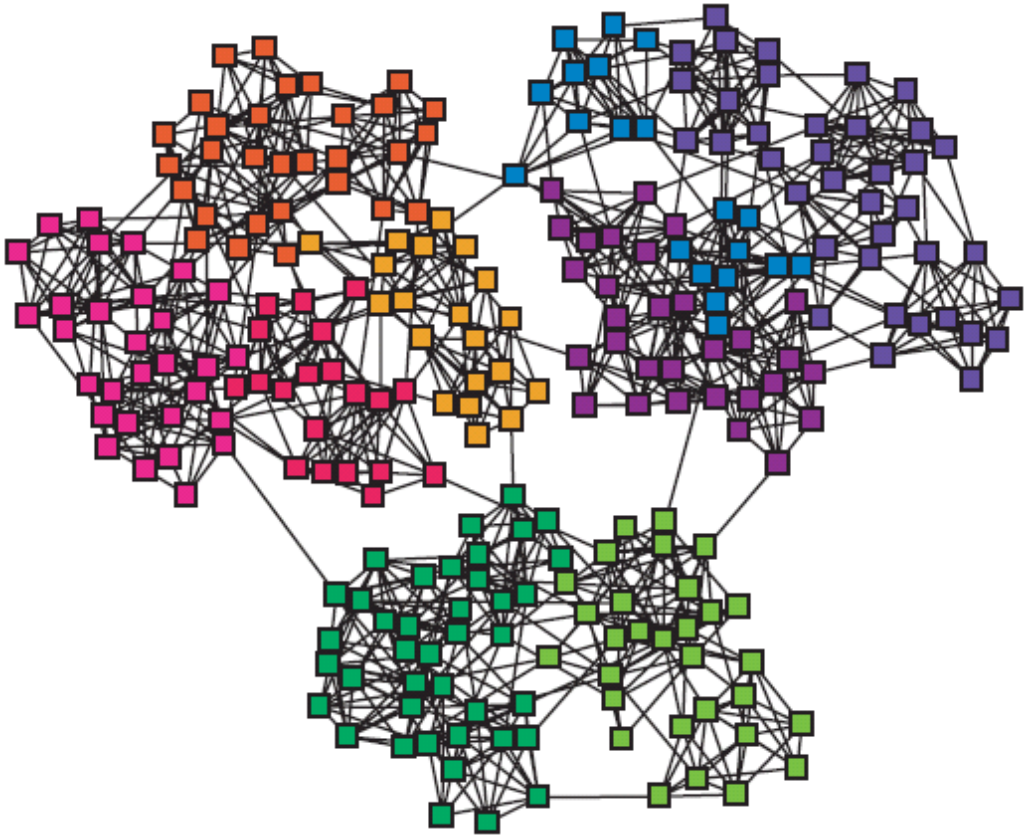
Online Social Networks and Media

Graph Partitioning

Introduction

modules, cluster, communities, groups, *partitions*
(more on this today)



Outline

PART I

1. Introduction: what, why, types?
2. Cliques and vertex similarity
3. Background: Cluster analysis
4. Hierarchical clustering (betweenness)
5. Modularity
6. *How to evaluate (if time allows)*

Outline

PART II

1. Cuts
2. Spectral Clustering
3. Dense Subgraphs

partitions

4. Community Evolution
5. *How to evaluate (from Part I)*

Graph partitioning

The general problem

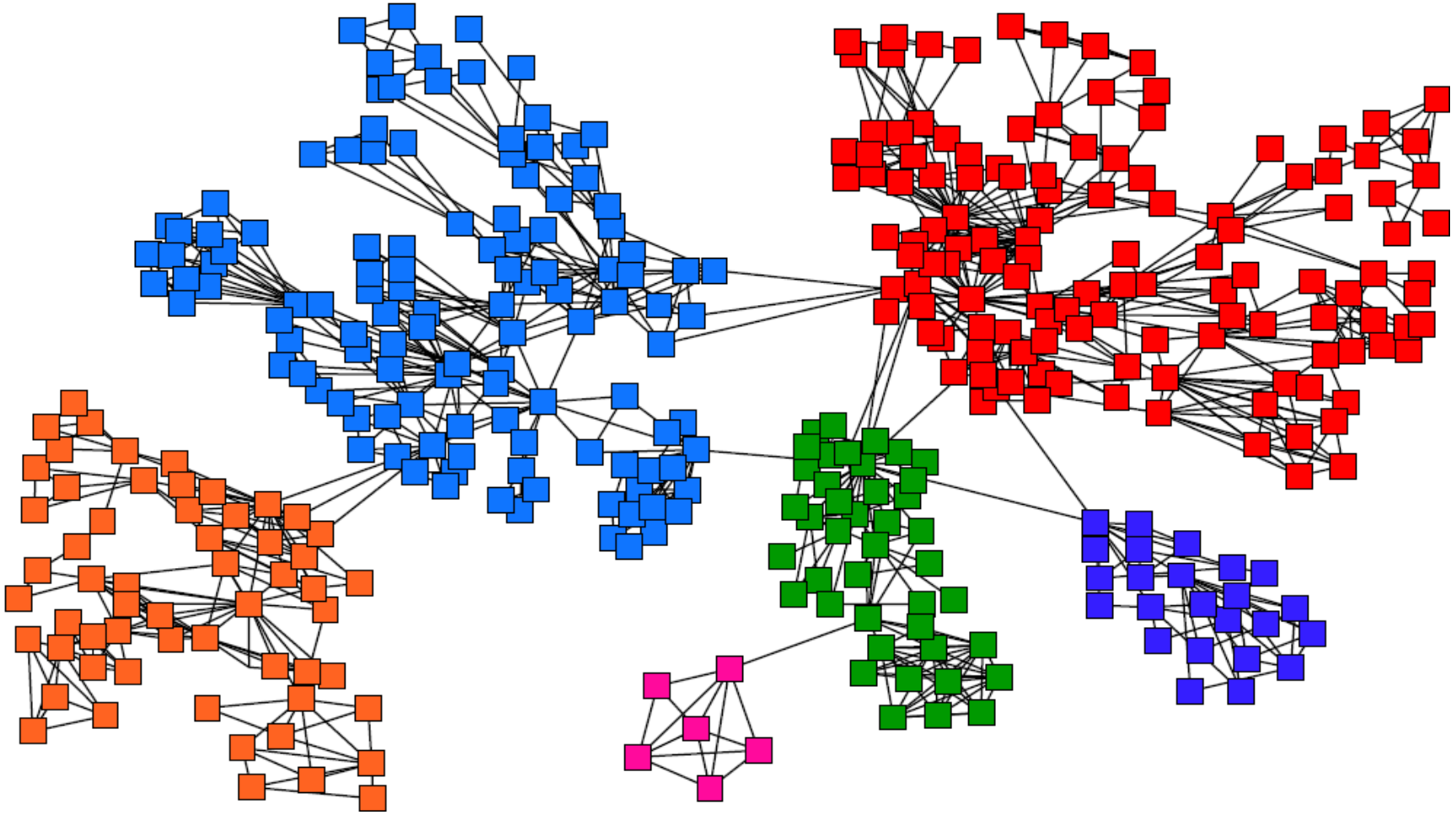
- Input: a graph G = (V, E)
  - edge (u, v) denotes similarity between u and v
  - weighted graphs: weight of edge captures the degree of similarity

