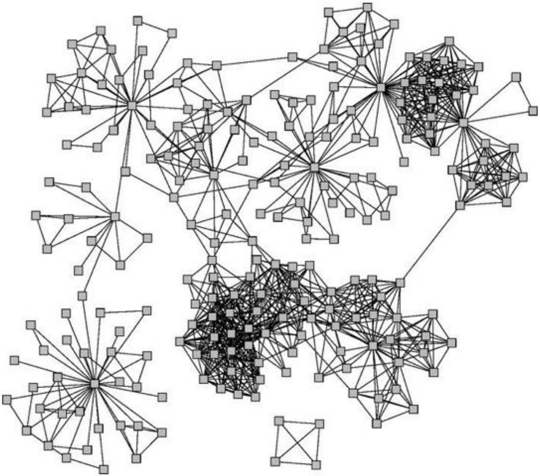
# Network structure

I. Makarov & L.E. Zhukov

**BigData Academy MADE from Mail.ru Group**

**Social Network Analysis and Machine Learning on Graphs**

# Network structure



# Typical network structure

Core-periphery structure of a network

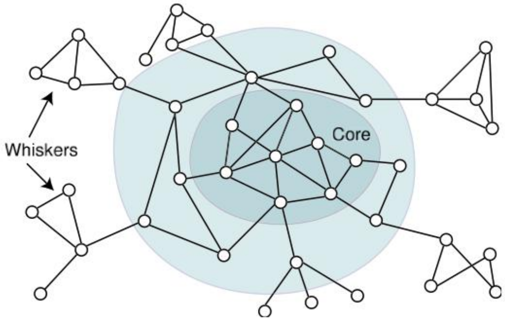
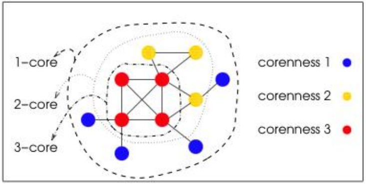


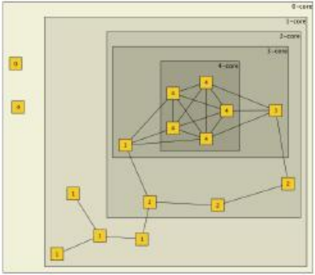

image from J. Leskovec, K. Lang, 2010

# Graph cores

**Definition**

A *k-core* is the largest subgraph such that each vertex is connected to at least $k$ others in subset



Every vertex in $k$-core has a degree $k_i \geq k$

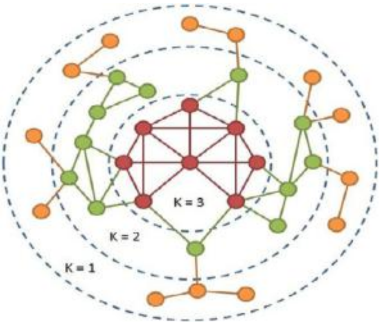
$(k+1)$-core is always subgraph of $k$-core

The core number of a vertex is the highest order of a core that contains this vertex

# k-core decomposition

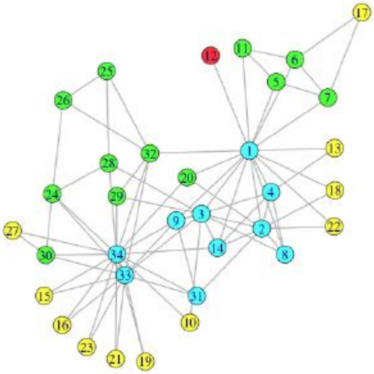
V. Batageli, M. Zaversnik, 2002

- If from a given graph G = (V, E) recursively delete all vertices, and lines incident with them, of degree less than k, the remaining graph is the k-core.

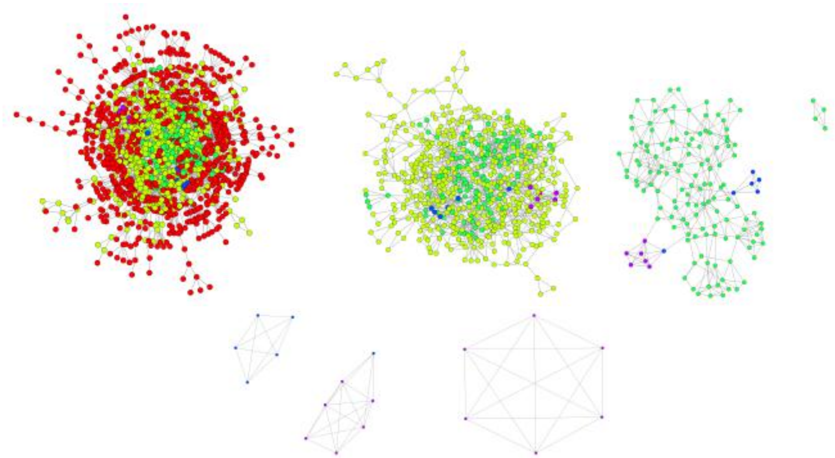


# K-cores

Zachary karate club: 1,2,3,4 - cores



# k-cores



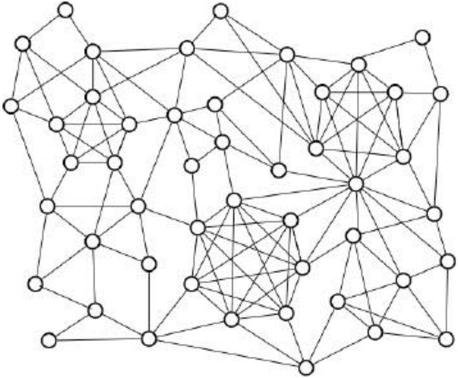
k-cores: 1:1458, 2:594, 3:142, 4:12, 5:6

k-shells: 1:864-red, 2:452-pale green, 3:130-green, 5:6-blue, 6:6-purple

# Graph cliques

## Definition

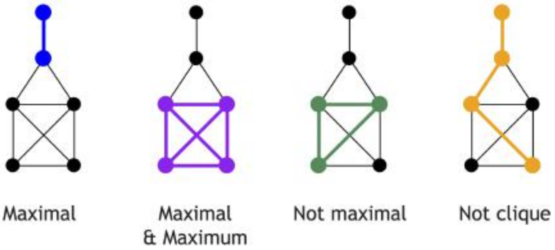
A *clique* is a complete (fully connected) subgraph, i.e. a set of vertices where each pair of vertices is connected.



Cliques can overlap

# Graph cliques

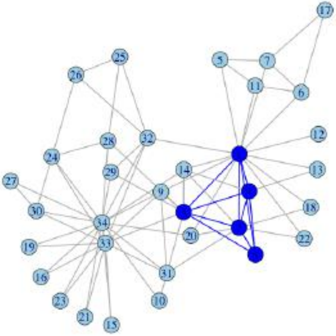
- A **maximal clique** is a clique that cannot be extended by including one more adjacent vertex (not included in larger one)
- A **maximum clique** is a clique of the largest possible size in a given graph

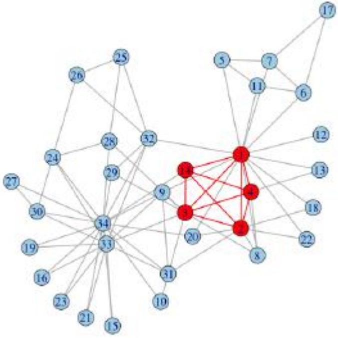



- Graph clique number is the size of the maximum clique

image from D. Eppstein

Maximum cliques




Maximal cliques:

| | | | | |
|---|---|---|---|---|
| Clique size: | 2 | 3 | 4 | 5 |
| Number of cliques: | 11 | 21 | 2 | 2 |

Zachary, 1977

# Graph cliques

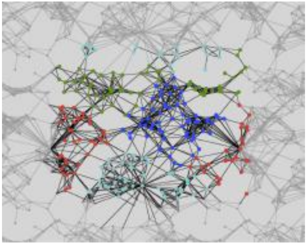
Computational issues:

- Finding click of fixed given size $k$ - $O(n^k k^2)$
- Finding maximum clique $O(3^{n/3})$
- But in sparse graphs...

