Network communities

Leonid E. Zhukov

School of Data Analysis and Artificial Intelligence
Department of Computer Science
National Research University Higher School of Economics

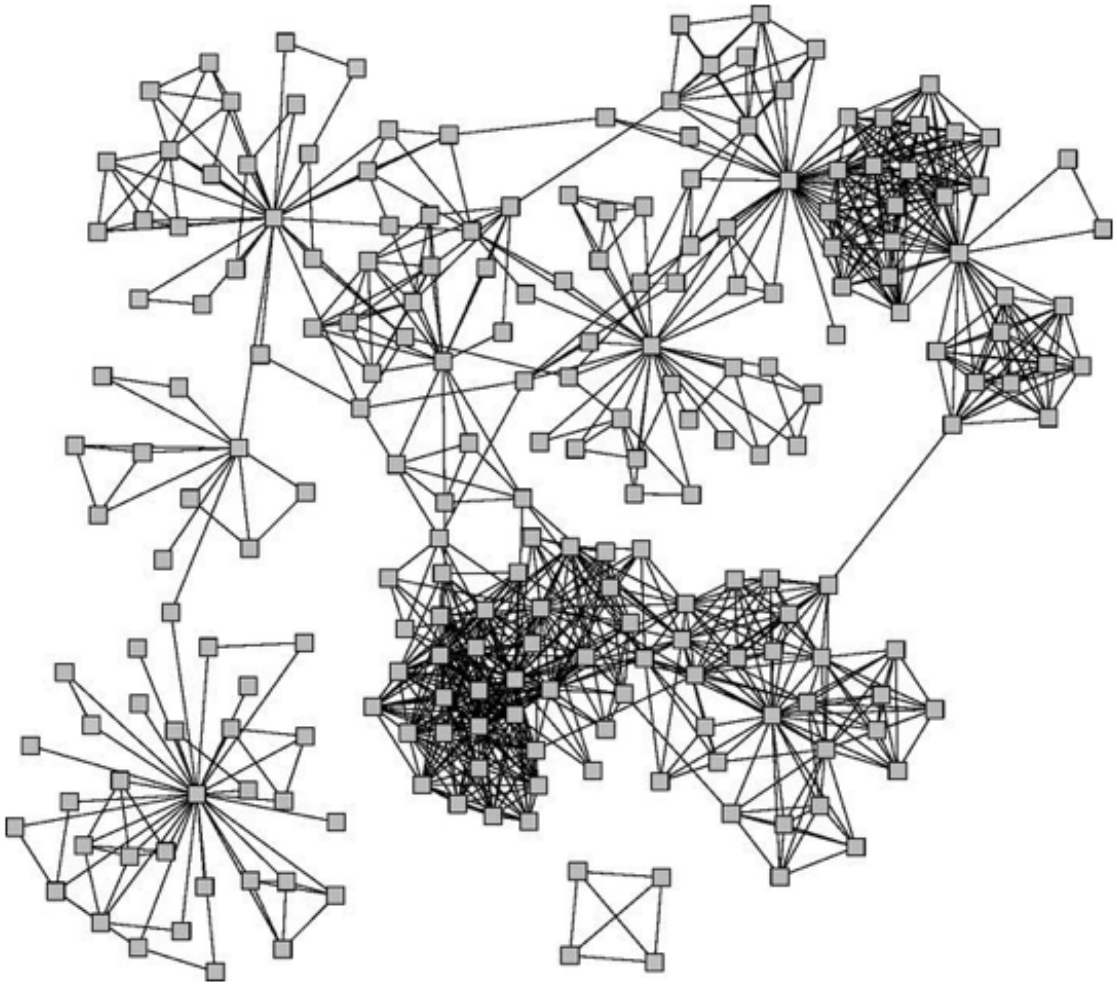
Social Network Analysis

MAGoLEGO course

Lecture outline

1. Cohesive subgroups
   - Graph cliques
2. Network communities
   - Density, cuts, modularity
3. Algorithms
   - Edge betweenness
   - Modularity maximization
   - Label propagation
   - Fast community unfolding
   - Walktrap

Network communities



Connected and undirected graphs

Network communities

What makes a community (cohesive subgroup):

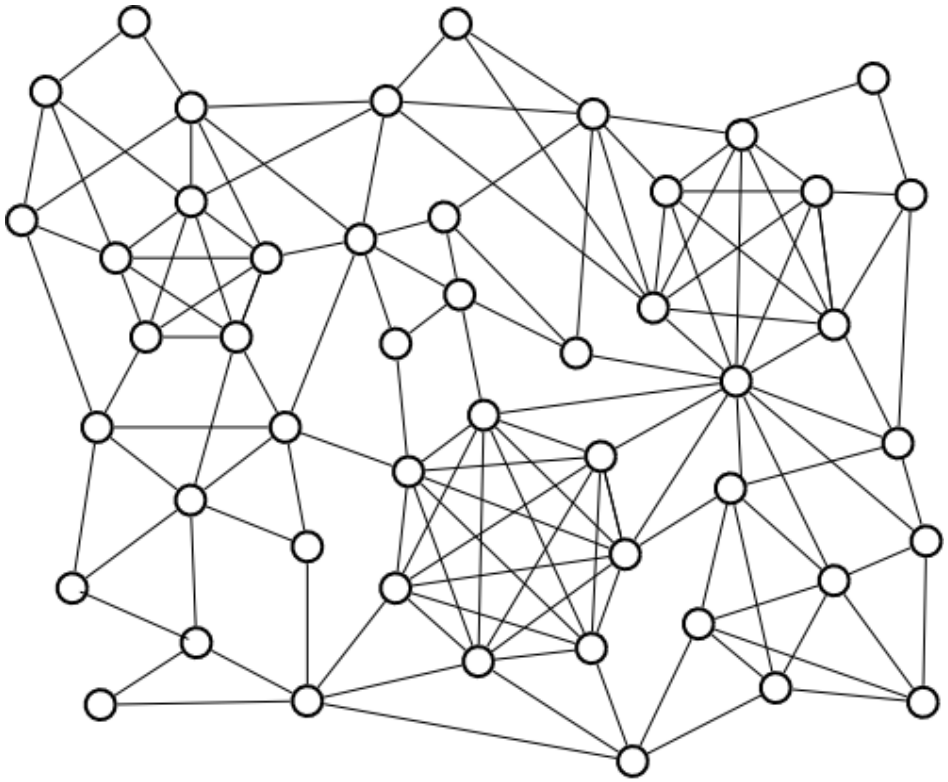
- Mutuality of ties. Everyone in the group has ties (edges) to one another
- Compactness. Closeness or reachability of group members in small number of steps, not necessarily adjacency
- Density of edges. High frequency of ties within the group
- Separation. Higher frequency of ties among group members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

Wasserman and Faust

Graph cliques

Definition

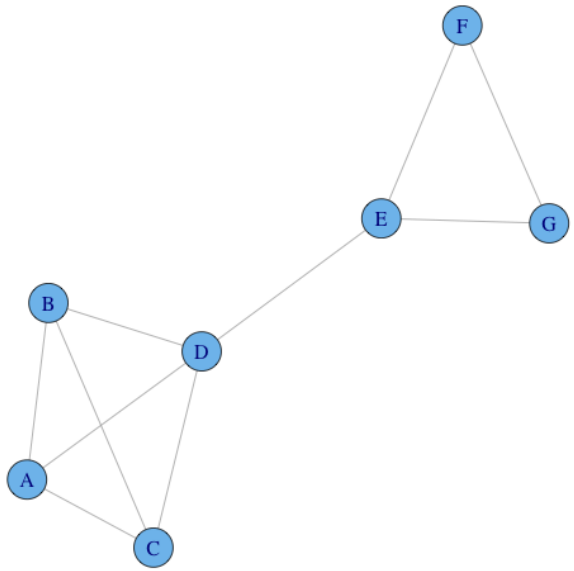
A *clique* is a complete (fully connected) subgraph, i.e. a set of vertices where each pair of vertices is connected.



Cliques can overlap

Graph cliques

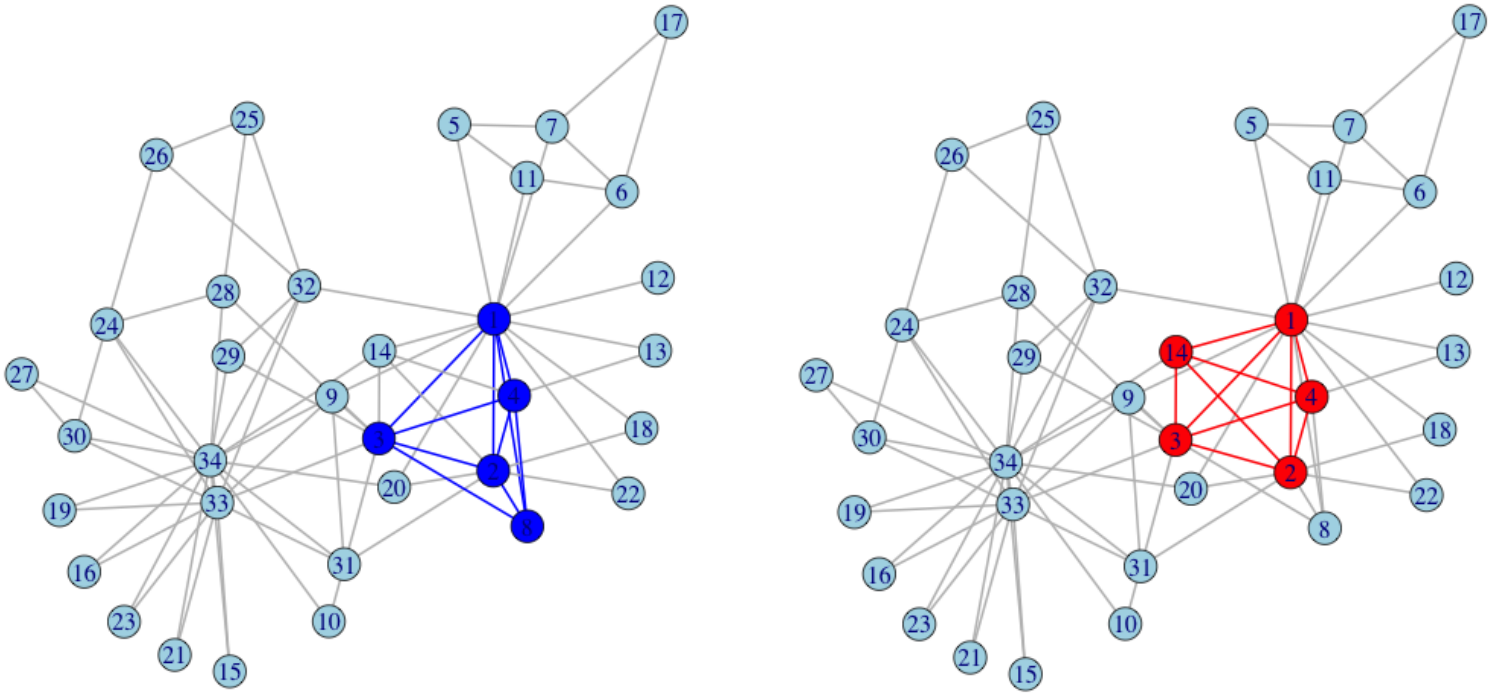
- A **maximal clique** is a clique that cannot be extended by including one more adjacent vertex (not included in larger one)
- A **maximum clique** is a clique of the largest possible size in a given graph
- Graph clique number is the size of the maximum clique



igraph: cliques(), maximal.cliques(), largest.cliques()