Partitioning as an optimization problem:

- Partition the nodes in the graph such that nodes *within clusters* are *well interconnected* (high edge weights), and nodes *across clusters* are *sparsely interconnected* (low edge weights)

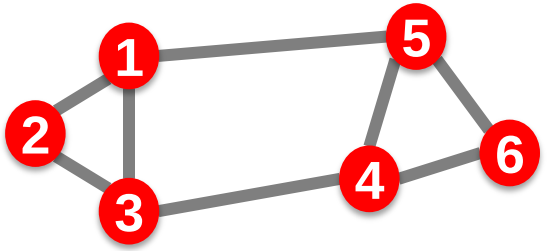
- most graph partitioning problems are NP hard

Graph Partitioning



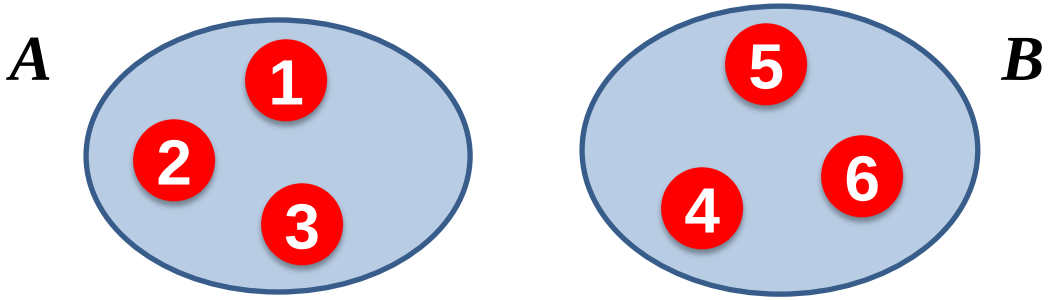
Graph Partitioning

Undirected graph $G(V, E)$:



Bi-partitioning task:

Divide vertices into two disjoint groups $\boldsymbol{A}, \boldsymbol{B}$



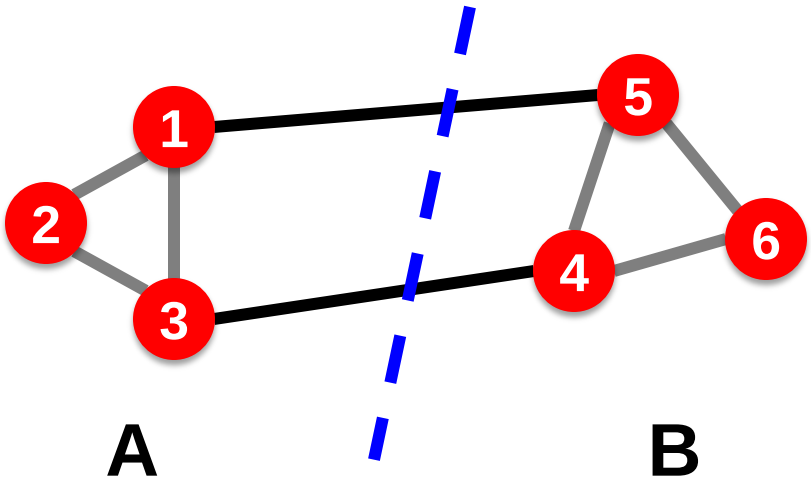
How can we define a "good" partition of $\boldsymbol{G}$?

How can we efficiently identify such a partition?

Graph Partitioning

What makes a good partition?

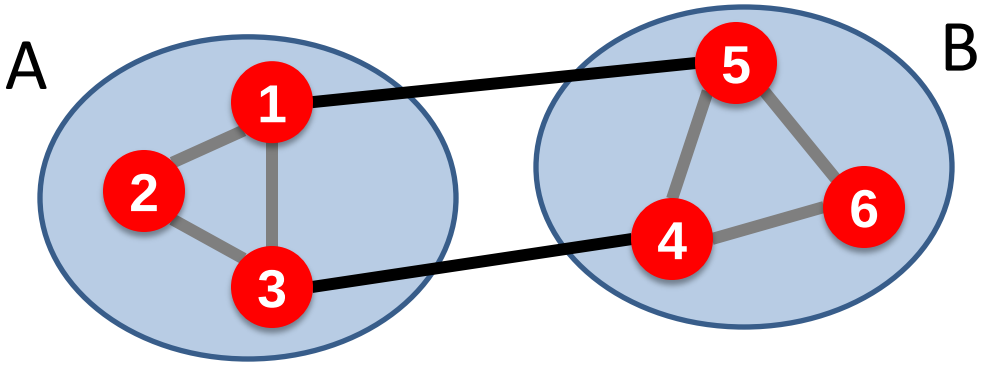
- Maximize the number of within-group connections
- Minimize the number of between-group connections



Graph Cuts

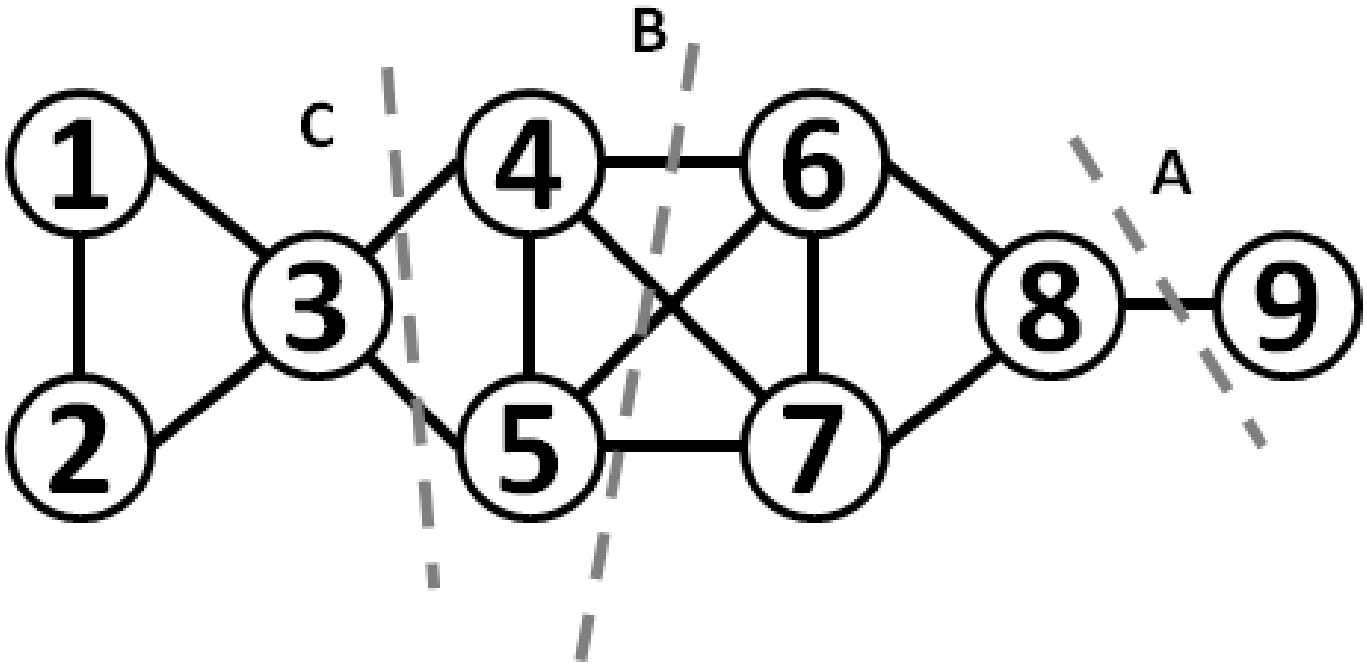
Express *partitioning objectives* as a function of the "edge cut" of the partition