# Network communities

**Definition**

*Network communities* are groups of vertices such that vertices inside the group connected with many more edges than between groups.



- Community detection is an assignment of vertices to communities.
- Will consider non-overlapping communities
- Graph partitioning problem

# Network communities

What makes a community (cohesive subgroup):

- Mutuality of ties. Almost everyone in the group has ties (edges) to one another
- Compactness. Closeness or reachability of group members in small number of steps, not necessarily adjacency
- Density of edges. High frequency of ties within the group
- Separation. Higher frequency of ties among group members compared to non-members

Wasserman and Faust

# Community density

- Graph $G(V, E)$, $n = |V|$, $m = |E|$
- Community - set of nodes $S$
  $n_s$-number of nodes in $S$, $m_s$ - number of edges in $S$
- Graph density
$$\rho = \frac{m}{n(n-1)/2}$$
- community internal density
$$\delta_{int} = \frac{m_s}{n_s(n_s-1)/2}$$
- external edges density
$$\delta_{ext} = \frac{m_{ext}}{n_s(n-n_s)}$$
- community (cluster): $\delta_{int} > \rho$, $\delta_{ext} < \rho$

# Modularity

- Compare fraction of edges within the cluster to expected fraction in random graph with identical degree sequence

$$Q = \frac{1}{4}(m_s - E(m_s))$$

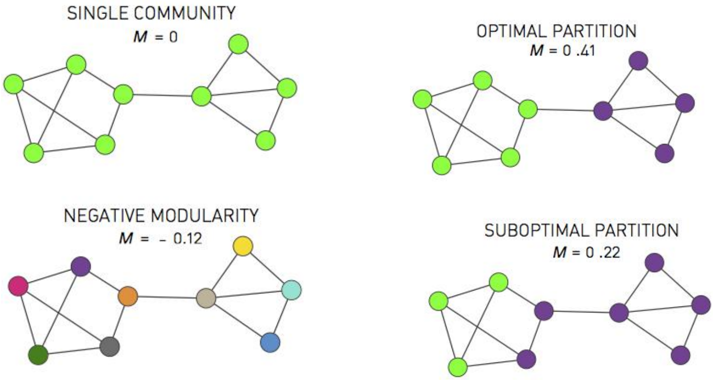
- Modularity score

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

$m_u$ - number of internal edges in a community $u$,
$k_u$ - sum of node degrees within a community

- Modularity score range $Q \in [-1/2, 1)$, single community $Q = 0$

# Modularity



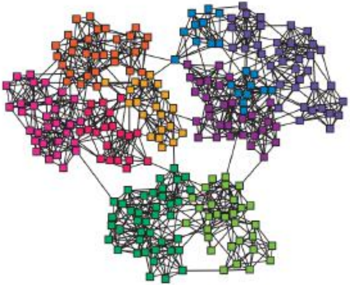

- The higher the modularity score - the better are communities

from A.L. Barabasi 2016

# Heuristic approach

Focus on edges that connect communities.
Edge betweenness -number of shortest paths $\sigma_{st}(e)$ going through edge $e$

$$C_B(e) = \sum_{s \neq t} \frac{\sigma_{st}(e)}{\sigma_{st}}$$



Construct communities by progressively removing edges

# Edge betweenness

Newman-Girvan, 2004

---

**Algorithm:** Edge Betweenness

**Input:** graph G(V,E)

**Output:** Dendrogram

**repeat**

  For all $e \in E$ compute edge betweenness $C_B(e)$;

  remove edge $e_i$ with largest $C_B(e_i)$ ;

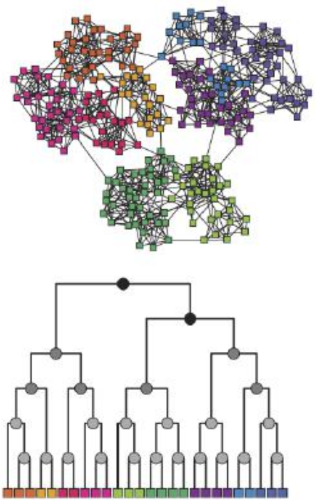
**until** *edges left*;

---

If bi-partition, then stop when graph splits in two components (check for connectedness)