Graph cliques

Maximum cliques



Maximal cliques:

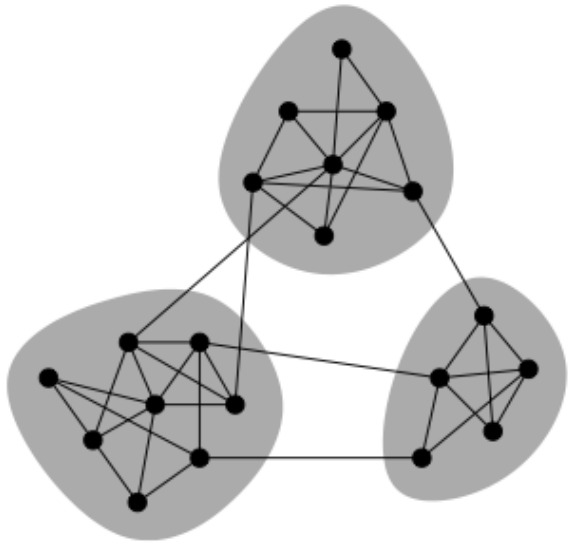
Clique size:	2	3	4	5
Number of cliques:	11	21	2	2

Zachary, 1977

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 cuts

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

$$\delta_{int}(C) = \frac{m_s}{n_s(n_s-1)/2}$$

- external edges density

$$\delta_{ext}(C) = \frac{m_{ext}}{n_c(n-n_c)}$$

- community (cluster): $\delta_{int} > \rho$, $\delta_{ext} < \rho$

Graph cuts

- Graph cut

$$Q = cut = c_s$$

- Ratio cut:

$$Q = \frac{cut}{|S|} + \frac{cut}{|V \setminus S|} = \frac{nc_s}{n_s(n - n_s)}$$

- Normalized cut:

$$Q = \frac{cut}{Vol(S)} + \frac{cut}{Vol(V \setminus S)} = \frac{c_s}{2m_s + c_s} + \frac{c_s}{2(m - m_s) + c_s}$$

- Conductance (quotient cut):

$$Q = \frac{cut}{\min\left(Vol(S), Vol(V \setminus S)\right)} = \frac{c_s}{2m_s + c_s}$$

c_s - edges crossing the boundary between S and $V \setminus S$

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

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

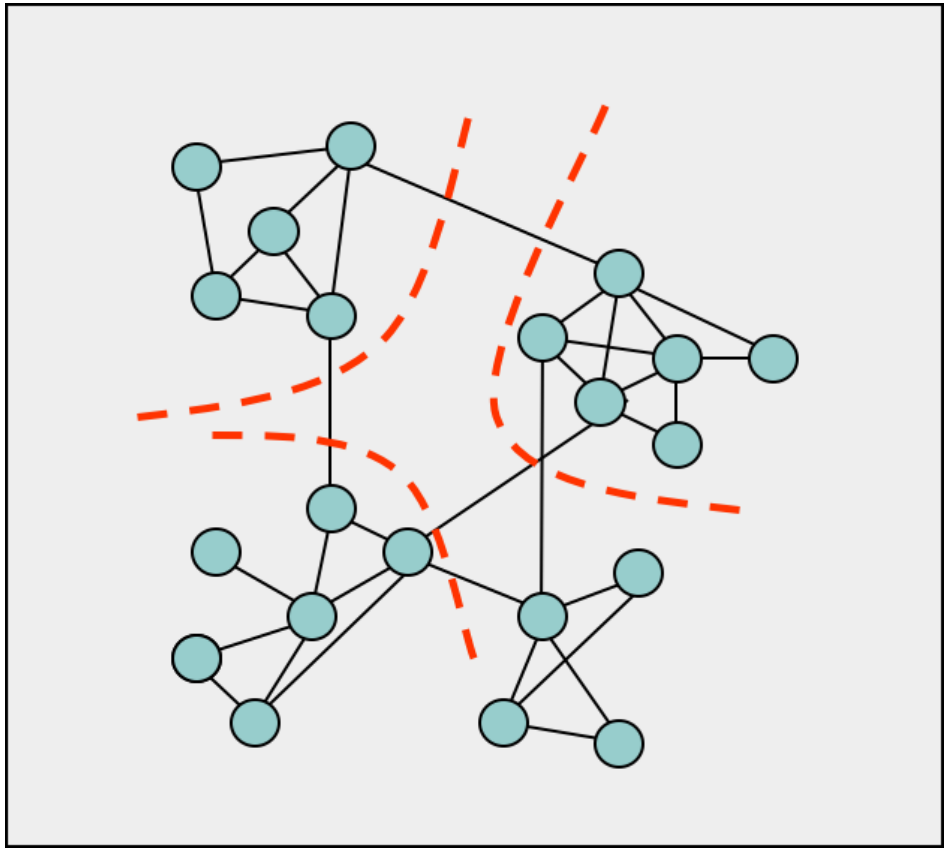
e_{uu} - fraction of edges within community u
$a_u = \sum_u e_{uv}$ - fraction of ends of edges attached to nodes in u

- The higher the modularity score - the better are communities
- Modularity score range $Q \in [-1/2, 1)$, single community $Q = 0$

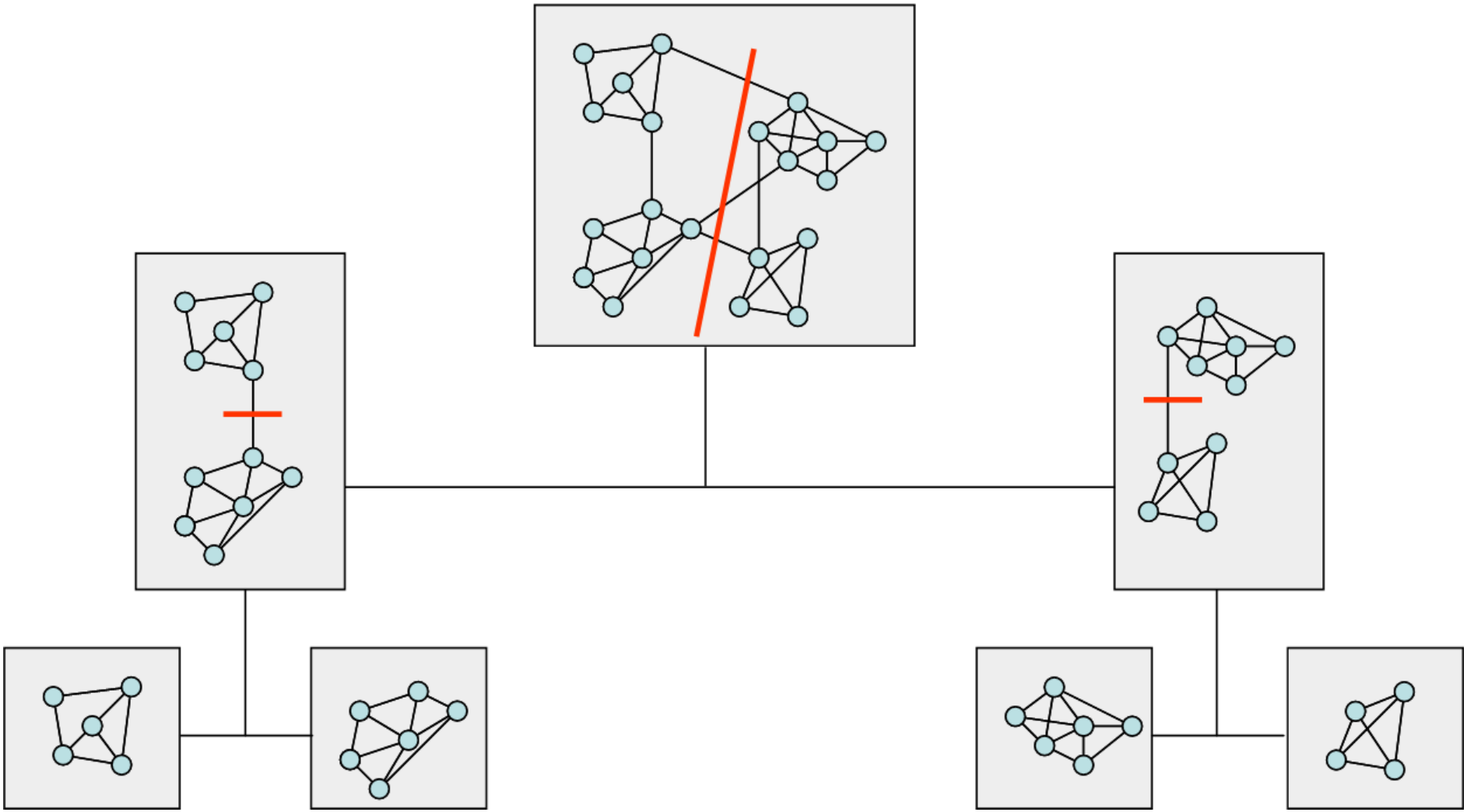
Community detection

- Consider only sparse graphs $m \ll n^2$
- Each community should be connected
- Combinatorial optimization problem:
  - optimization criterion (cut, conductance, modularity)
  - 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 partition
- Balanced class partition vs communities