Cut: Set of edges with only one vertex in a group: $cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$



$cut(A,B) = 2$

An example



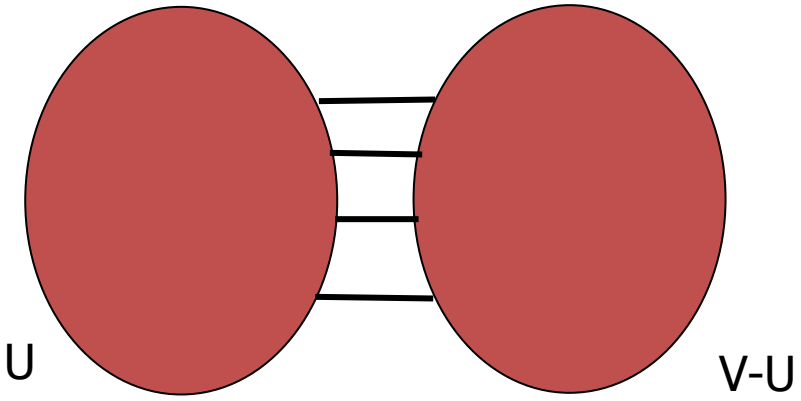
Min Cut

min-cut: the min number of edges such that when removed cause the graph to become disconnected

Minimizes the number of connections between partition

$$\arg\min_{A,B} \text{cut}(A,B)$$

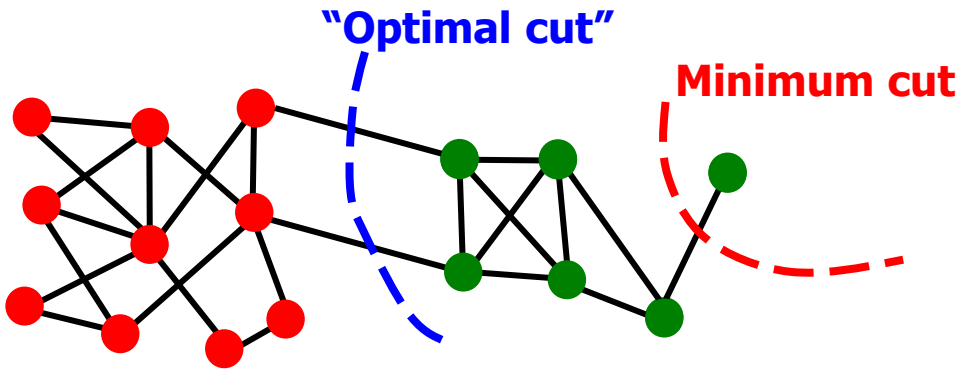
$$\min_{U} E(U, V-U) = \sum_{i \in U} \sum_{j \in V-U} A[i, j]$$



This problem can be solved in polynomial time

Min-cut/Max-flow algorithm

Min Cut



Problem:

- Only considers external cluster connections
- Does not consider internal cluster connectivity

Graph Bisection

- Since the minimum cut does not always yield good results we need extra constraints to make the problem meaningful.
- Graph Bisection refers to the problem of partitioning the nodes of the graph into two *equal sets*.
- Kernighan-Lin algorithm: Start with random equal partitions and then swap nodes to improve some quality metric (e.g., cut, modularity, etc).

Cut Ratio

Ratio Cut

Normalize cut by the *size* of the groups

$$\text{Ratio-cut} = \frac{\text{Cut}(U, V-U)}{|U|} + \frac{\text{Cut}(U, V-U)}{|V-U|}$$

Normalized Cut

Normalized-cut

Connectivity between groups relative to the *density* of each group

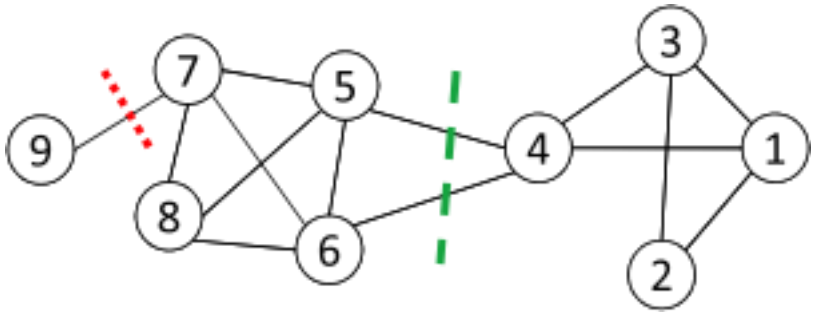
$$\text{Normalized-cut} = \frac{\text{Cut(U,V}-\text{U)}}{Vol(U)} + \frac{\text{Cut(U,V}-\text{U)}}{Vol(V-U)}$$

$vol(U)$: total weight of the edges with at least one endpoint in U: $vol(U) = \sum_{i \in U} d_i$

Why use these criteria?