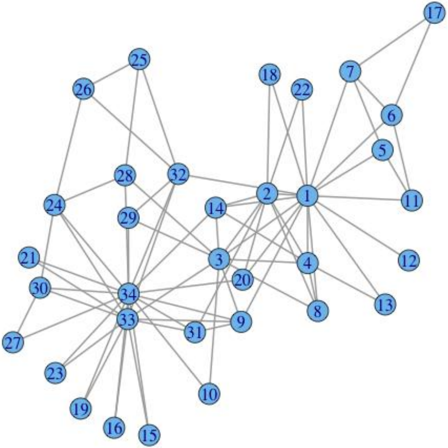
# Edge betweenness

Hierarchical algorithm, dendrogram



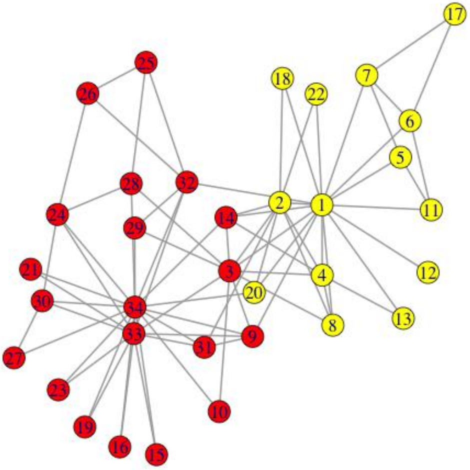
# Edge betweenness

Zachary karate club



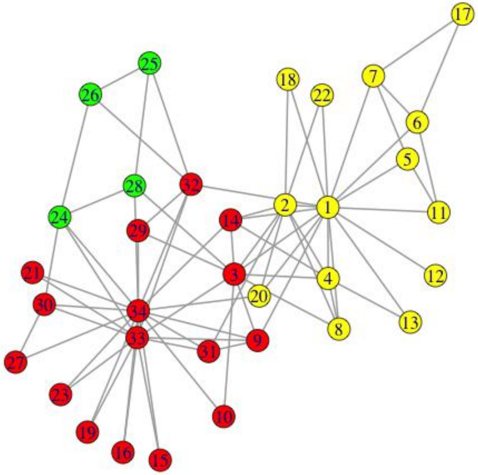
# Edge betweenness

Zachary karate club

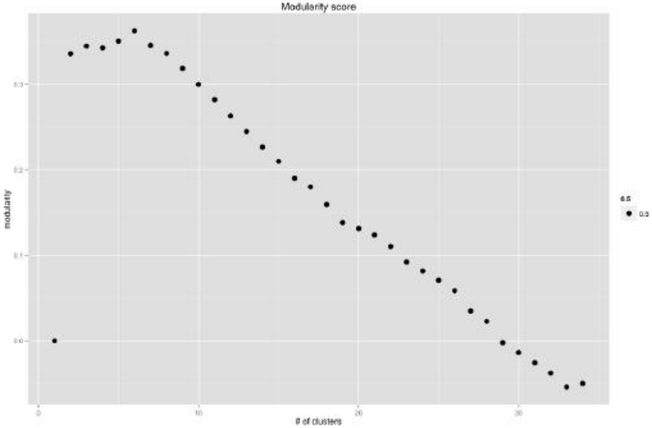


# Edge betweenness

Zachary karate club

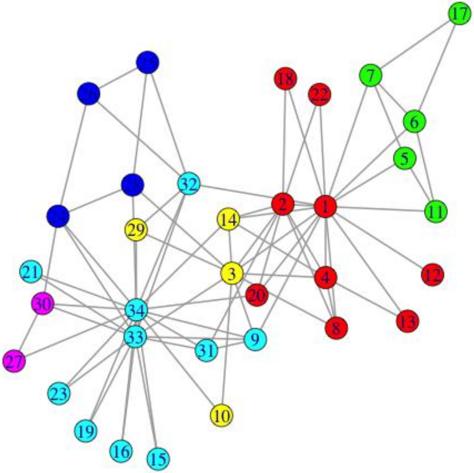


# Edge betweenness



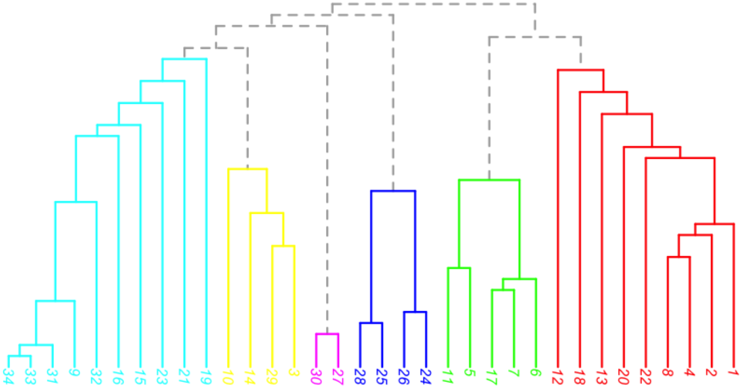
# Edge betweenness

best: clusters = 6, modularity = 0.345



# Edge betweenness

Zachary karate club



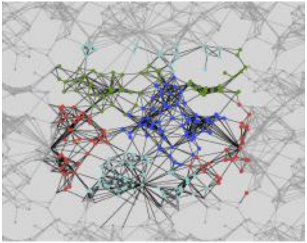
# Lecture outline

1. Network cores
2. Cliques
3. Network communities
4. Graph paritioning
5. Spectral optimization
   - Min cut
   - Normalized cut
   - Modularity maximization
6. Multilevel spectral
7. Overlapping commmunities
8. Multi-level optimization
9. Random walk methods

# Network communities

**Definition**

*Network communities* are groups of vertices such that vertices inside the group connected with many more edges than between groups.



- Graph partitioning problem

# Graph partitioning

Combinatorial problem:

- Number of ways to divide network of $n$ nodes in 2 groups (bi-partition):

$$\frac{n!}{n_1!n_2!}, \qquad n = n_1 + n_2$$

- Dividing into $k$ non-empty groups (Stirling numbers of the second kind)

$$S(n,k) = \frac{1}{k!}\sum_{j=0}^{n}(-1)^j C_k^j (k-j)^n$$

- Number of all possible partitions (n-th Bell number):

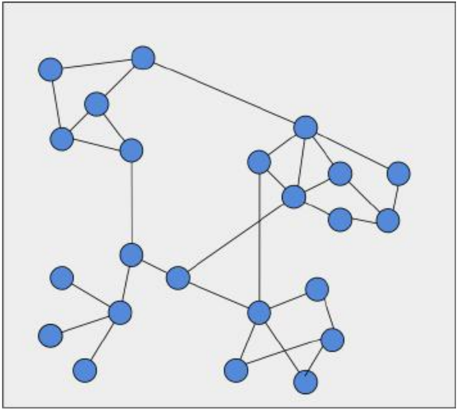
$$B_n = \sum_{k=1}^{n} S(n,k)$$

$B_{20} = 5,832,742,205,057$

# Community detection

- Consider only sparse graphs $m \ll n^2$
- Each community should be connected
- Combinatorial optimization problem:
  - optimization criterion
  - optimization method
- Exact solution NP-hard
  (bi-partition: $n = n_1 + n_2$, $n!/(n_1!n_2!)$ combinations)
- Solved by greedy, approximate algorithms or heuristics
- Recursive top-down 2-way partition, multiway partiton
- Balanced class partition vs communities

# Graph cut



# Optimization criterion: graph cut

Graph $G(E, V)$ partition: $V = V_1 + V_2$

- Graph cut

$$Q = cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} e_{ij}$$

- Ratio cut:

$$Q = \frac{cut(V_1, V_2)}{||V_1||} + \frac{cut(V_1, V_2)}{||V_2||}$$

- Normalized cut:

$$Q = \frac{cut(V_1, V_2)}{Vol(V_1)} + \frac{cut(V_1, V_2)}{Vol(V_2)}$$

- Quotient cut (conductance):

$$Q = \frac{cut(V_1, V_2)}{\min\left(Vol(V_1), Vol(V_2)\right)}$$

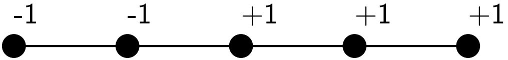
where: $Vol(V_1) = \sum_{i \in V_1, j \in V} e_{ij} = \sum_{i \in V_1} k_i$

# Optimization methods

- Greedy optimization:
  – Local search [Kernighan and Lin, 1970], [Fidducia and Mettheyses, 1982]
- Approximate optimization:
  – Spectral graph partitioning [M. Fiedler, 1972], [Pothen et al 1990], [Shi and Malik, 2000]
  – Multicommodity flow [Leighton and Rao, 1988]
- Heuristics algorithms:
  – Multilevel graph partitioning (METIS) [G. Karypis, Kumar 1998]
- Randomized algorithms:
  –Randomized min cut [D. Karger, 1993]

# Graph cuts

- Let $V = V^+ + V^-$ be partitioning of the nodes
- Let $\mathbf{s} = \{+1, -1, +1, ... -1, +1\}^T$ - indicator vector




$$s(i) = \begin{cases} +1: & \text{if} \quad v(i) \in V^+ \\ -1: & \text{if} \quad v(i) \in V^- \end{cases}$$