Multiway partitioning



Recursive partitioning



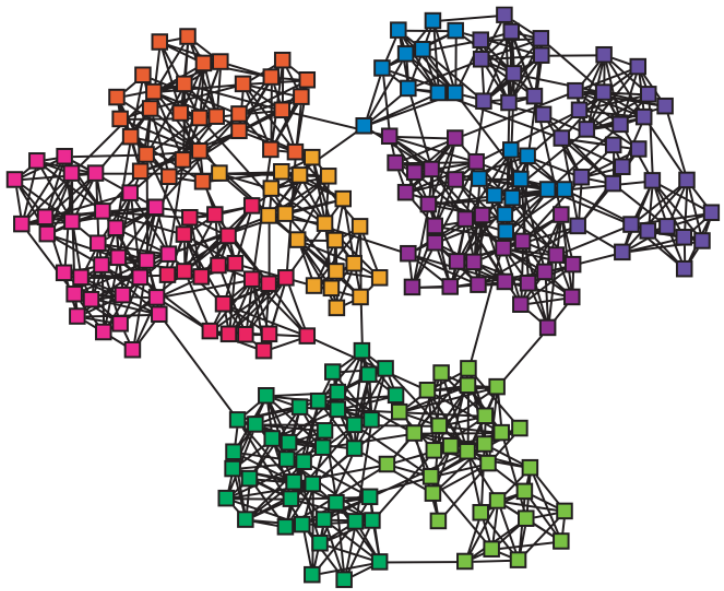
recursive partitioning

Edge betweenness

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 algorithm

Newman-Girvan, 2004

Algorithm: Edge Betweenness

Input: graph G(V,E)

Output: Dendrogram/communities

repeat

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

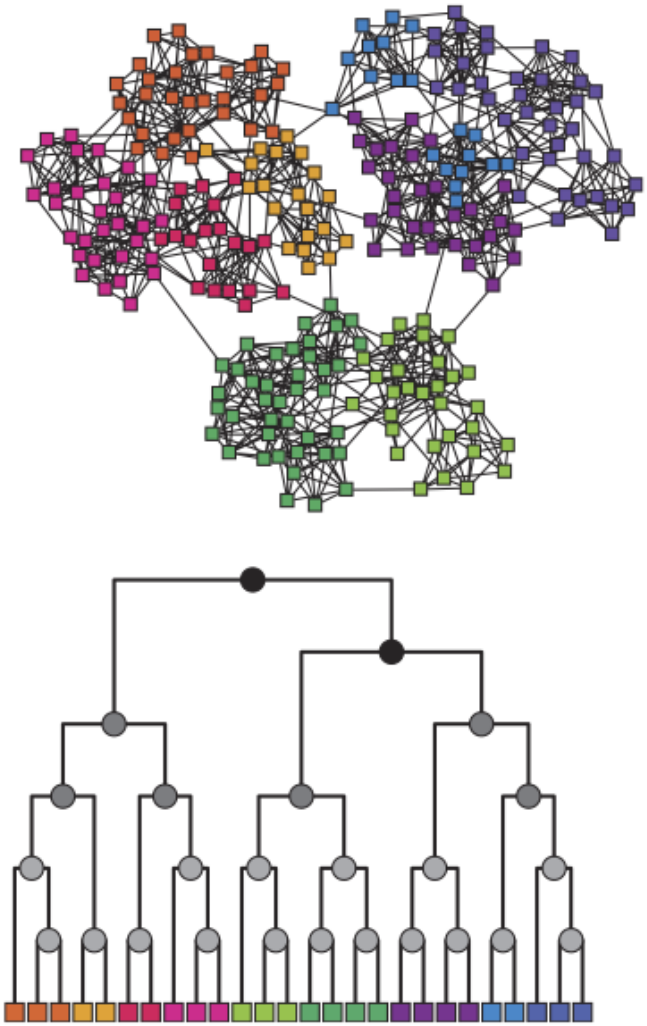
remove edge e_i with largest $C_B(e_i)$;

until *edges left*;

