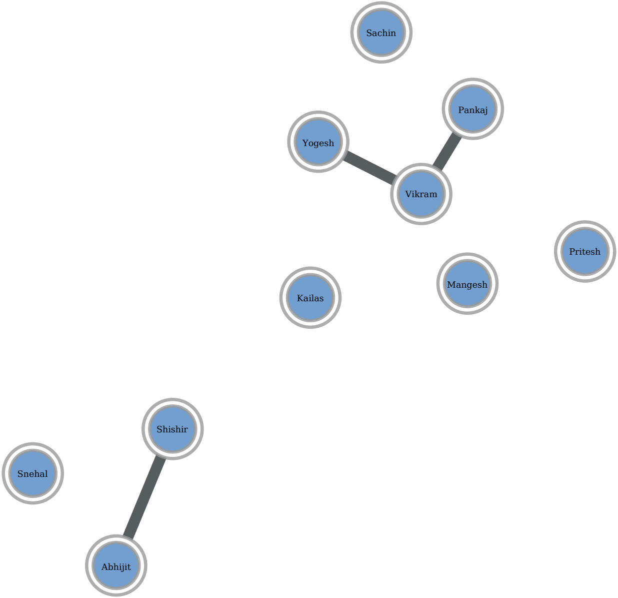
Girvan-Newman algorithm



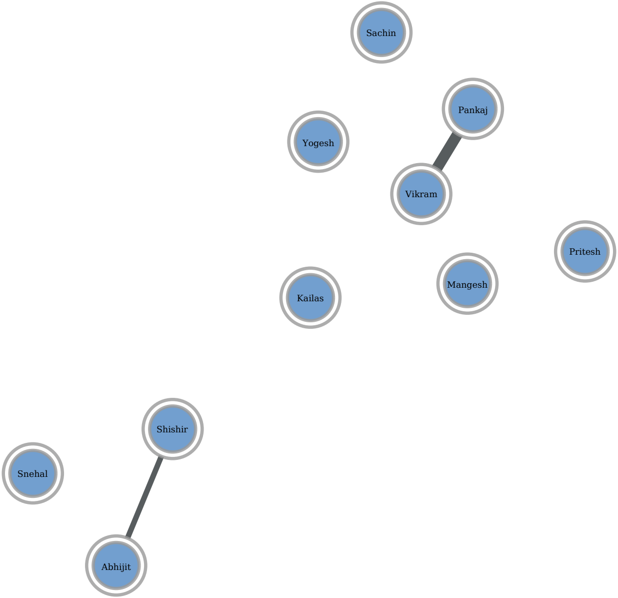
Girvan-Newman algorithm



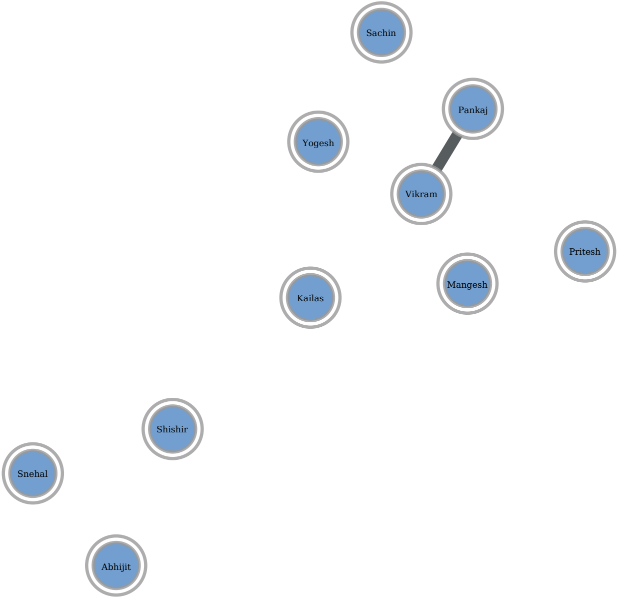
Girvan-Newman algorithm



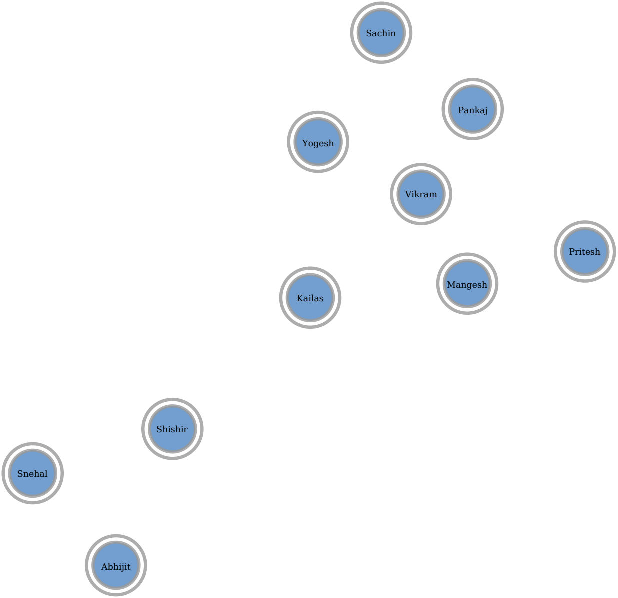
Girvan-Newman algorithm



Girvan-Newman algorithm



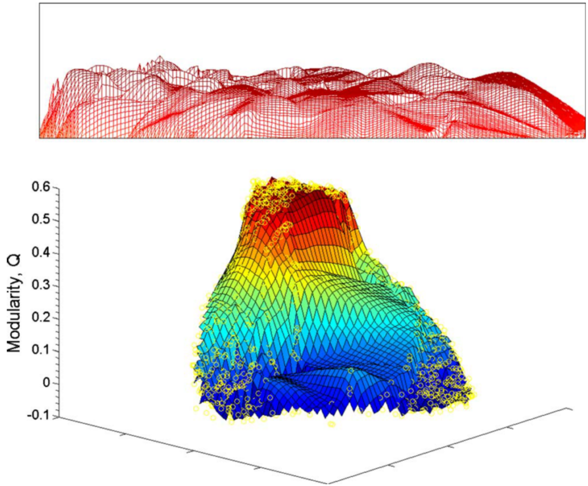
Girvan-Newman algorithm



Problems with traditional community detection algorithms

- Degeneracy
- Resolution limit
- Structure vs Noise

Degeneracy



[4]

[4] Good et al., Performance of modularity in practical contexts, PRE 81, 046106 (2010).

Resolution limit

- Maximizing the modularity can fail to resolve small sized modules
- Modular structures like cliques can be hidden in the larger groups of nodes with higher modularity score
- Peak of the modularity function may not coincide with divisions that identify such modular structures

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

Contribution of the group s,