- Number of edges, connecting $V^+$ and $V^-$

$$cut(V^+, V^-) = \frac{1}{4} \sum_{e(i,j)} (s(i) - s(j))^2 = \frac{1}{8} \sum_{i,j} A_{ij} (s(i) - s(j))^2 =$$

$$= \frac{1}{4} \sum_{i,j} (k_i \delta_{ij} s(i)^2 - A_{ij} s(i) s(j)) = \frac{1}{4} \sum_{i,j} (k_i \delta_{ij} - A_{ij}) s(i) s(j)$$

$$cut(V^+, V^-) = \frac{1}{4} \sum_{i,j} (D_{ij} - A_{ij}) s(i) s(j)$$

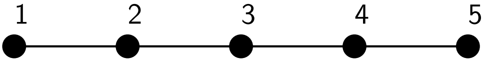
# Graph cuts

- Graph Laplacian: $\mathbf{L}_{ij} = \mathbf{D}_{ij} - \mathbf{A}_{ij}$, where $\mathbf{D}_{ii} = diag(k_i)$

$$\mathbf{L}_{ij} = \begin{cases} k(i)\,, & if\ \ i = j \\ -1\,, & if\ \exists\ e(i,j) \\ 0\,, & otherwise \end{cases}$$

- Laplacian matrix 5x5:

$$\mathbf{L} = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}$$



# Graph cuts

- Graph Laplacian: $\mathbf{L} = \mathbf{D} - \mathbf{A}$
- Graph cut:

$$Q(\mathbf{s}) = cut(V^+, V^-) = \frac{1}{4}\sum_{i,j} L_{ij} s(i) s(j) = \frac{\mathbf{s}^T \mathbf{L} \mathbf{s}}{4}$$

- Minimal cut:

$$\min_{\mathbf{s}} Q(\mathbf{s})$$

- Balanced cut constraint:

$$\sum_i s(i) = 0$$

- Integer minimization problem, exact solution is NP-hard!

# Spectral method - relaxation

- Discrete problem $\rightarrow$ continuous problem
- Discrete problem: find
$$\min_{\mathbf{s}}(\frac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s})$$
under constraints: $s(i) = \pm 1$, $\sum_i s(i) = 0$;
- Relaxation - continuous problem: find
$$\min_{\mathbf{x}}(\frac{1}{4}\mathbf{x}^T\mathbf{L}\mathbf{x})$$
under constraints: $\sum_i x(i)^2 = n$ , $\sum_i x(i) = 0$
- Given $x(i)$, round them up by $s(i) = sign(x(i))$
- Exact constraint satisfies relaxed equation, but not other way around!

# Spectral method - computations

- Constraint optimization problem (Lagrange multipliers):

$$Q(\mathbf{x}) = \frac{1}{4}\mathbf{x}^T\mathbf{L}\mathbf{x} - \lambda(\mathbf{x}^T\mathbf{x} - n), \quad \mathbf{x}^T\mathbf{e} = 0$$

- Eigenvalue problem:

$$\mathbf{L}\mathbf{x} = \lambda\mathbf{x}, \quad \mathbf{x} \perp \mathbf{e}$$

- Solution:

$$Q(\mathbf{x_i}) = \frac{n}{4}\lambda_i$$

- First (smallest) eigenvector:

$$\mathbf{L}\mathbf{e} = 0, \quad \lambda = 0, \quad \mathbf{x_1} = \mathbf{e}$$

- Looking for the second smallest eigenvalue/eigenvector $\lambda_2$ and $\mathbf{x}_2$
- Minimization of Rayleigh-Ritz quotient:

$$\min_{\mathbf{x} \perp \mathbf{x}_1}\left(\frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}\right)$$

# Spectral graph theory

- $\lambda_1 = 0$
- Number of $\lambda_i = 0$ equal to the number of connected components

- $0 \leq \lambda_2 \leq 2$
  $\lambda_2 = 0$, disconnected graph
  $\lambda_2 = 1$, totally connected

- Graph diameter (longest shortest path)

$$D(G) >= \frac{4}{n\lambda_2}$$

# Spectral graph partitioning algorithm

**Algorithm:** Spectral graph partitioning - normalized cuts

**Input:** adjacency matrix $\mathbf{A}$

**Output:** class indicator vector $\mathbf{s}$

compute $\mathbf{D} = diag(deg(\mathbf{A}))$;

compute $\mathbf{L} = \mathbf{D} - \mathbf{A}$;

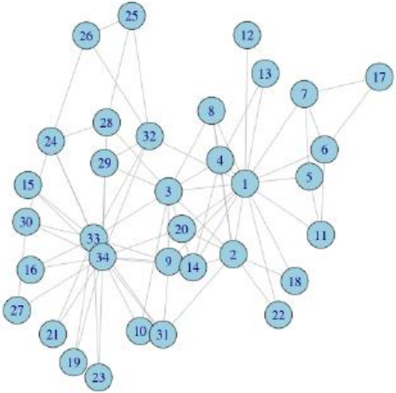
solve for second smallest eigenvector:

min cut: $\mathbf{Lx} = \lambda\mathbf{x}$;

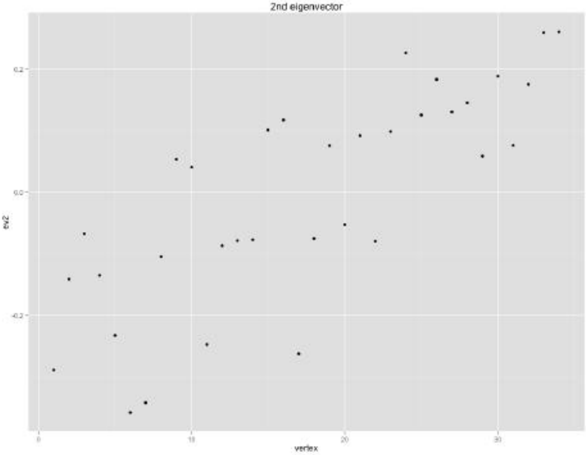
normalized cut : $\mathbf{Lx} = \lambda\mathbf{Dx}$;

set $\mathbf{s} = sign(\mathbf{x}_2)$

# Example

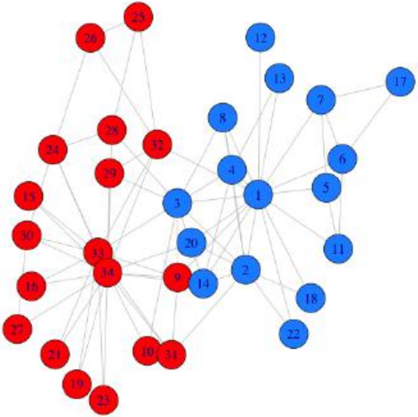

25
26
12
13
17
7
8
28
32
24
6
29
4
15
1
5
3
30
20
11
33
2
34
9
14
16
18
27
22
21
10
31
19
23