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

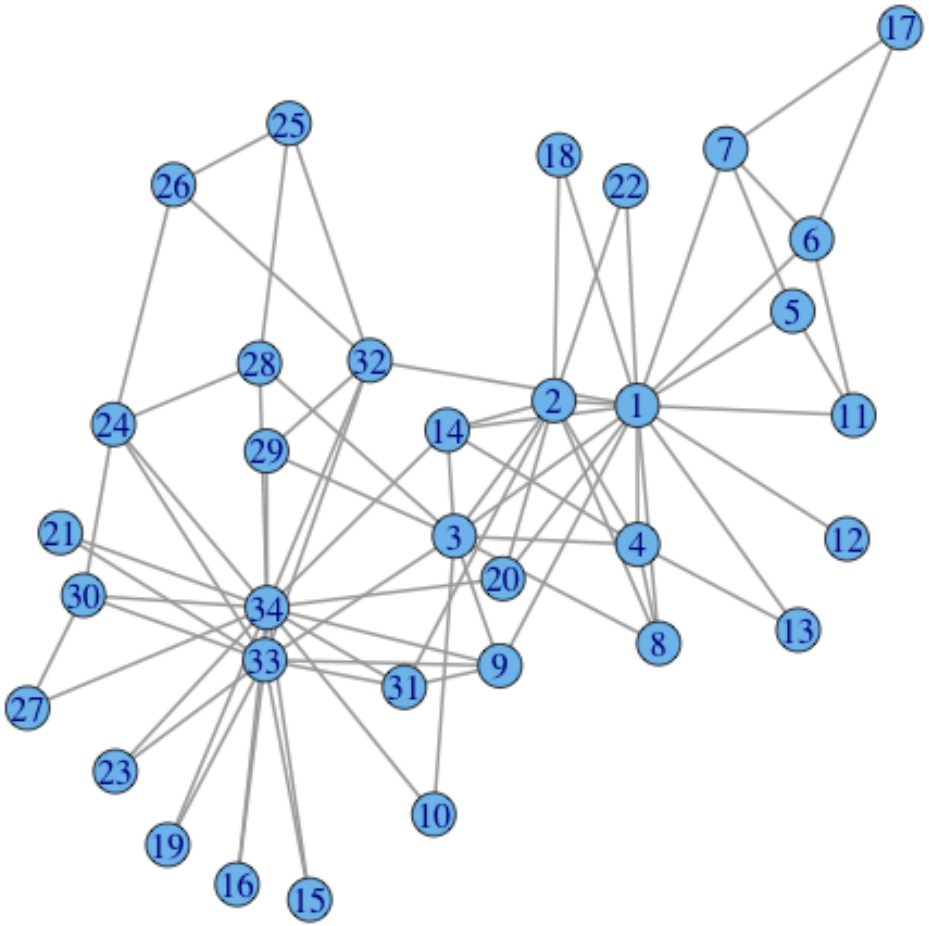
Edge betweenness

Hierarchical algorithm, dendrogram



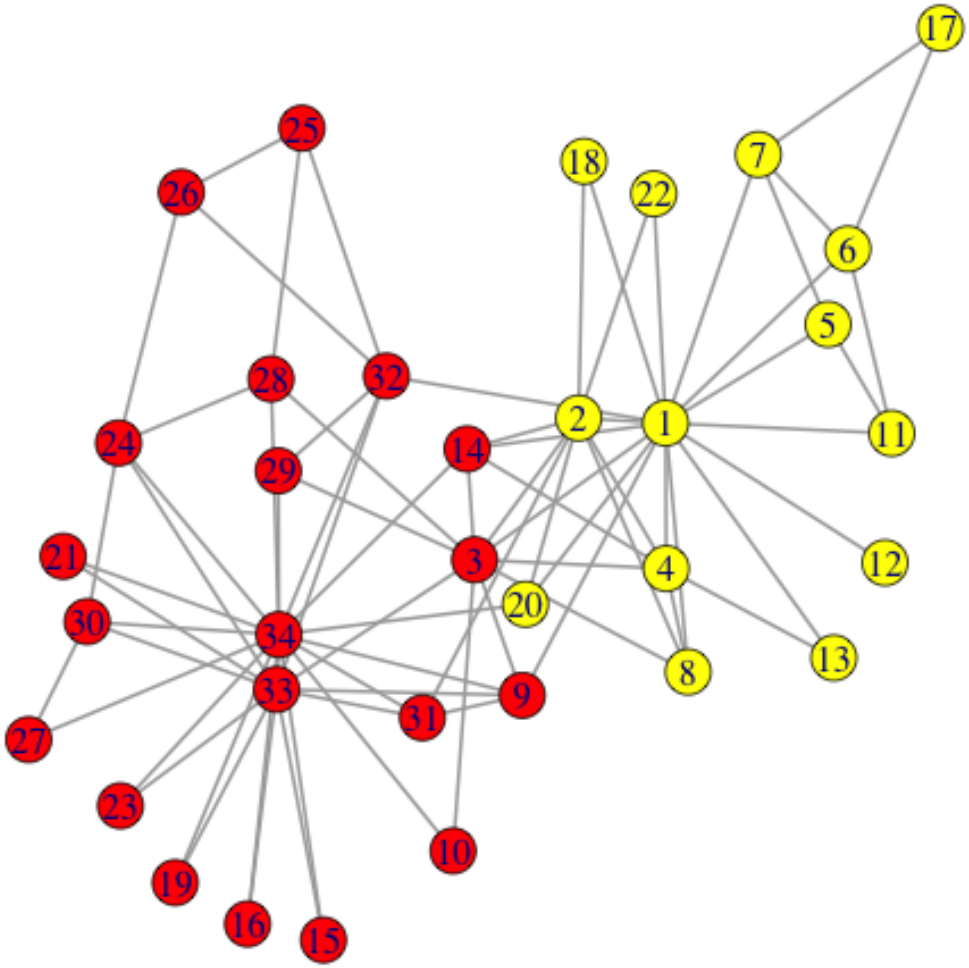
Edge betweenness

Zachary karate club



Edge betweenness

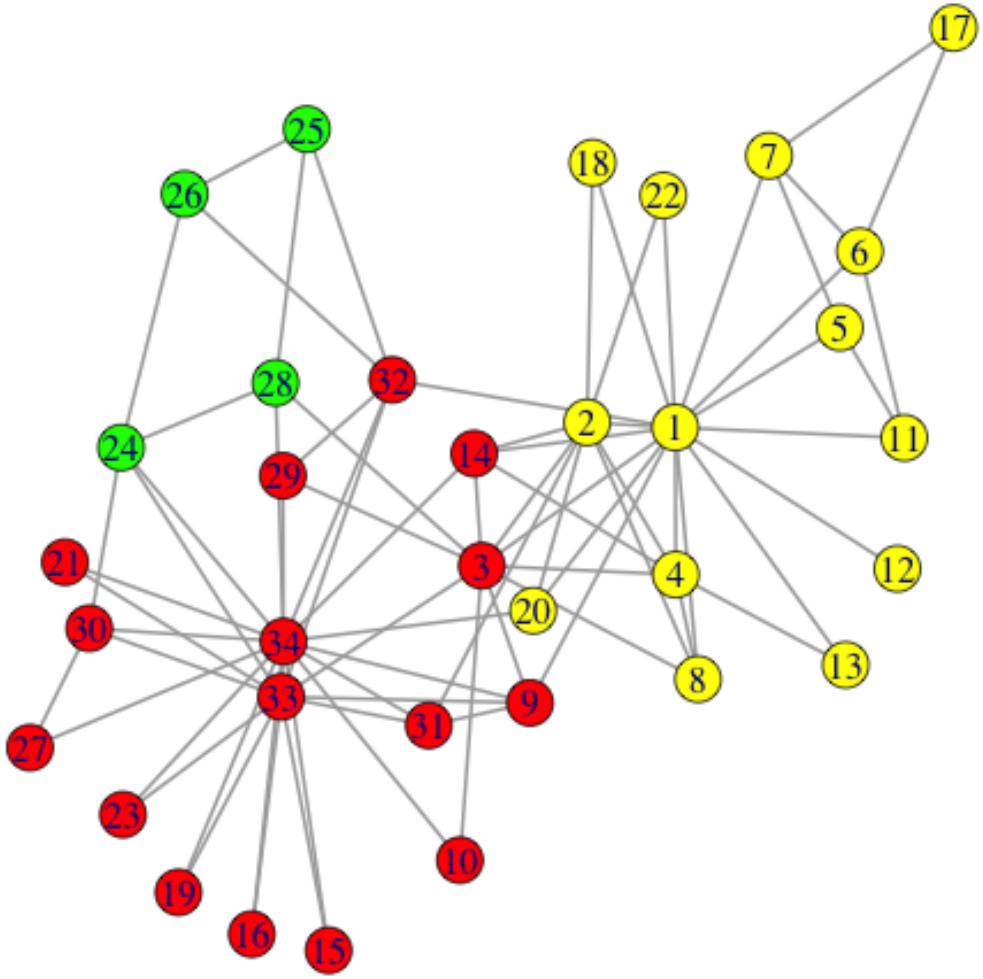
Zachary karate club



igraph:edge.betweenness.community()

Edge betweenness

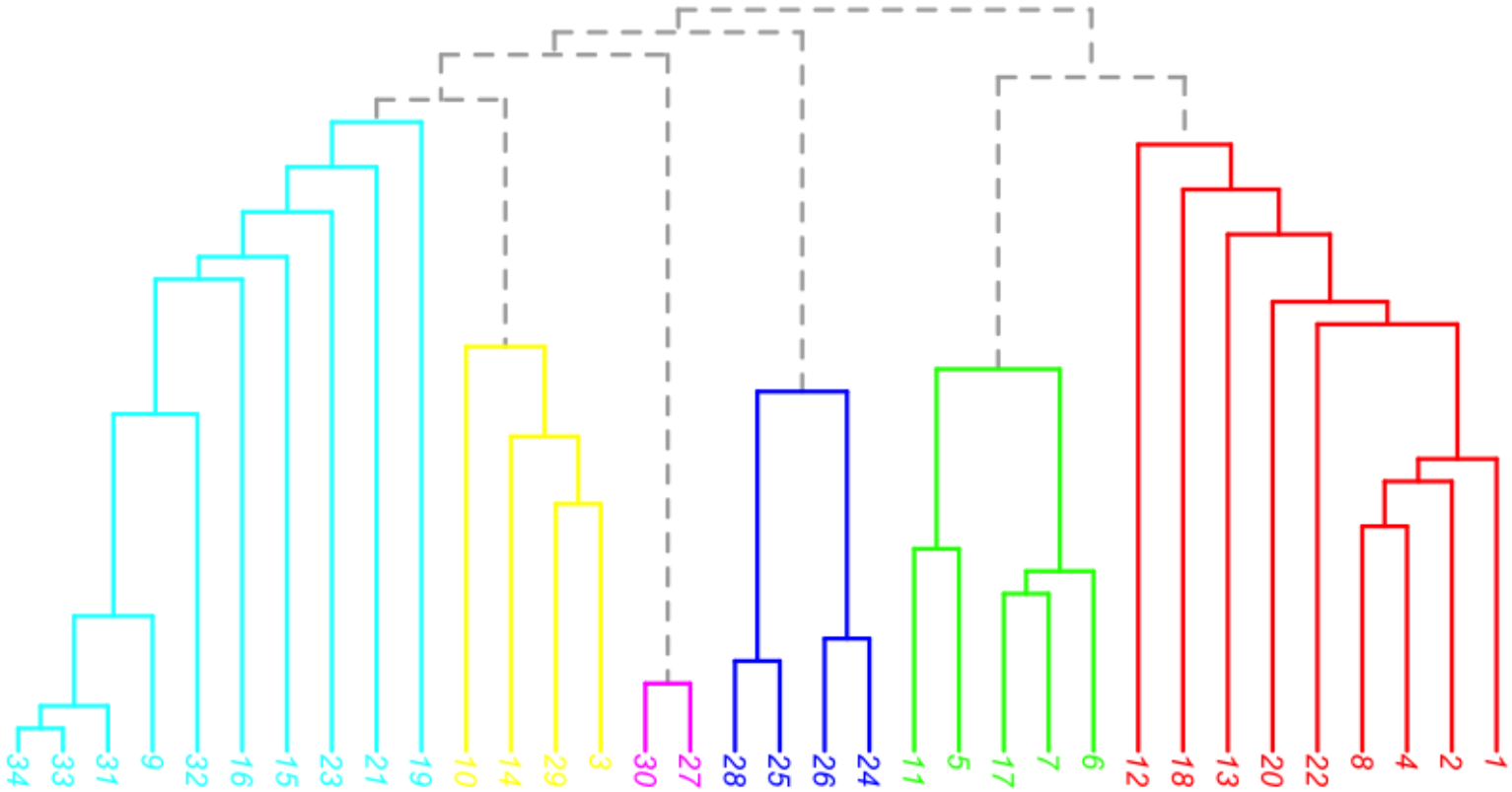
Zachary karate club



igraph:edge.betweenness.community()

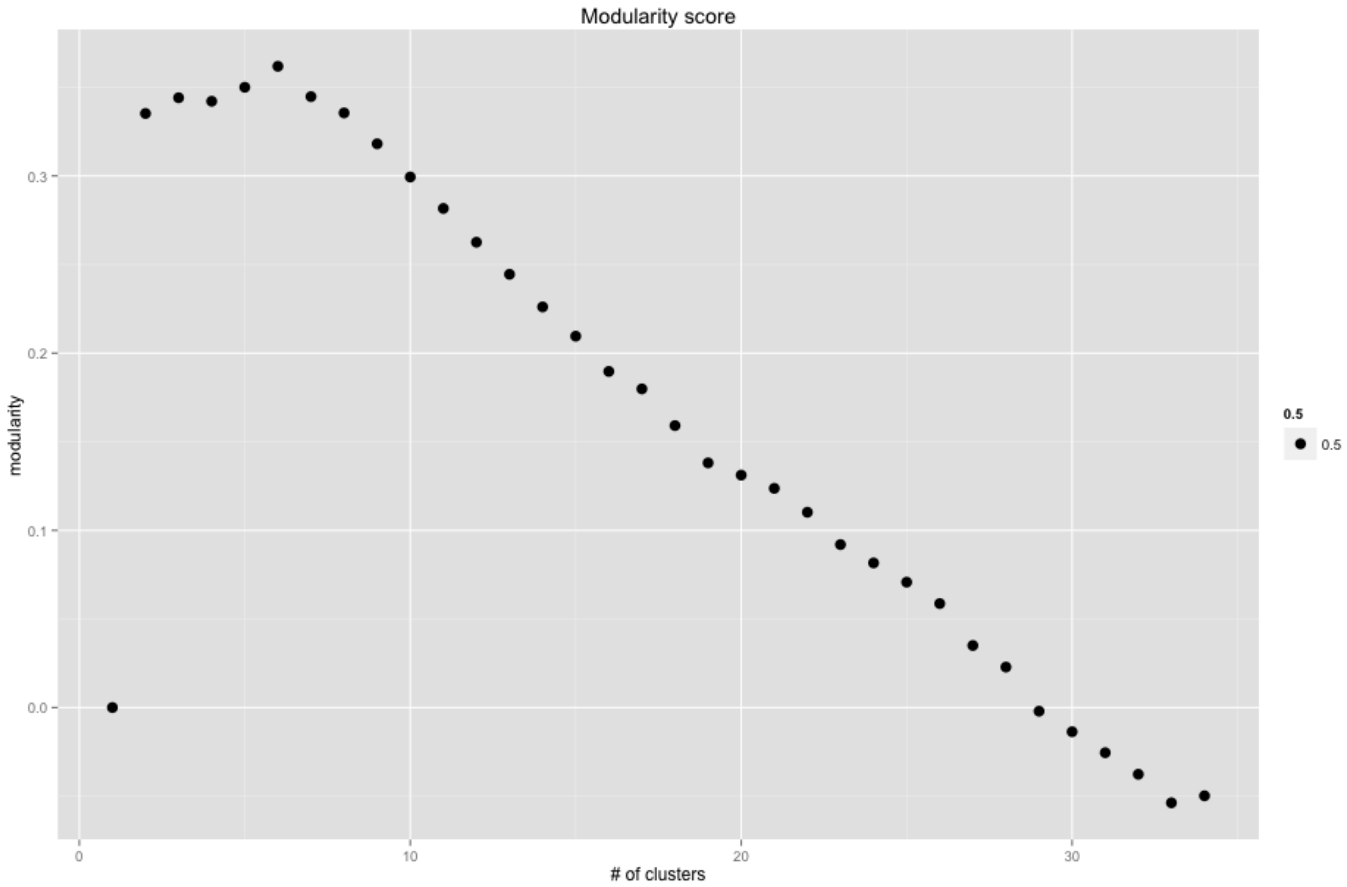
Edge betweenness

Zachary karate club



igraph:dendPlot()