- Produce more balanced partitions

An example



Red is Min-Cut

Ratio-Cut(Red) = $\frac{1}{1} + \frac{1}{8} = \frac{9}{8}$

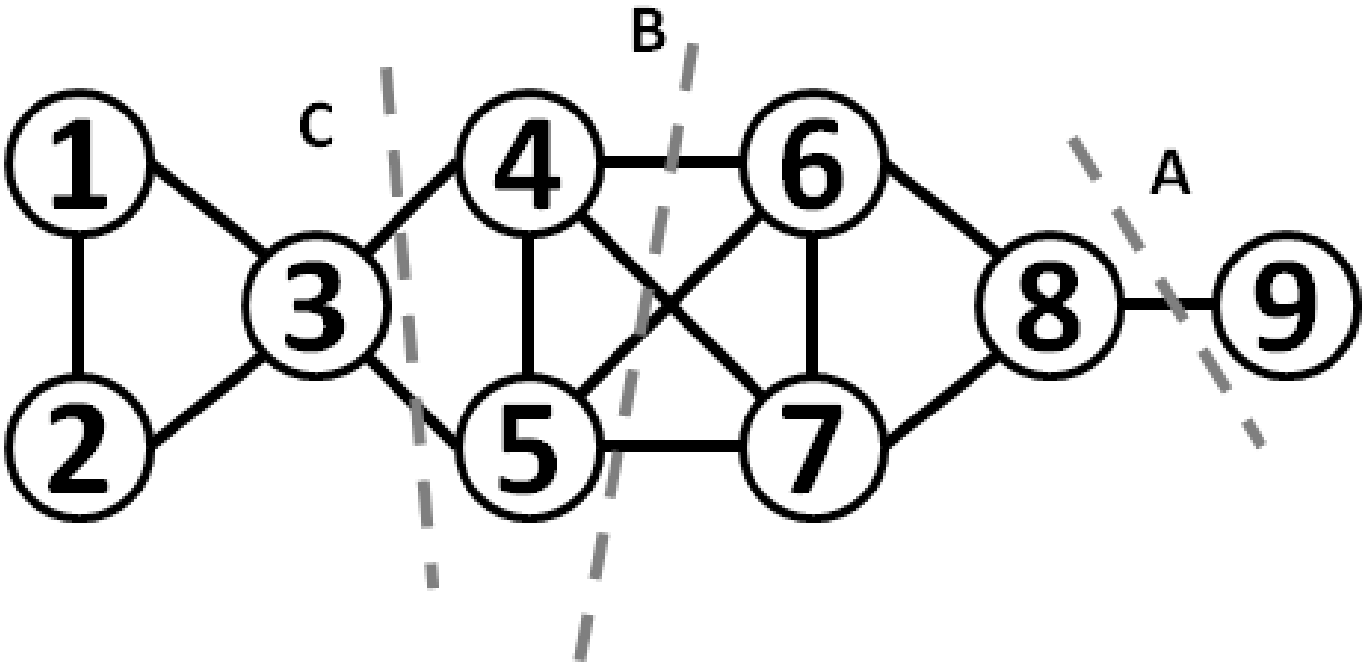
Ratio-Cut(Green) = $\frac{2}{5} + \frac{2}{4} = \frac{18}{20}$

Normalized-Cut(Red) = $\frac{1}{1} + \frac{1}{27} = \frac{28}{27}$

Normalized-Cut(Green) = $\frac{2}{12} + \frac{2}{16} = \frac{14}{48}$

Normalized is even better for Green due to density

An example



Which of the three cuts has the best (min, normalized, ratio) cut?

Graph expansion

Graph expansion:

$$\alpha = \min_{U} \frac{\text{cut}(U, V - U)}{\min\{|U|, |V - U|\}}$$

Graph Cuts

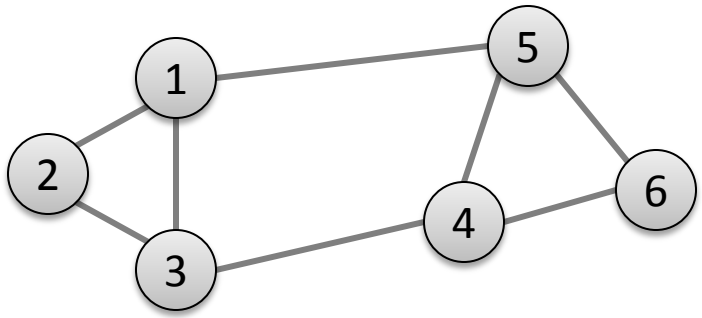
Ratio and normalized cuts can be reformulated in matrix format and solved using spectral clustering

SPECTRAL CLUSTERING

Matrix Representation

Adjacency matrix (A):

- $n \times n$ matrix
- $A=[a_{ij}]$, $a_{ij}=1$ if edge between node i and j



	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

Important properties:

- Symmetric matrix
- Eigenvectors are real and orthogonal

If the graph is weighted, $a_{ij}= w_{ij}$

Spectral Graph Partitioning

$\boldsymbol{x}$ is a vector in $\Re^n$ with components $(\boldsymbol{x_1}, \ldots, \boldsymbol{x_n})$

– Think of it as a label/value of each node of $\boldsymbol{G}$

- What is the meaning of $A \cdot x$?

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$y_i = \sum_{j=1}^{n} A_{ij} x_j = \sum_{(i,j) \in E} x_j$$

Entry y_i is a sum of labels x_j of neighbors of i

Spectral Analysis

i^{th} coordinate of $A \cdot x$:

- Sum of the x-values of neighbors of i
- Make this a new value at node j

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\boldsymbol{A \cdot x = \lambda \cdot x}$$

Spectral Graph Theory:

- Analyze the "spectrum" of a matrix representing G
- Spectrum: Eigenvectors x_i of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues λ_i: $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

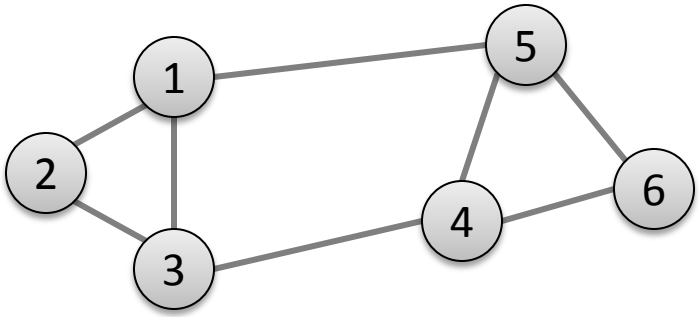
Spectral clustering: use the eigenvectors of A or graphs derived by it

Most based on the graph Laplacian

Matrix Representation

Degree matrix (D):

- $n \times n$ diagonal matrix
- $D=[d_{ii}]$, d_{ii} = degree of node i



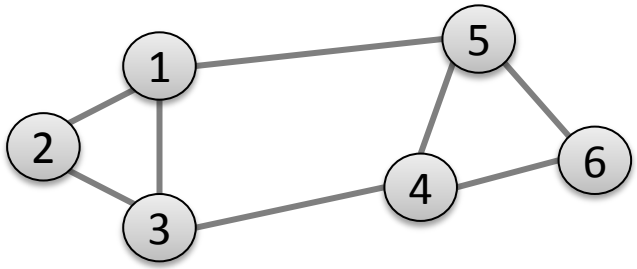
	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

Matrix Representation

Laplacian matrix (L):

- $n \times n$ symmetric matrix

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$$



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

Laplacian Matrix properties

- The matrix L is symmetric and positive semi-definite
  - all eigenvalues of L are positive

positive definite: if $z^T M z$ is non-negative, for every non-zero column vector z

- The matrix L has 0 as an eigenvalue, and corresponding eigenvector $w_1 = (1,1,\ldots,1)$
  - $\lambda_1 = 0$ is the smallest eigenvalue