# Example



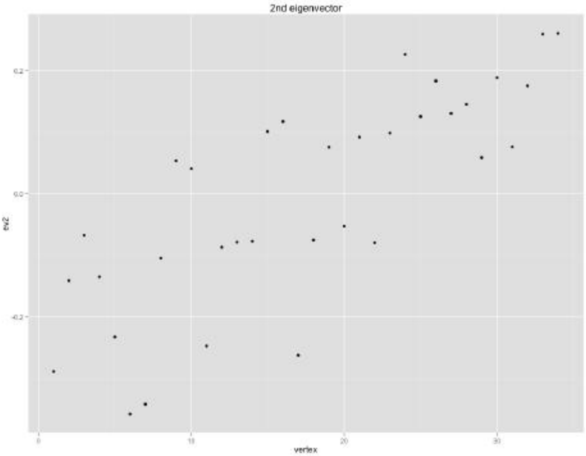

Eigenvalues: $\lambda_1 = 0,\ \lambda_2 = 0.2,\ \lambda_3 = 0.25\ \ldots$

# Example

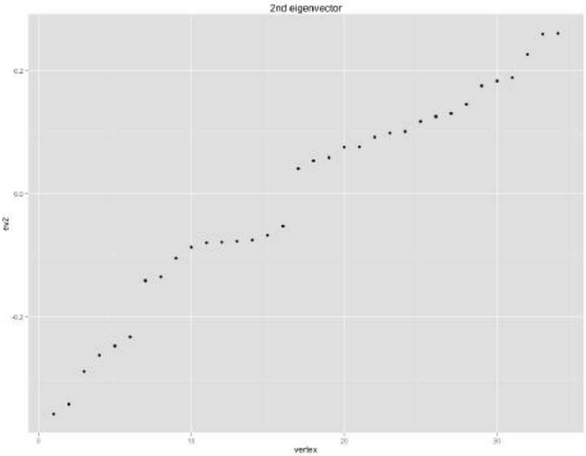


25
26
12
13
17
7
8
28
32
24
4
6
29
5
15
3
1
30
20
11
33
2
34
9
14
16
18
27
22
21
10
31
19
23

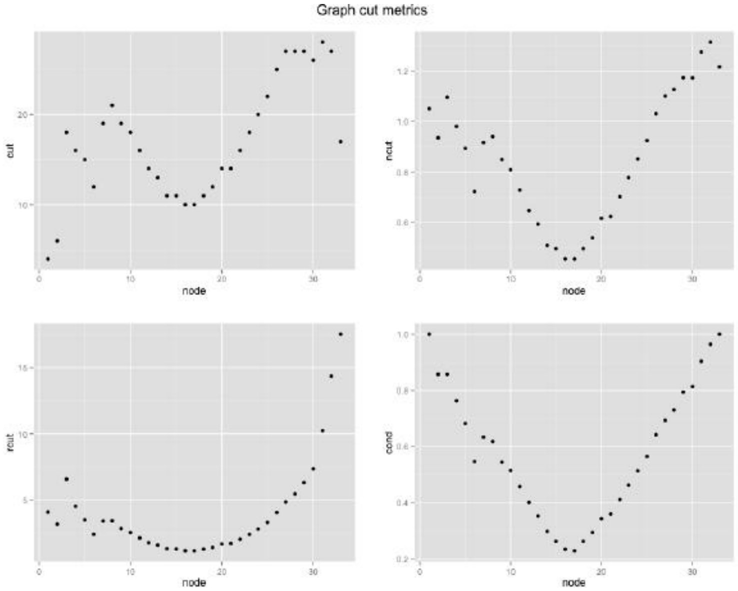
# Example



# Spectral ordering



# Cut metrics



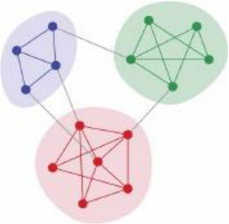
# Optimization criterion: modularity

- Modularity:

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

where $n_c$ - number of classes and

$$\delta(c_i, c_j) = \begin{cases} 1: & \text{if} \quad c_i = c_j \\ 0: & \text{if} \quad c_i \neq c_j \end{cases}$$ - kronecker delta



[Maximization!]

# Spectral modularity maximization

- Direct modularity maximization for bi-partitioning, [Newman, 2006]
- Let two classes $c_1 = V^+$, $c_2 = V^-$, indicator variable $s = \pm 1$

$$\delta(c_i, c_j) = \frac{1}{2}(s_i s_j + 1)$$

- Modularity

$$Q = \frac{1}{4m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) (s_i s_j + 1) = \frac{1}{4m} \sum_{i,j} B_{ij} s_i s_j$$

where

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

M. Newman, 2006

# Spectral modularity maximization

- Qudratic form:

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

- Integer optimization - NP, relaxation $s \to x$, $x \in R$
- Keep norm $||x||^2 = \sum_i x_i^2 = \mathbf{x}^T\mathbf{x} = n$
- Quadratic optimization

$$Q(\mathbf{x}) = \frac{1}{4m}\mathbf{x}^T\mathbf{B}\mathbf{x} - \lambda(\mathbf{x}^T\mathbf{x} - n)$$

- Eigenvector problem

$$\mathbf{B}\mathbf{x}_i = \lambda_i\mathbf{x}_i$$

- Approximate modularity

$$Q(\mathbf{x_i}) = \frac{n}{4m}\lambda_i$$

- Modularity maximization - largest $\lambda = \lambda_{max}$

# Modularity maximization

---

**Algorithm:** Spectral modularity maximization: two-way partition

**Input:** adjacency matrix $\mathbf{A}$

**Output:** class indicator vector $\mathbf{s}$

compute $\mathbf{k} = deg(\mathbf{A})$;

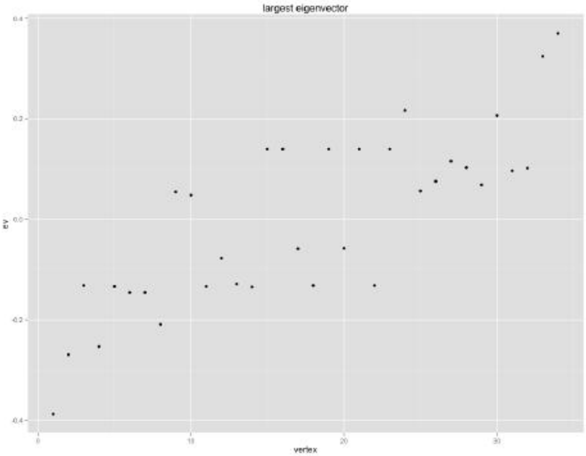
compute $\mathbf{B} = \mathbf{A} - \frac{1}{2m}\mathbf{k}\mathbf{k}^T$;

solve for maximal eigenvector $\mathbf{Bx} = \lambda \mathbf{x}$;

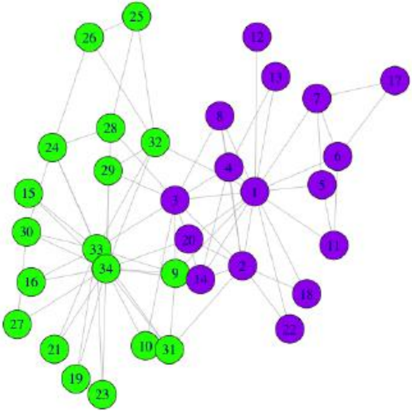
set $\mathbf{s} = sign(\mathbf{x}_{max})$