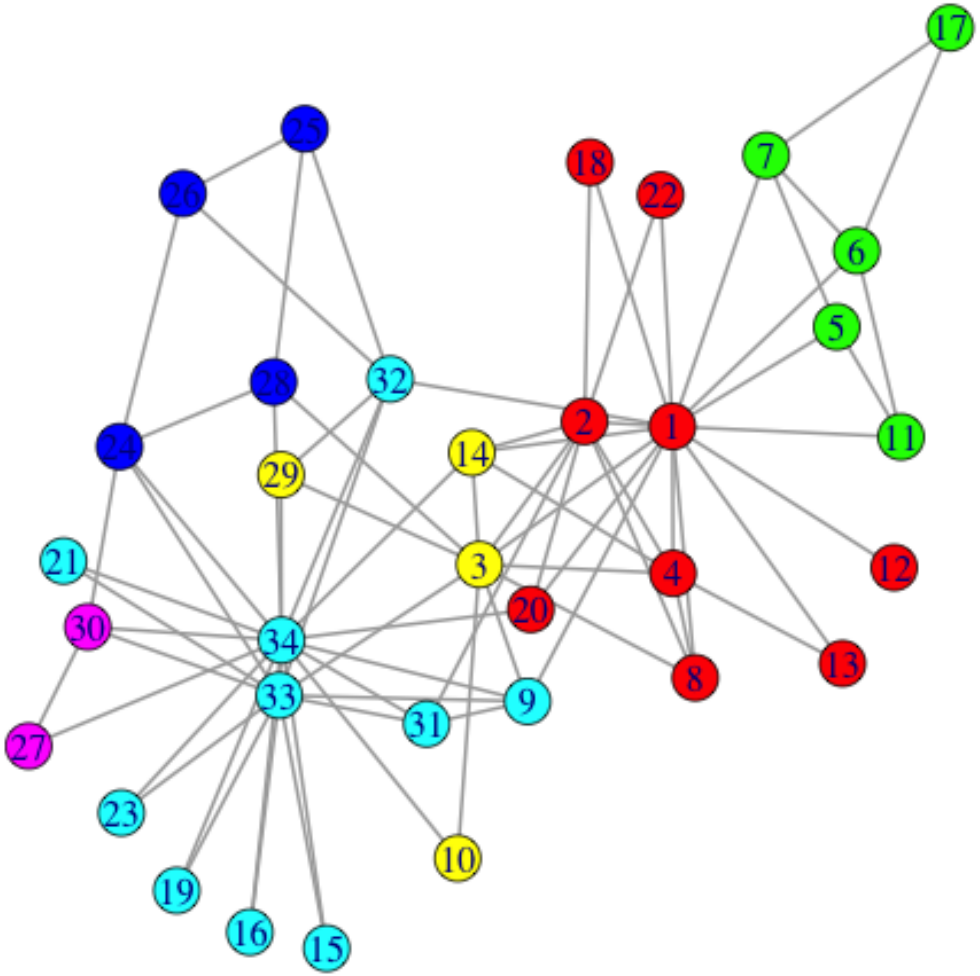
Edge betweenness



igraph:modularity()

Edge betweenness

best: clusters = 6, modularity = 0.345



igraph:edge.betweenness.community()

Spectral graph partitioning

- Indicator vector $s_i = \pm 1$
- Integer optimization problem
- Normalized cuts:

$$Q = \frac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s}, \quad \mathbf{L} = \mathbf{D} - \mathbf{A}$$

- Modularity optimization:

$$Q = \frac{1}{4m}\mathbf{s}^T\mathbf{B}\mathbf{s}, \quad \mathbf{B}_{ij} = \mathbf{A}_{ij} - \frac{k_i k_j}{2m}$$

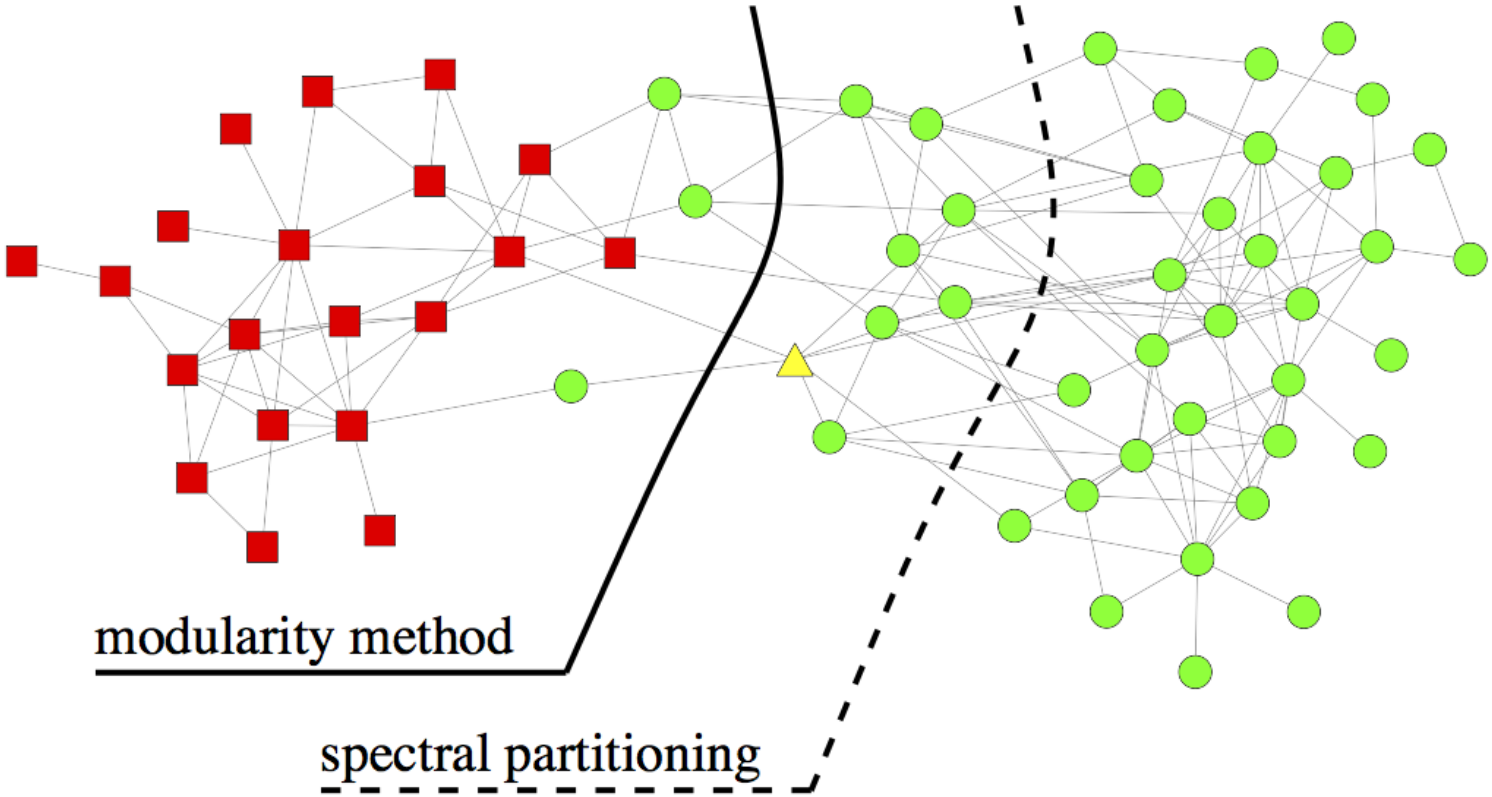
- Relaxation $s \to x$, $s \in Z^n$, $x \in R^n$
- Quadratic optimization problem under constraints
- Solved by finding min/max eigenvalues and eigenvectors of $\mathbf{L}$ or $\mathbf{B}$

$$\mathbf{L}\mathbf{x} = \lambda\mathbf{D}\mathbf{x}, \quad \text{or} \quad \mathbf{B}\mathbf{x} = \lambda\mathbf{x},$$

- Eigenvector rounds up to indicator vector $\mathbf{s} = sign(\mathbf{x})$

Fiedler 1973, Poten and Simon 1990, Newman 2006

Spectral partitioning



Newman, 2006

Spectral modularity maximization

M. Newman, 2006

Algorithm: Spectral modularity maximization: two-way partition

Input: adjacency matrix $\mathbf{A}$

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

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

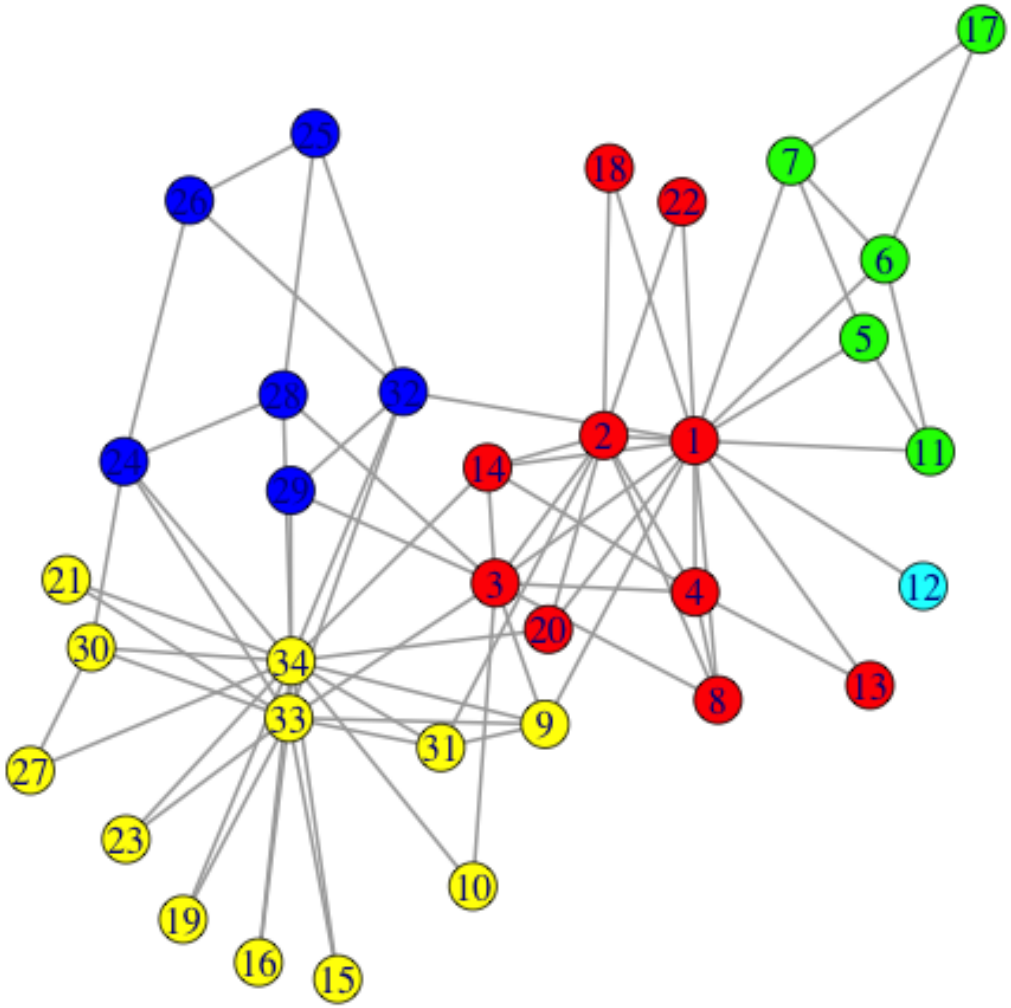
compute $\mathbf{B} = \mathbf{A} - \frac{1}{2m}\mathbf{k}\mathbf{k}^T$;