Proof: Let w_1 be the column vector with all 1s -- show $Lw_1 = 0w_1$

The second smallest eigenvalue

The second smallest eigenvalue (also known as Fielder value) λ_2 satisfies

$$\lambda_2 = \min_{x \perp w_1, \|x\|=1} x^T L x$$

The second smallest eigenvalue

- For the Laplacian

$$x \perp w_1 \Rightarrow \sum_i x_i = 0$$

- The expression:

$$x^T L x$$

is

$$\sum_{(i,j)\in E} (x_i - x_j)^2$$

The second smallest eigenvalue

Thus, the eigenvector for eigenvalue λ_2 (called the Fielder vector) minimizes

$$\min_{x \neq 0} \sum_{(i,j) \in E} (x_i - x_j)^2 \quad \text{where} \quad \sum_i x_i = 0$$

- Intuitively, minimum when *x_i and x_j close whenever there is an edge between nodes i and j* in the graph.
- x must have some positive and some negative components

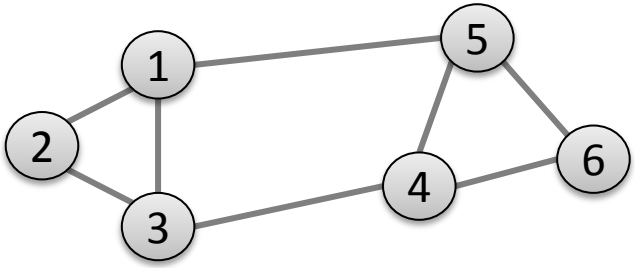
Cuts + eigenvalues: intuition

- A *partition* of the graph by taking:
  - one set to be the nodes i whose corresponding vector component x_i is *positive* and
  - the other set to be the nodes whose corresponding vector component is *negative*.

- The *cut* between the two sets will have a small number of edges because $(x_i - x_j)^2$ is likely to be smaller if both x_i and x_j have the same sign than if they have different signs.

- Thus, *minimizing* $x^T L x$ under the required constraints will end giving x_i and x_j the same sign if there is an edge (i, j).

Example



Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	−5	−1	−1	−1
	1	2	4	−2	1	0
	1	1	1	3	−1	1
	1	−1	−5	−1	1	1
	1	−2	4	−2	−1	0
	1	−1	1	3	1	−1

Other properties of L

Let G be an undirected graph with non-negative weights. Then

- the multiplicity *k* of the eigenvalue 0 of *L* equals the number of connected components $A_1, \ldots, A_k$ in the graph
- the eigenspace of eigenvalue 0 is spanned by the indicator vectors $1A_1, \ldots, 1A_k$ of those components

Proof (sketch)

If connected (k = 1)

$$0 = x^{\tau} L x = \sum_{(i,j) \in E} (x_i - x_j)^2$$

Assume k connected components, both A and L block diagonal, if we order vertices based on the connected component they belong to (recall the "tile" matrix)

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix}$$

L_i Laplacian of the i-th component

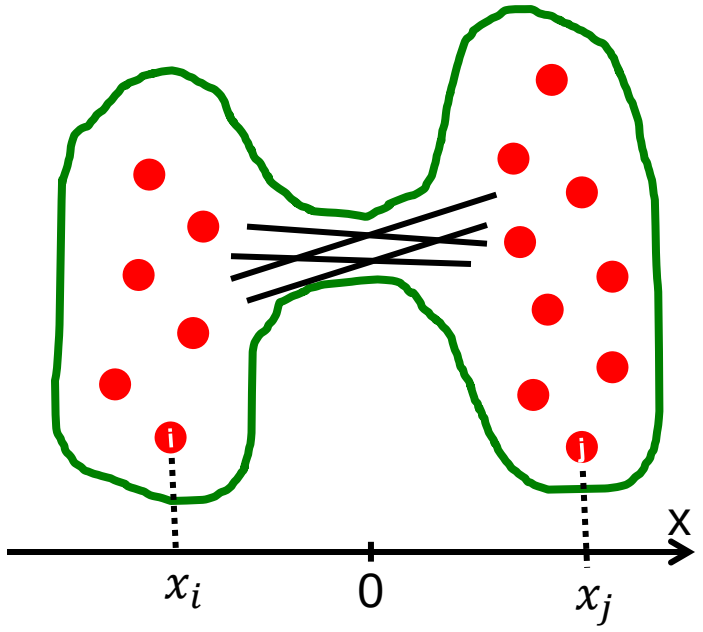
for all block diagonal matrices, that the spectrum is given by the union of the spectra of each block, and the corresponding eigenvectors are the eigenvectors of the block, filled with 0 at the positions of the other blocks.

Cuts + eigenvalues: summary

- What we know about x?
  - x is unit vector: $\sum_i x_i^2 = 1$
  - x is orthogonal to 1st eigenvector $(1, \ldots, 1)$ thus: $\sum_i x_i \cdot 1 = \sum_i x_i = 0$

$$\lambda_2 = \min_{\substack{\text{All labelings of nodes } i \text{ so that } \sum x_i = 0}} \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_i x_i^2}$$

We want to assign values x_i to nodes i such that few edges cross 0.
(we want x_i and x_j to subtract each other)



Balance to minimize

Spectral Clustering Algorithms

Three basic stages:

Pre-processing

- Construct a matrix representation of the graph

Decomposition

- Compute eigenvalues and eigenvectors of the matrix
- *Map each point to a lower-dimensional representation based on one or more eigenvectors*

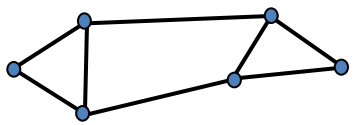
Grouping

- Assign points to two or more clusters, based on the new representation

Spectral Partitioning Algorithm

Pre-processing:

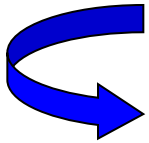
Build Laplacian matrix L of the graph



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

Decomposition:

- Find eigenvalues λ and eigenvectors x of the matrix L



$\lambda =$

0.0
1.0
3.0
3.0
4.0
5.0

$X =$

0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

- Map vertices to corresponding components of λ_2

1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

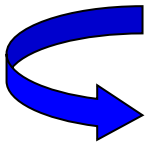
How do we now find the clusters?