---

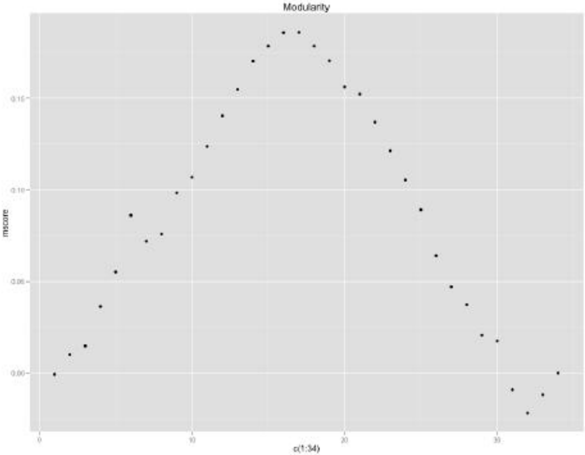
# Example



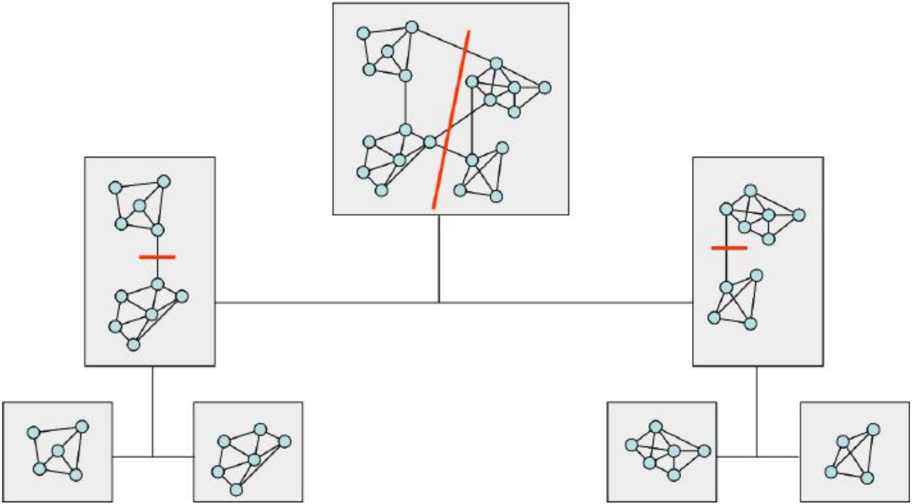
# Example


25
26
12
13
17
7
8
28
32
24
6
4
29
5
1
15
3
30
20
11
33
2
34
9
14
16
18
27
22
21
10
31
19
23

# Example

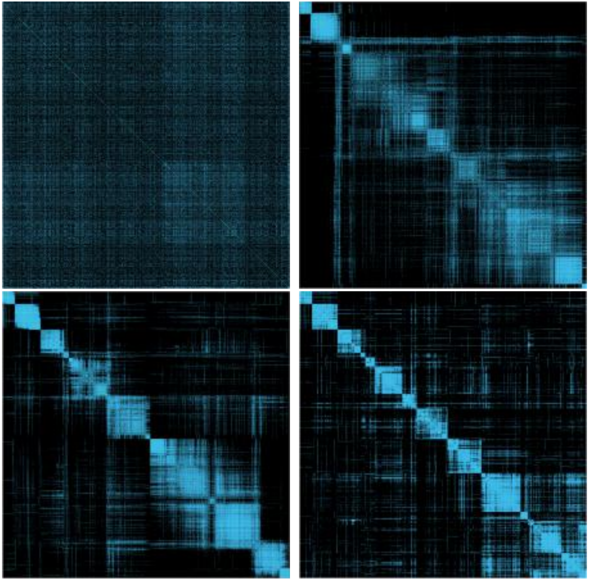


# Multilevel spectral



recursive partitioning

# Multilevel spectral



# Lecture outline

1. Network cores
2. Cliques
3. Network communities
4. Graph paritioning
5. Spectral optimization
   - Min cut
   - Normalized cut
   - Modularity maximization
6. Multilevel spectral
7. Overlapping commmunities
8. Multi-level optimization
9. Random walk methods

# Community detection

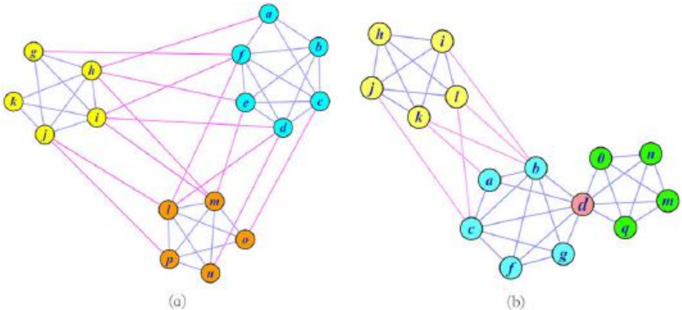
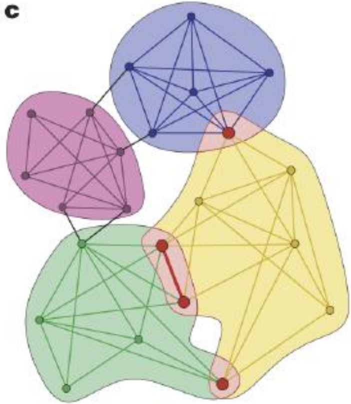


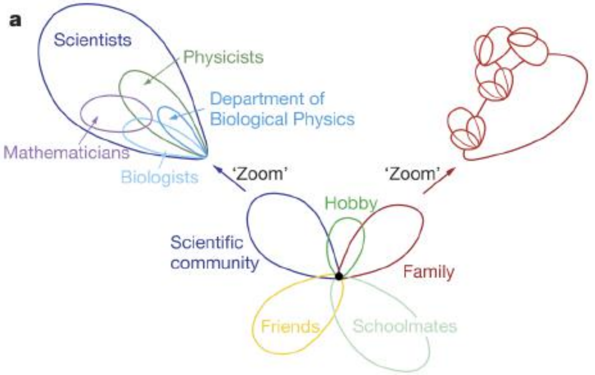

image from W. Liu , 2014

# Overlapping communities




Palla, 2005

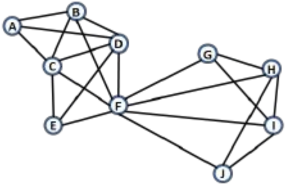
# Overlapping communities

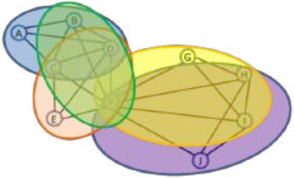



Palla, 2005

# $k$-clique community

- $k$-clique is a clique (complete subgraph) with $k$ nodes
- $k$-clique community a union of all $k$-cliques that can be reached from each other through a series of adjacent $k$-cliques
- two k-cliques are said to be adjacent if they share $k-1$ nodes.