$$Q_s = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, s)\delta(c_j, s) = \frac{e_s}{m} - \left(\frac{d_s}{2m} \right)^2$$

Resolution limit

The group s is a module whenever $Q_s > 0 \Rightarrow \frac{e_s}{m} > \left(\frac{d_s}{2m}\right)^2$

Consider two modules s_1 and s_2 with $e_{s_1 s_2}$ edges between them
The change in modularity if we merge these:

$$\triangle Q_{s_1 s_2} = \frac{e_{s_1 s_2}}{m} - 2\left(\frac{d_{s_1}}{2m}\right)\left(\frac{d_{s_2}}{2m}\right) > 0$$

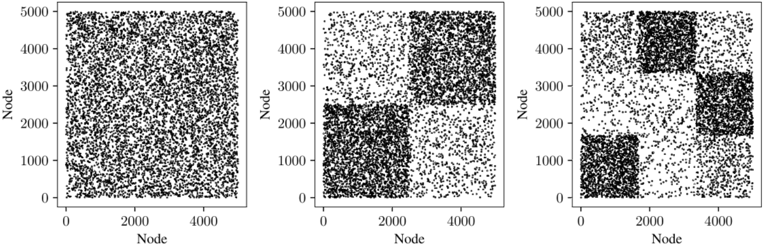
whenever:

$$e_{s_1 s_2} > \frac{d_{s_1} d_{s_2}}{2m} \rightarrow 0$$

Thus, modules would be merged even when the number of links $e_{s_1 s_2}$ between them is small! [5]

[5] Fortunato, Barthelemy, Resolution limit in community detection, PNAS, (2006)

Structure vs Noise



[6]

[6] Tiago Peixoto, Bayesian stochastic blockmodeling

Conclusions

- Community structure is a fundamental property of networks
- Community detection is an ill-defined problem
- (Too) many algorithms exist
- Community detection is still an open problem!