Spectral Partitioning Algorithm

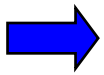
Grouping:

- Sort components of reduced 1-dimensional vector
- Identify clusters by splitting the sorted vector in two

- How to choose a splitting point?
  - Naïve approaches:
    - Split at 0 or median value
  - More expensive approaches:
    - Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)



1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6



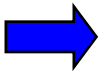
Split at 0:

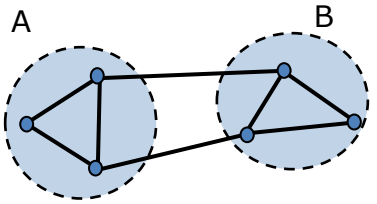
Cluster A: Positive points

Cluster B: Negative points

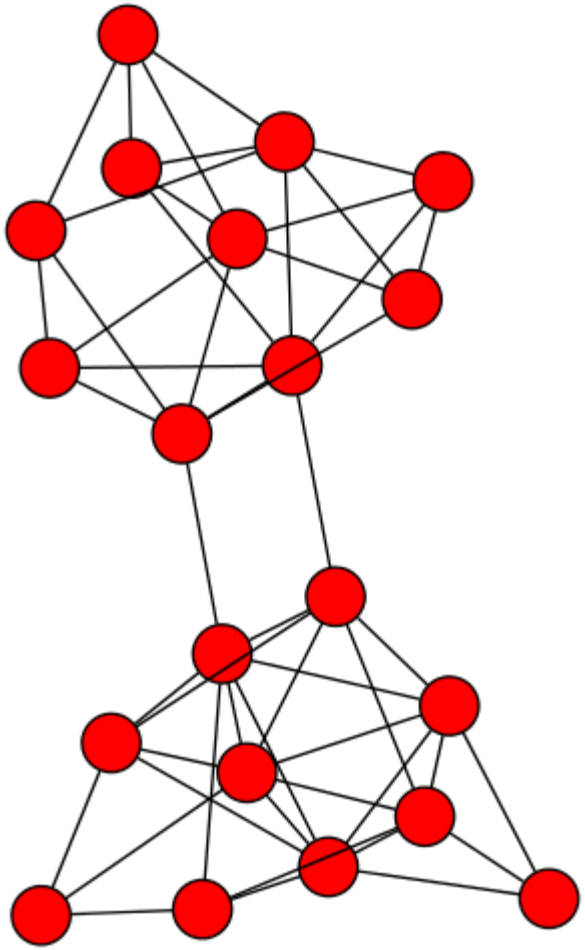
1	0.3
2	0.6
3	0.3

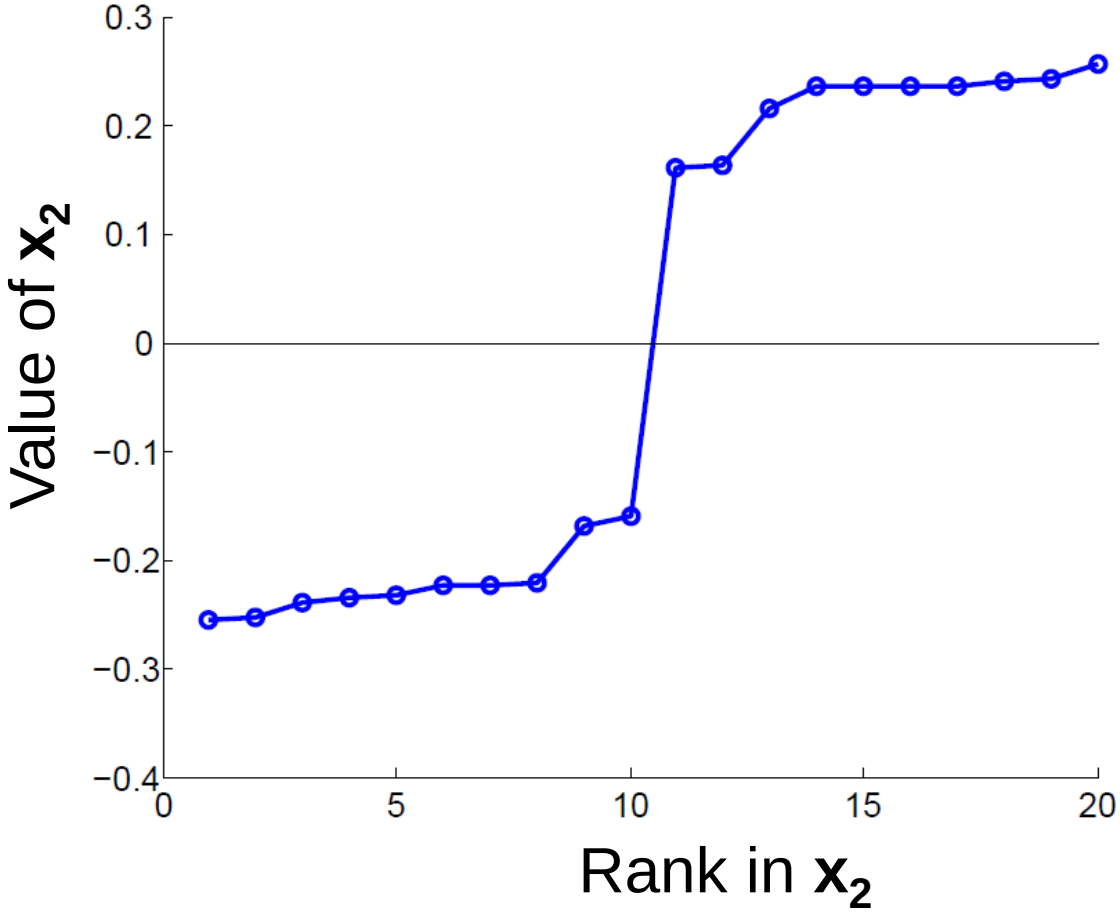
4	-0.3
5	-0.3
6	-0.6





Example: Spectral Partitioning

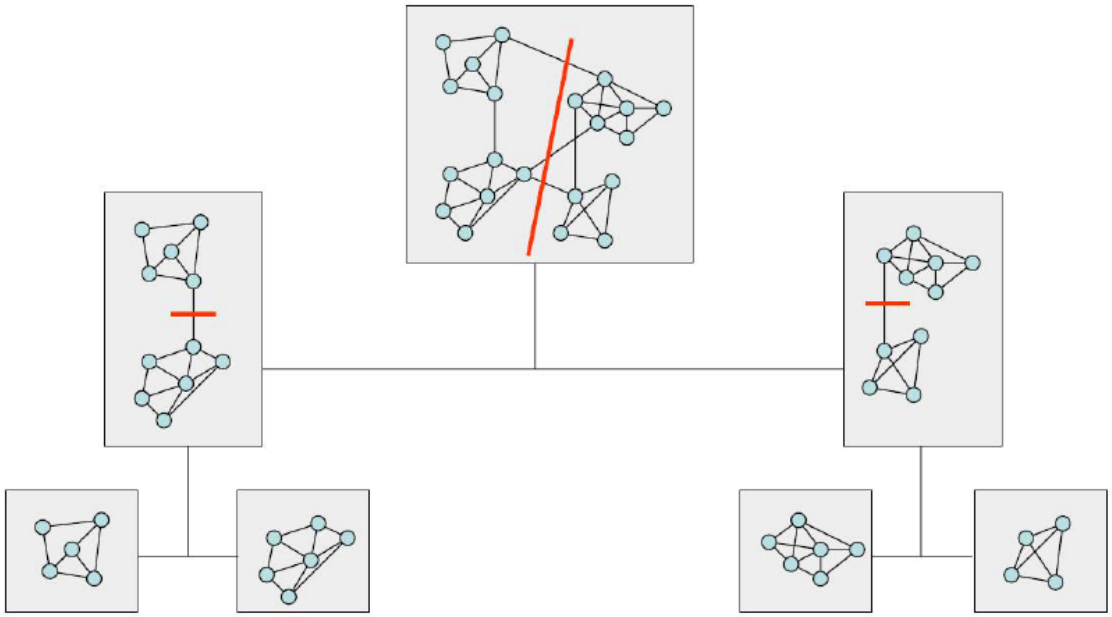




k-Way Spectral Clustering

How do we partition a graph into k clusters?