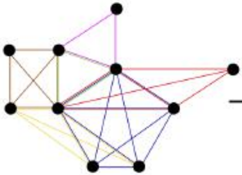
Adjacent 4-cliques

# k-clique percolation

- Find all maximal cliques
- Create clique overlap matrix
- Threshold matrix at value $k - 1$
- Communities = connected components

Palla, 2005

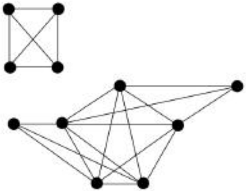
# $k$-clique percolation



| | | | | | | |
|---|---|---|---|---|---|---|
| | 5 | 3 | 2 | 1 | 3 | 1 |
| | 3 | 4 | 2 | 1 | 1 | 1 |
| | 2 | 2 | 3 | 2 | 1 | 2 |
| | 1 | 1 | 2 | 3 | 0 | 1 |
| | 3 | 1 | 1 | 0 | 4 | 2 |
| | 1 | 1 | 2 | 1 | 2 | 4 |

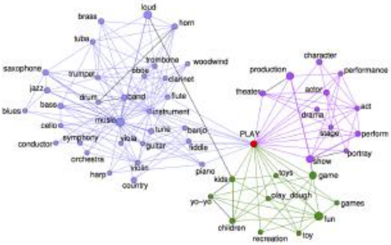
k=4

| | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 1 | 0 | 0 | 1 | 0 |
| | 1 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 1 |

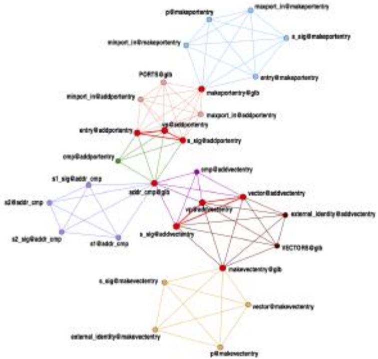


Palla, 2005

# *k*-clique percolation




$k = 4$

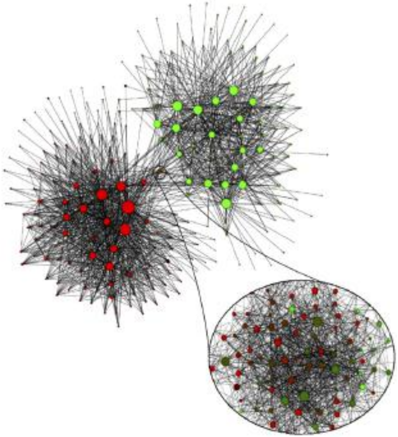


$k = 5$

Palla, 2005

# Fast community unfolding

Multi-resolution scalable method



2 mln mobile phone network
V. Blondel et.al., 2008

# Fast community unfolding

"The Louvain method"

- Heuristic method for greedy modularity optimization
- Find partitions with high modularity
- Multi-level (multi-resolution) hierarchical scheme
- Scalable

Modularity:

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

V. Blondel et.al., 2008

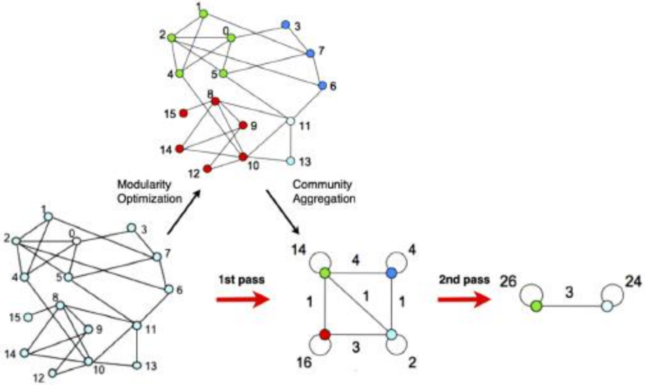
# Fast community unfolding

Algorithm

- Assign every node to its own community
- Phase I
  - For every node evaluate modularity gain from removing node from its community and placing it in the community of its neighbor
  - Place node in the community maximizing modularity gain
  - repeat until no more improvement (local max of modularity)
- Phase II
  - Nodes from communities merged into "super nodes"
  - Weight on the links added up
- Repeat until no more changes (max modularity)

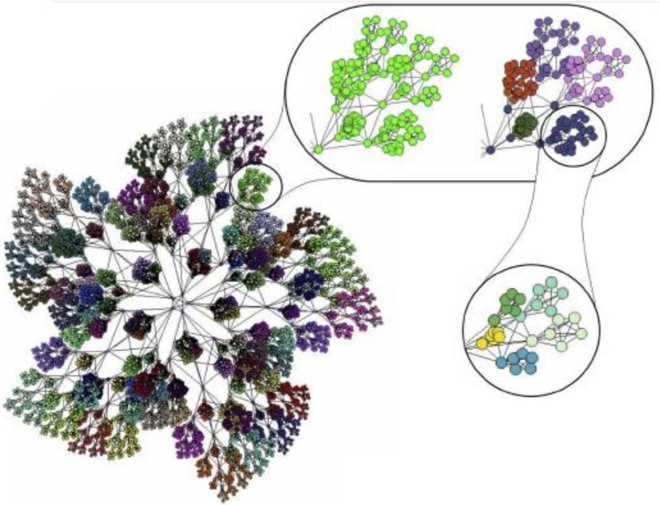
V. Blondel et.al., 2008

# Fast community unfolding



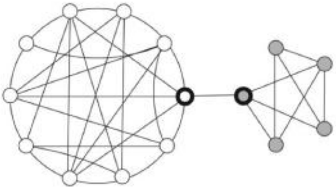

V. Blondel et.al., 2008

# Fast community unfolding



V. Blondel et.al., 2008

# Communities and random walks



- Random walks on a graph tend to get trapped into densely connected parts corresponding to communities.

Walktrap

- Consider random walk on graph
- At each time step walk moves to NN uniformly at random $P_{ij} = \frac{A_{ij}}{d(i)}$, $P = D^{-1}A$, $D_{ii} = diag(d(i))$
- $P^t_{ij}$ - probability to get from $i$ to $j$ in $t$ steps, $t \ll t_{mixing}$
- Assumptions: for two $i$ and $j$ in the same community $P^t_{ij}$ is high
- if $i$ and $j$ are in the same community, then $\forall k$, $P^t_{ik} \approx P^t_{jk}$
- Distance between nodes:

$$r_{ij}(t) = \sqrt{\sum_{k=1}^{n} \frac{(P^t_{ik} - P^t_{jk})^2}{d(k)}} = ||D^{-1/2}P^t_i - D^{-1/2}P^t_j||$$

P. Pons and M. Latapy, 2006

# Walktrap

Computing node distance $r_{ij}$

- Direct (exact) computation: $P^t_{ij} = (P^t)_{ij}$ or $P^t_i = P^t p^0_i$, $p^0_i(k) = \delta_{ik}$
- Approximate computation (simulation):
  - Compute $K$ random walks of length $t$ starting form node $i$
  - Approximate $P^t_{ik} \approx \frac{N_{ik}}{K}$, number of walks end up on $k$

Distance between communities:

$$P^t_{Cj} = \frac{1}{|C|} \sum_{i \in C} P^t_{ij}$$

$$r_{C_1 C_2}(t) = \sqrt{\sum_{k=1}^{n} \frac{(P^t_{C_1 k} - P^t_{C_2 k})^2}{d(k)}} = ||D^{-1/2} P^t_{C_1} - D^{-1/2} P^t_{C_2}||$$

P. Pons and M. Latapy, 2006

# Walktrap

Algorithm (hierarchical clustering)

- Assign each vertex to its own community $S_1 = \{\{v\}, v \in V\}$
- Compute distance between all adjacent communities $r_{C_i C_j}$
- Choose two "closest" communities that minimizes (Ward's methods):

$$\Delta\sigma(C_1, C_2) = \frac{1}{n}\left(\sum_{i\in C_3} r_{iC_3}^2 - \sum_{i\in C_1} r_{iC_1}^2 - \sum_{i\in C_2} r_{iC_2}^2\right)$$