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

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

Leading eigenvector

clusters = 5, modularity = 0.437



igraph:leading.eigenvector.community()

Label propagation algorithm

U.N. Raghavan, R. Albert, S. Kumara, 2007

Algorithm: Label propagation

Input: Graph G(V,E)

Output: Communities

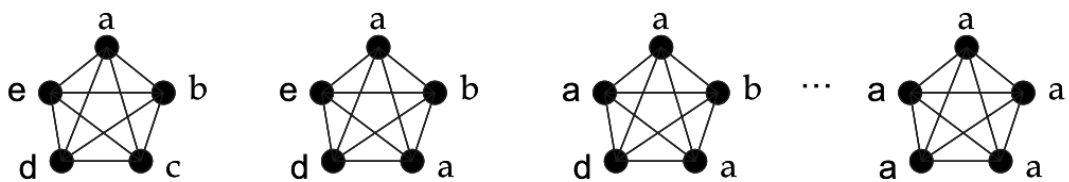
Initialize labels on all nodes;

Randomized node order;

repeat

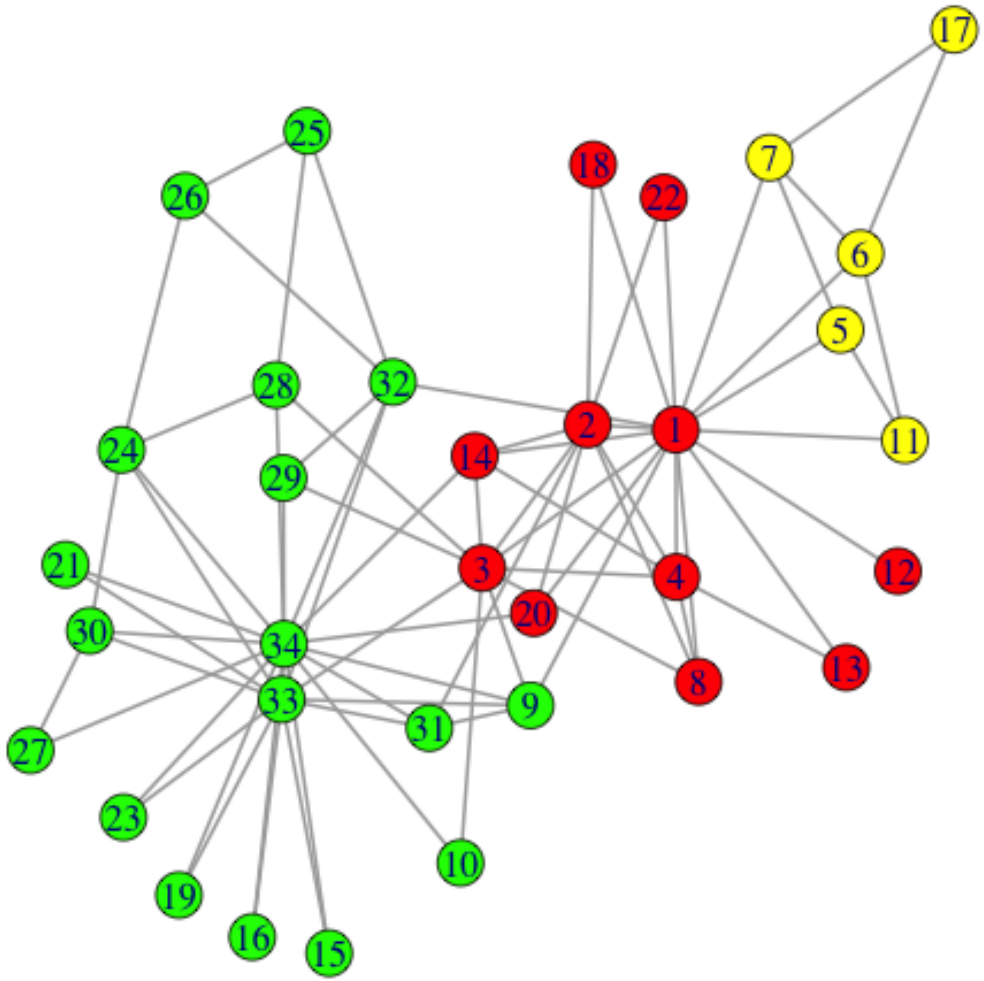
> For every node replace its label with occurring with the highest frequency among neighbors (ties are broken uniformly randomly);

until *every node has a label that the maximum number of the neighbors have*;



Label propagation

clusters = 3, modularity = 0.435



igraph:label.propagation.community()

Label propagation

image from Lab41 blog

Fast community unfolding

V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, 2008 "The Louvain method"

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

V. Blondel et.al., 2008

Fast community unfolding algorithm

Algorithm: Fast unfolding

Input: Graph G(V,E)

Output: Communities

Assign every node to its own community;

repeat

  repeat

    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;

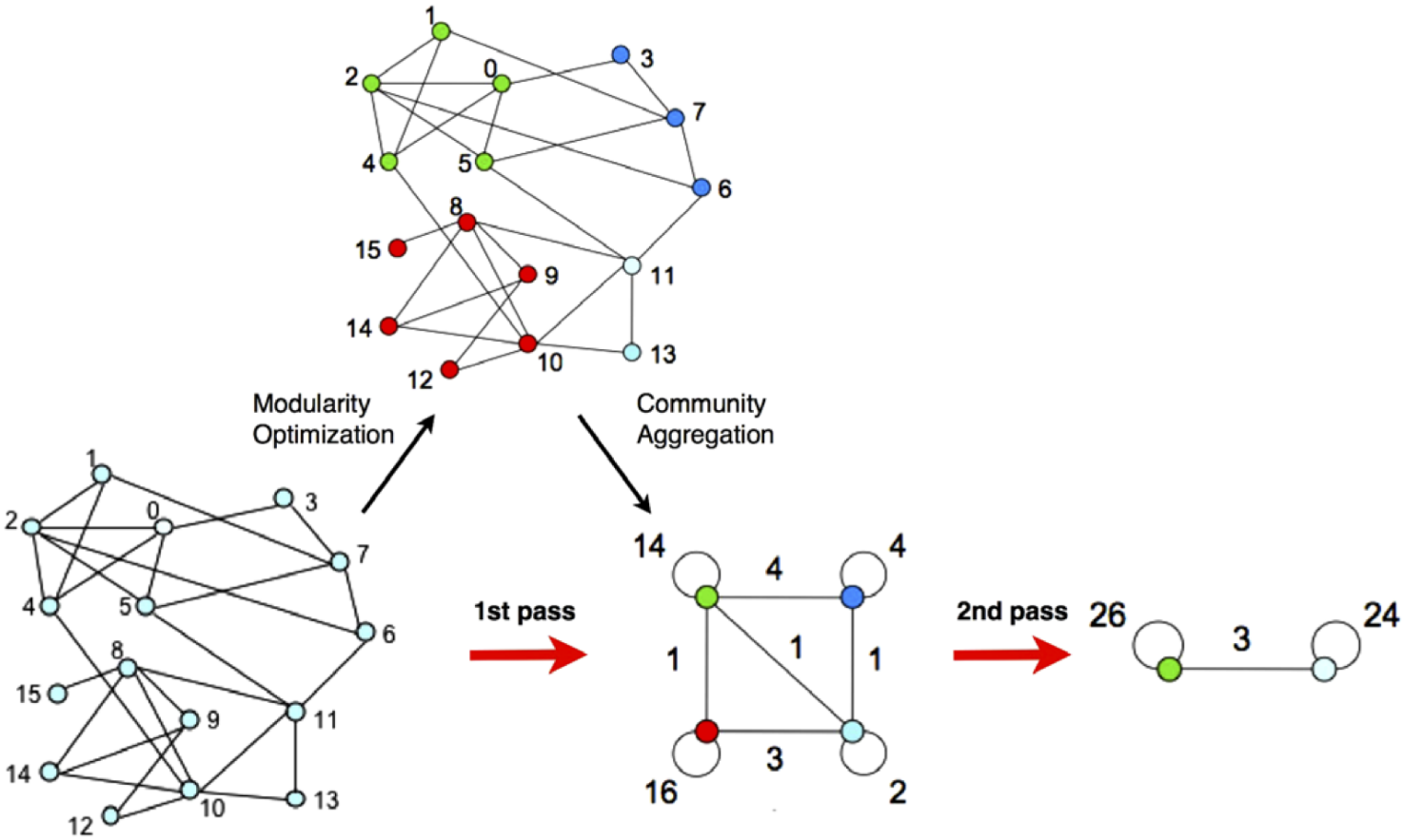
  until *no more improvement (local max of modularity)*;

  Nodes from communities merged into "super nodes" ;

  Weight on the links added up

until *no more changes (max modularity)*;