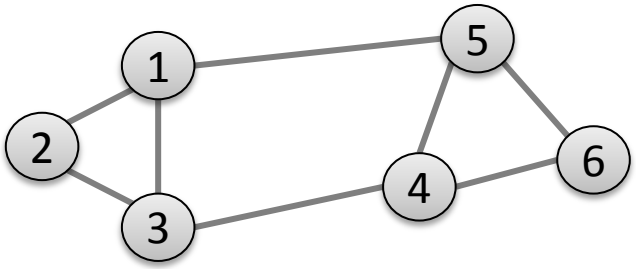
- *Recursively apply a bi-partitioning algorithm* in a hierarchical divisive manner
    - Disadvantages: Inefficient, unstable



k-Way Spectral Clustering

Use *several of the eigenvectors* to partition the graph.

If we use m eigenvectors, and set a threshold for each, we can get a partition into 2^m groups, each group consisting of the nodes that are above or below threshold for each of the eigenvectors, in a particular pattern.



Example

Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	−5	−1	−1	−1
	1	2	4	−2	1	0
	1	1	1	3	−1	1
	1	−1	−5	−1	1	1
	1	−2	4	−2	−1	0
	1	−1	1	3	1	−1

If we use both the 2nd and 3rd eigenvectors,
nodes 2 and 3 (negative in both)
5 and 6 (negative in 2nd, positive in 3rd)
1 and 4 alone

- Note that each eigenvector except the first is the vector x that minimizes $x^T Lx$, subject to the constraint that it is orthogonal to all previous eigenvectors.
- Thus, while each eigenvector tries to produce a minimum-sized cut, successive eigenvectors have to satisfy more and more constraints => the cuts progressively worse.

Spectral Clustering

- Use the lowest *k* eigenvalues of L to construct the *n*x*k* graph G' that has these eigenvectors as columns
- *The n-rows represent the graph vertices in a k-dimensional Euclidean space*
- Group these vertices in *k* clusters using k-means clustering or similar techniques

Spectral clustering (besides graphs)

Can be used to cluster any points (not just vertices), as long as an appropriate similarity matrix

Needs to be symmetric and non-negative

How to construct a graph:

- ε-neighborhood graph: connect all points whose pairwise distances are smaller than ε
- k-nearest neighbor graph: connect each point with each k nearest neigbhor
- full graph: connect all points with weight in the edge (i, j) equal to the similarity of i and j

Summary

- The values of x minimize

$$\min_{x \neq 0} \sum_{(i,j) \in E} (x_i - x_j)^2 \qquad \sum_i x_i = 0$$

- For weighted matrices

$$\min_{x \neq 0} \sum_{(i,j)} A[i,j](x_i - x_j)^2 \qquad \sum_i x_i = 0$$

- The ordering according to the x_i values will group similar (connected) nodes together

- Physical interpretation: The stable state of springs placed on the edges of the graph

Normalized Graph Laplacians

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

$$L_{rw} = D^{-1} L = I - D^{-1} W$$

L_{rw} closely connected to random walks (to be discussed in future lectures)

$$x^{\tau} L_{sym} x = \sum_{(i,j) \in E} \left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}} \right)^2$$

Cuts and spectral clustering

$$\mathrm{cut}(A_1, \ldots, A_k) := \sum_{i=1}^{k} \mathrm{cut}(A_i, \overline{A}_i)$$

$$\mathrm{RatioCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, \overline{A}_i)}{|A_i|}$$

$$\mathrm{Ncut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, \overline{A}_i)}{\mathrm{vol}(A_i)}.$$

Relaxing Ncut leads to normalized spectral clustering, while relaxing RatioCut leads to unnormalized spectral clustering

Finding an Optimal Cut (sketch)

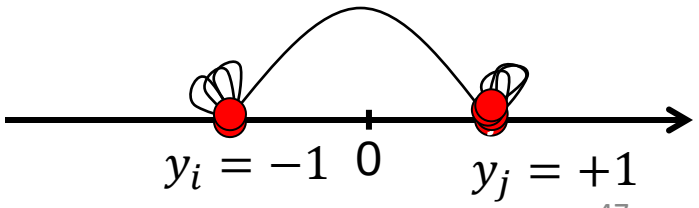
- Express partition (A,B) as a vector

$$y_i = \begin{cases} +1 & if\ i \in A \\ -1 & if\ i \in B \end{cases}$$

- We can minimize the cut of the partition by finding a non-trivial vector x that minimizes:

$$\arg\min_{y \in [-1,+1]^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2$$

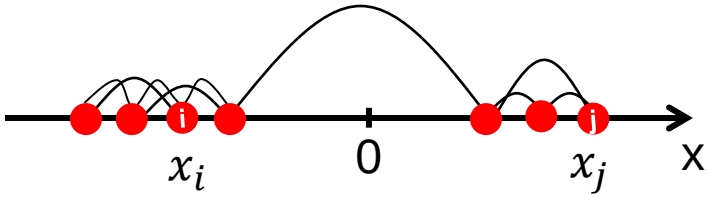
Can not solve exactly. Let us relax y and allow it to take any real value (instead of two)



Finding an Optimal Cut (sketch)

Rayleigh Theorem

$$\min_{y \in \Re^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2 = y^T L y$$



- $\lambda_2 = \min_y f(y)$: The minimum value of $f(y)$ is given by the 2nd smallest eigenvalue λ_2 of the Laplacian matrix L
- $\mathrm{x} = \arg\min_{\mathrm{y}} f(y)$: The optimal solution for y is given by the corresponding eigenvector x, referred as the Fiedler vector

Finding an Optimal Cut (sketch)

Need to re-transform the real-valued solution vector f of the relaxed problem into a discrete indicator vector. Simplest way, use the sign

$$\begin{cases} v_i \in A & \text{if } f_i \geq 0 \\ v_i \in \overline{A} & \text{if } f_i < 0. \end{cases}$$

Consider the coordinates f_i as points in R and cluster them into two groups C by the k-means clustering algorithm.

$$\begin{cases} v_i \in A & \text{if } f_i \in C \\ v_i \in \overline{A} & \text{if } f_i \in \overline{C} \end{cases}$$