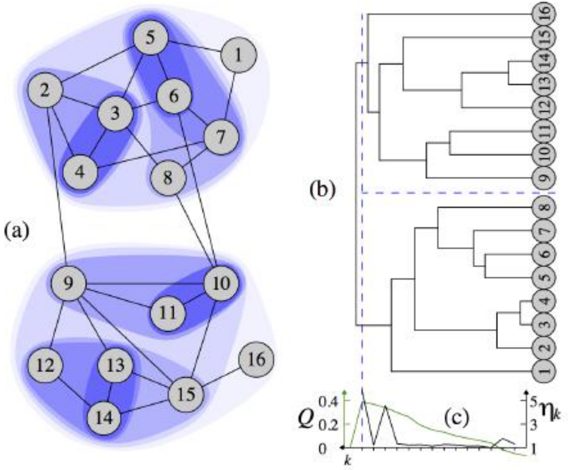
  and merge them $S_{k+1} = (S_k \setminus \{C_1, C_2\}) \cup C_3,\ C_3 = C_1 \cup C_2$

- update distance between communities

After $n-1$ steps finish with one community $S_n = \{V\}$

P. Pons and M. Latapy, 2006

# Walktrap




P. Pons and M. Latapy, 2006

# Community detection algorithms

| Author | Ref. | Label | Order |
|---|---|---|---|
| Eckmann & Moses | (Eckmann and Moses, 2002) | EM | $O(m\langle k^2\rangle)$ |
| Zhou & Lipowsky | (Zhou and Lipowsky, 2004) | ZL | $O(n^3)$ |
| Latapy & Pons | (Latapy and Pons, 2005) | LP | $O(n^3)$ |
| Clauset et al. | (Clauset *et al.*, 2004) | NF | $O(n \log^2 n)$ |
| Newman & Girvan | (Newman and Girvan, 2004) | NG | $O(nm^2)$ |
| Girvan & Newman | (Girvan and Newman, 2002) | GN | $O(n^2 m)$ |
| Guimerà et al. | (Guimerà and Amaral, 2005; Guimerà *et al.*, 2004) | SA | parameter dependent |
| Duch & Arenas | (Duch and Arenas, 2005) | DA | $O(n^2 \log n)$ |
| Fortunato et al. | (Fortunato *et al.*, 2004) | FLM | $O(m^3 n)$ |
| Radicchi et al. | (Radicchi *et al.*, 2004) | RCCLP | $O(m^4/n^2)$ |
| Donetti & Muñoz | (Donetti and Muñoz, 2004, 2005) | DM/DMN | $O(n^3)$ |
| Bagrow & Bollt | (Bagrow and Bollt, 2005) | BB | $O(n^3)$ |
| Capocci et al. | (Capocci *et al.*, 2005) | CSCC | $O(n^2)$ |
| Wu & Huberman | (Wu and Huberman, 2004) | WH | $O(n+m)$ |
| Palla et al. | (Palla *et al.*, 2005) | PK | $O(\exp(n))$ |
| Reichardt & Bornholdt | (Reichardt and Bornholdt, 2004) | RB | parameter dependent |

| Author | Ref. | Label | Order |
|---|---|---|---|
| Girvan & Newman | (Girvan and Newman, 2002; Newman and Girvan, 2004) | GN | $O(nm^2)$ |
| Clauset et al. | (Clauset *et al.*, 2004) | Clauset et al. | $O(n \log^2 n)$ |
| Blondel et al. | (Blondel *et al.*, 2008) | Blondel et al. | $O(m)$ |
| Guimerà et al. | (Guimerà and Amaral, 2005; Guimerà *et al.*, 2004) | Sim. Ann. | parameter dependent |
| Radicchi et al. | (Radicchi *et al.*, 2004) | Radicchi et al. | $O(m^4/n^2)$ |
| Palla et al. | (Palla *et al.*, 2005) | Cfinder | $O(\exp(n))$ |
| Van Dongen | (Dongen, 2000a) | MCL | $O(nk^2)$, $k < n$ parameter |
| Rosvall & Bergstrom | (Rosvall and Bergstrom, 2007) | Infomod | parameter dependent |
| Rosvall & Bergstrom | (Rosvall and Bergstrom, 2008) | Infomap | $O(m)$ |
| Donetti & Muñoz | (Donetti and Muñoz, 2004, 2005) | DM | $O(n^3)$ |
| Newman & Leicht | (Newman and Leicht, 2007) | EM | parameter dependent |
| Ronhovde & Nussinov | (Ronhovde and Nussinov, 2009) | RN | $O(m^\beta \log n)$, $\beta \sim 1.3$ |

Fortunato, 2010

# Summary

Lectures 1-5 Descriptive Network Analysis

- Network characteristics:
  – Power law node degree distribution
  – Small diameter
  – High clustering coefficient (transitivity)
- Network models:
  – Random graphs
  – Preferential attachment
  – Small world
- Centrality measures:
  – Degree centrality
  – Closeness centrality
  – Betweenness centrality
- Link analysis:
  – Page rank
  – HITS

# Summary

Lectures 1-5 Descriptive Network Analysis

- Structural equivalence
  - Vertex equivalence
  - Vertex similarity
- Assortative mixing
  - Assortative and disassortative networks
  - Mixing by node degree
  - Modularity
- Network structures:
  - Cliques
  - k-cores
- Network communities:
  - Graph partitioning
  - Overlapping communities
  - Heuristic methods
  - Random walk based methods

# References


- Finding and evaluating community structure in networks, M.E.J. Newman, M. Girvan, Phys. Rev E, 69, 2004
- Modularity and community structure in networks, M.E.J. Newman, PNAS, vol 103, no 26, pp 8577-8582, 2006
- S. E. Schaeffer. Graph clustering. Computer Science Review, 1(1):27–64, 2007.
- S. Fortunato. Community detection in graphs, Physics Reports, Vol. 486, Iss. 3–5, pp 75-174, 2010

# References


- M. Fiedler. Algebraic connectivity of graphs, Czech. Math. J, 23, pp 298-305, 1973
- A. Pothen, H. Simon and K. Liou. Partitioning sparse matrices with eigenvectors of graphs, SIAM Journal of Matrix Analysis, 11, pp 430-452, 1990
- Bruce Hendrickson and Robert Leland. A Multilevel Algorithm for Partitioning Graphs, Sandia National Laboratories, 1995
- Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, N 8, pp 888-905, 2000
- M.E.J. Newman. Modularity and community structure in networks. PNAS Vol. 103, N 23, pp 8577-8582, 2006
- B. Good, Y.-A. de Montjoye, A. Clauset. Performance of modularity maximization in practical contexts, Physical Review E 81, 046106, 2010

# References


- G. Palla, I. Derenyi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (2005) 814?818.
- P. Pons and M. Latapy, Computing communities in large networks using random walks, Journal of Graph Algorithms and Applications, 10 (2006), 191-218.
- V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. P10008 (2008).
- J. Leskovec, K.J. Lang, A. Dasgupta, and M.W. Mahoney. Statistical properties of community structure in large social and information networks. In WWW 08: Procs. of the 17th Int. Conf. on World Wide Web, pages 695-704, 2008.

# References


- M.A Porter, J-P Onella, P.J. Mucha. Communities in Networks, Notices of the American Mathematical Society, Vol. 56, No. 9, 2009
- S. E. Schaeffer. Graph clustering. Computer Science Review, 1(1), pp 27-64, 2007.
- S. Fortunato. Community detection in graphs, Physics Reports, Vol. 486, Iss. 3-5, pp 75-174, 2010