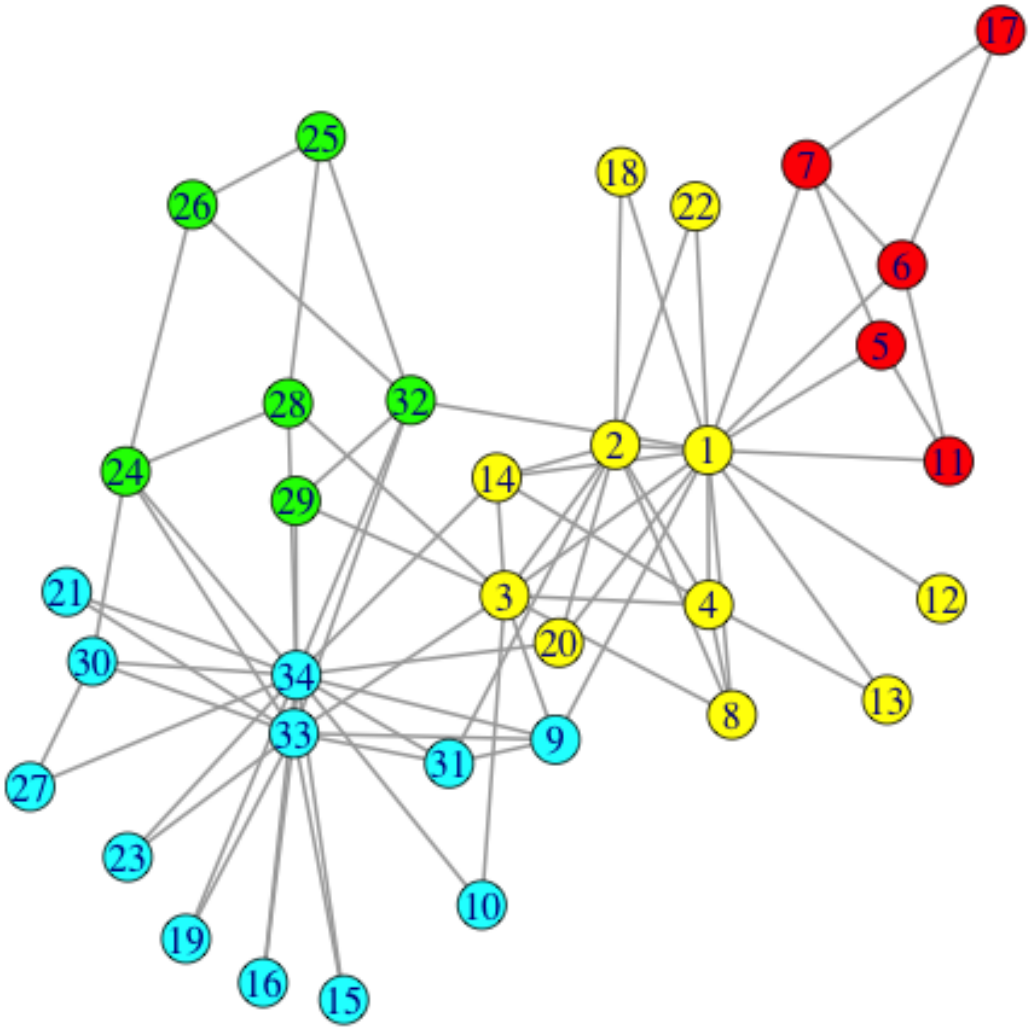
Fast community unfolding



V. Blondel et.al., 2008

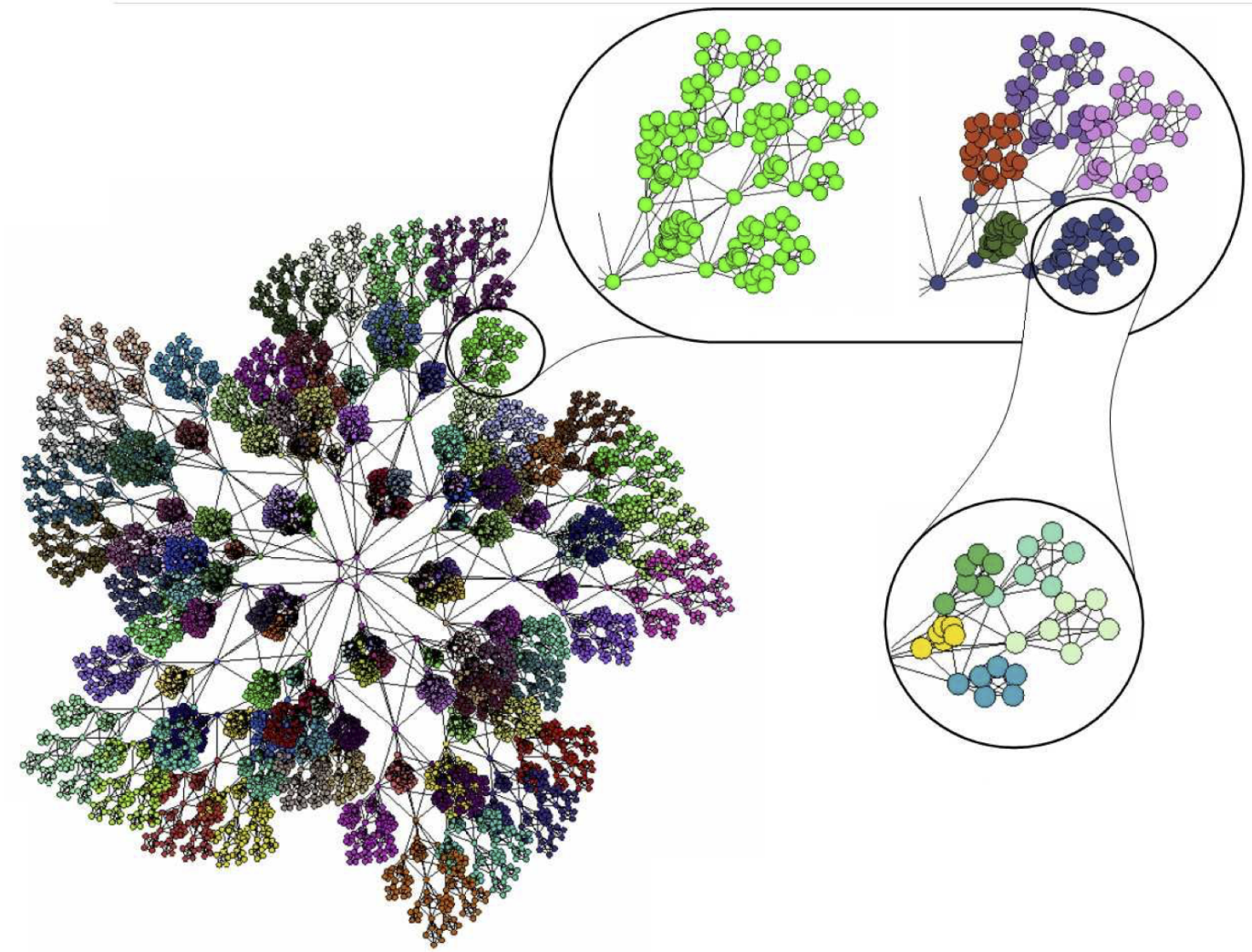
Fast community unfolding

clusters = 4, modularity = 0.445



igraph:multilevel.community()

Fast community unfolding



V. Blondel et.al., 2008

Walktrap community

P. Pons and M. Latapy, 2006 "Random walks based community detection"

- Consider random walk on graph
- At each time step walk moves to NN uniformly at random
- Distance between nodes $r_{ij}(t)$ is computed as probability P_{ij}^t to get from one to another in t steps
- Computations:
  - exact, matrix multiplication
  - approximate, random walk simulation
- Vertex clustering (agglomerative algorithm)

P. Pons and M. Latapy, 2006

Walktrap

Algorithm: Walktrap community detection

Input: Graph G(V,E)

Output: Dendrogram/communities

Assign each vertex to its own community;

Compute random walk distance between adjacent vertices;

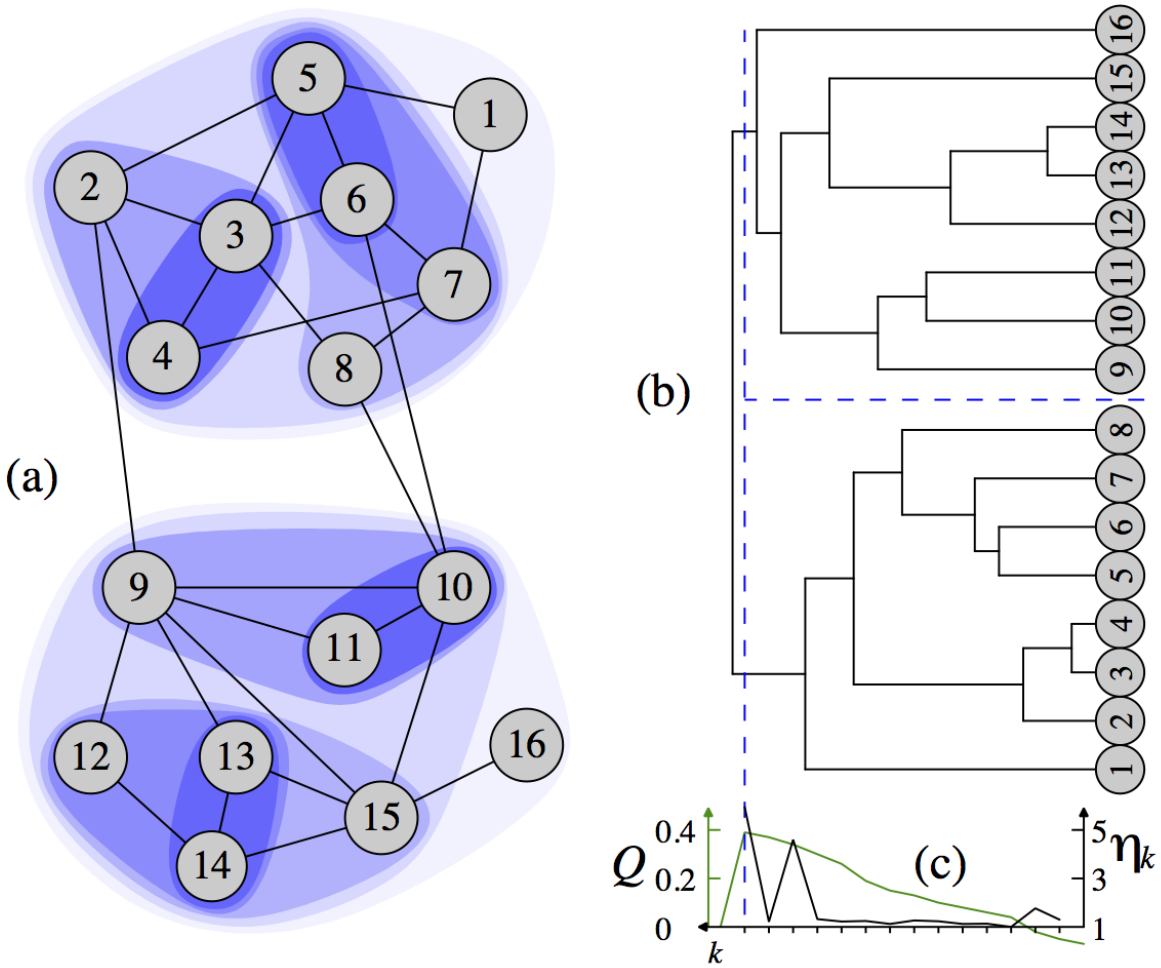
for *n-1 steps* **do**

  choose two "closest" communities and merge them ;

  update distance between communities

end

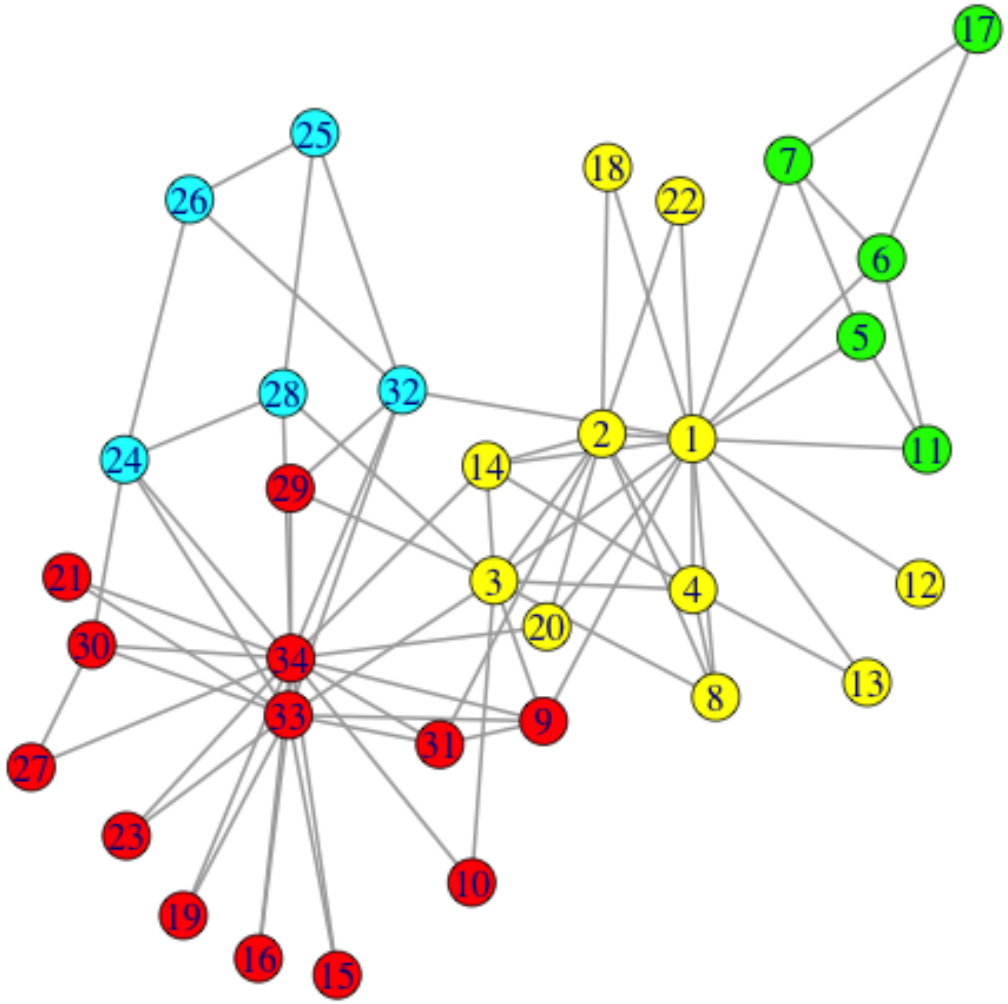
Walktrap



P. Pons and M. Latapy, 2006

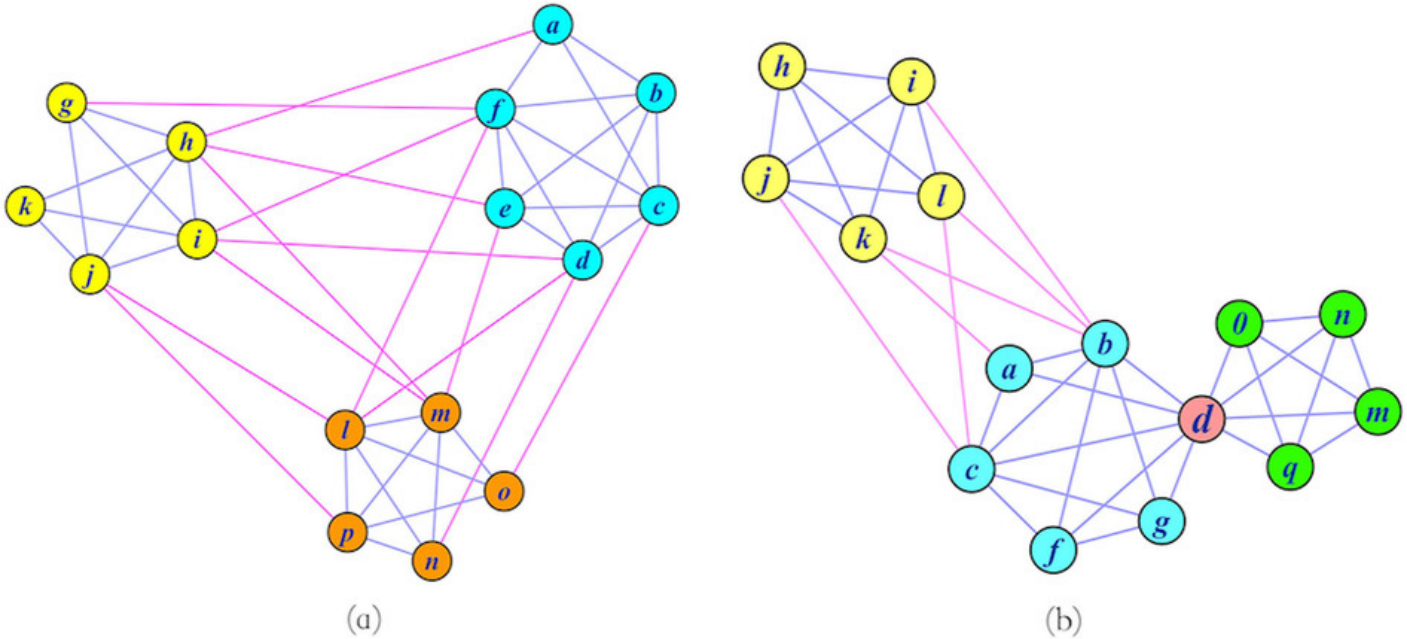
Walktrap

clusters = 4, modularity = 0.440



igraph: walktrap.community()

Overlapping communities

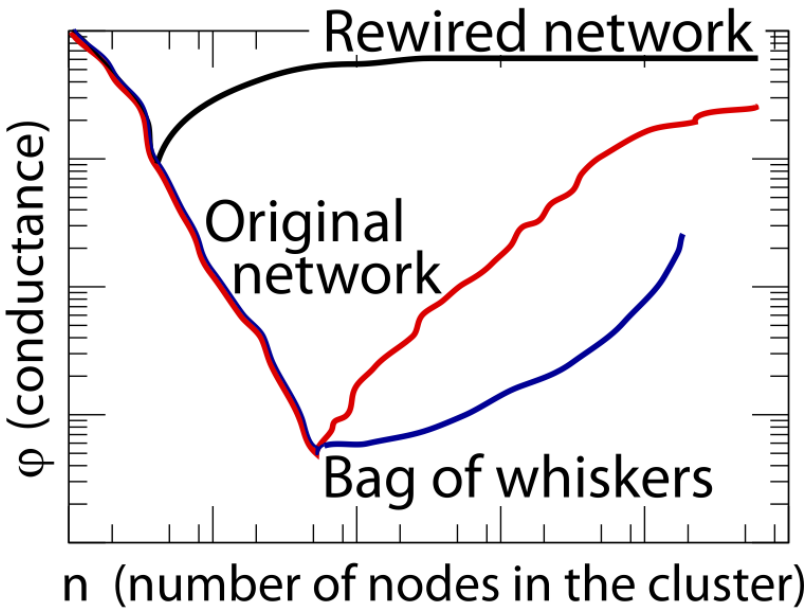


Community detection:

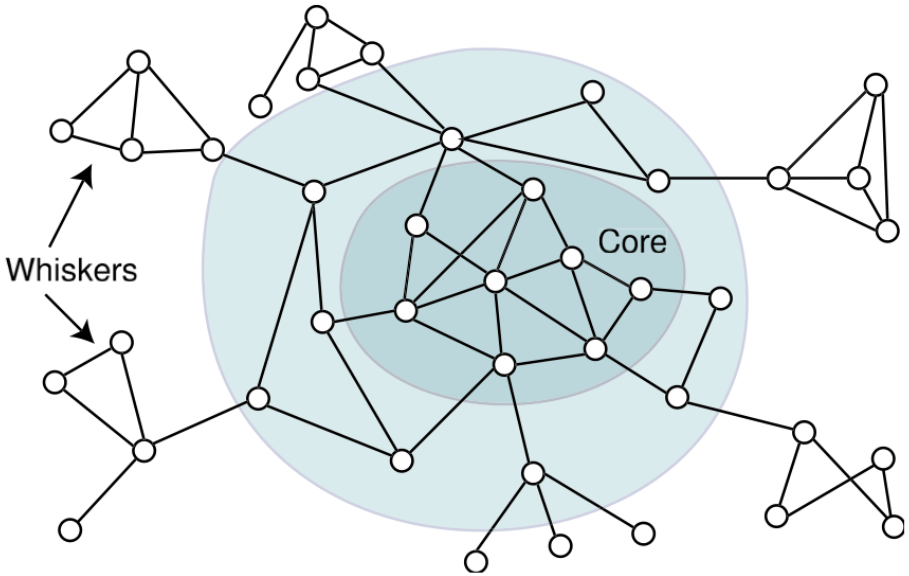
- Graph partitioning (sparse cuts)
- Vertex clustering (vertex similarity)

image from W. Liu , 2014

Real world communities



(a) Typical NCP plot



(b) Caricature of network structure

J. Leskovec, K. Lang, 2010

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	$O(n^2m)$
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	$O(m^3n)$
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

References

- S. Fortunato. Community detection in graphs, Physics Reports, Vol. 486, Iss. 35, pp 75-174, 2010
- S. E. Schaeffer. Graph clustering. Computer Science Review, 1(1):2764, 2007.
- Modularity and community structure in networks, M.E.J. Newman, PNAS, vol 103, no 26, pp 8577-8582, 2006
- Finding and evaluating community structure in networks, M.E.J. Newman, M. Girvan, Phys. Rev E, 69, 2004
- U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (3) (2007) 036106.
- 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).