Spectral partition

- Partition the nodes according to the ordering induced by the Fielder vector
- If $u = (u_1, u_2, \ldots, u_n)$ is the Fielder vector, then split nodes according to a threshold value s
  - bisection: s is the median value in u
  - ratio cut: s is the value that minimizes α
  - sign: separate positive and negative values ($s=0$)
  - gap: separate according to the largest gap in the values of u
- This works well (provably for special cases)

Fielder Value

Suppose there is a partition of G into A and B where $|A| \leq |B|$, s.t. $\alpha = \frac{(\#\ edges\ from\ A\ to\ B)}{|A|}$

- The value λ_2 is a good approximation of the graph expansion

$$\frac{\alpha^2}{2d_{max}} \leq \lambda_2 \leq 2\alpha$$

d_{max} = maximum degree

$$\frac{\lambda_2}{2} \leq \alpha \leq \sqrt{\lambda_2(2d_{max} - \lambda_2)}$$

- For the minimum ratio cut of the Fielder vector we have that

$$\frac{\alpha^2}{2d_{max}} \leq \lambda_2 \leq 2\alpha$$

- If the max degree d_{max} is bounded we obtain a good approximation of the minimum expansion cut

Approx. Guarantee of Spectral (proof)

- Suppose there is a partition of G into A and B where $|A| \leq |B|$, s.t. $\alpha = \frac{(\#\ edges\ from\ A\ to\ B)}{|A|}$ then $2\alpha \geq \lambda_2$
  - This is the approximation guarantee of the spectral clustering. It says the cut spectral finds is at most 2 away from the optimal one of score α.
- Proof:
  - Let: a=|A|, b=|B| and e= # edges from A to B
  - Enough to choose some x_i based on A and B such that: $\lambda_2 \leq \underbrace{\frac{\sum(x_i - x_j)^2}{\sum_i x_i^2}} \leq 2\alpha$ (while also $\sum_i x_i = 0$)

λ_2 **is only smaller**

Approx. Guarantee of Spectral

- Proof (continued):

(1) Set: $x_i = \begin{cases} -\frac{1}{a} & if\ i \in A \\ +\frac{1}{b} & if\ i \in B \end{cases}$

- Let's quickly verify that $\sum_i x_i = 0$: $a\left(-\frac{1}{a}\right) + b\left(\frac{1}{b}\right) = 0$

(2) Then: $\frac{\sum(x_i - x_j)^2}{\sum_i x_i^2} = \frac{\sum_{i\in A, j\in B}\left(\frac{1}{b}+\frac{1}{a}\right)^2}{a\left(-\frac{1}{a}\right)^2 + b\left(\frac{1}{b}\right)^2} = \frac{e\cdot\left(\frac{1}{a}+\frac{1}{b}\right)^2}{\frac{1}{a}+\frac{1}{b}} =$

$e\left(\frac{1}{a}+\frac{1}{b}\right) \le e\left(\frac{1}{a}+\frac{1}{a}\right) \le e\frac{2}{a} = 2\alpha$

Which proves that the cost achieved by spectral is better than twice the OPT cost

e … number of edges between A and B

Approx. Guarantee of Spectral

- Putting it all together:

$$2\alpha \geq \lambda_2 \geq \frac{\alpha^2}{2d_{max}}$$

  - where d_{max} is the maximum node degree in the graph
    - Note we only provide the 1st part: $2\alpha \geq \lambda_2$
    - We did not prove $\lambda_2 \geq \frac{\alpha^2}{2d_{max}}$
  - Overall this always certifies that λ_2 always gives a useful bound

Thanks to Aris Gionis

MAXIMUM DENSEST SUBGRAPH

Finding dense subgraphs

- Dense subgraph: A collection of vertices such that there are a lot of edges between them
  - E.g., find the subset of email users that talk the most between them
  - Or, find the subset of genes that are most commonly expressed together
- Similar to community identification but we do not require that the dense subgraph is sparsely connected with the rest of the graph.

Definitions

- Input: undirected graph $G = (V, E)$.
- Degree of node u: $\deg(u)$
- For two sets $S \subseteq V$ and $T \subseteq V$:

$$E(S, T) = \{(\mathrm{u}, \mathrm{v}) \in E : u \in S, v \in T\}$$

- $E(S) = E(S, S)$: edges within nodes in S
- Graph Cut defined by nodes in $S \subseteq V$:

$E(S, \bar{S})$: edges between S and the rest of the graph

- Induced Subgraph by set S : $G_S = (S, E(S))$

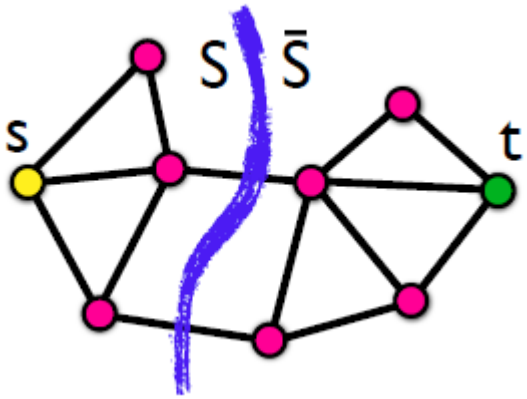
Definitions

- How do we define the density of a subgraph?
- Average Degree:

$$d(S) = \frac{2|E(S)|}{|S|}$$

- Problem: Given graph G, find subset S, that maximizes density d(S)
  - Surprisingly there is a polynomial-time algorithm for this problem.

Min-Cut Problem



Given a graph* $G = (V, E)$,
A source vertex $s \in V$,
A destination vertex $t \in V$

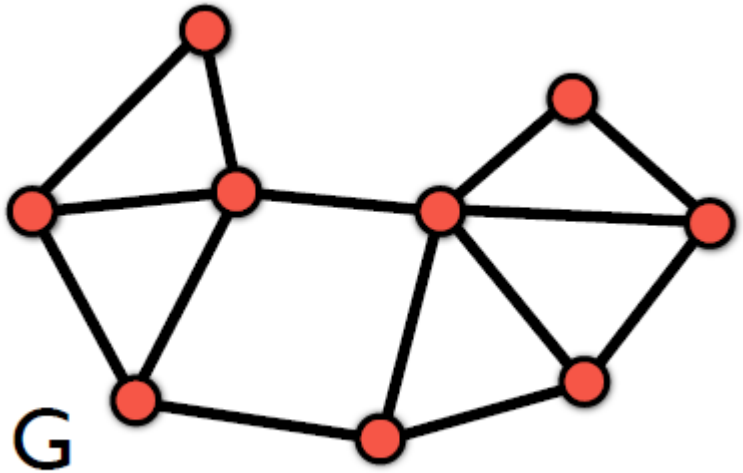
Find a set $S \subseteq V$
Such that $s \in S$ and $t \in \bar{S}$
That minimizes $E(S, \bar{S})$

* The graph may be weighted

Min-Cut = Max-Flow: the minimum cut maximizes the flow that can be sent from s to t. There is a polynomial time solution.

Decision problem

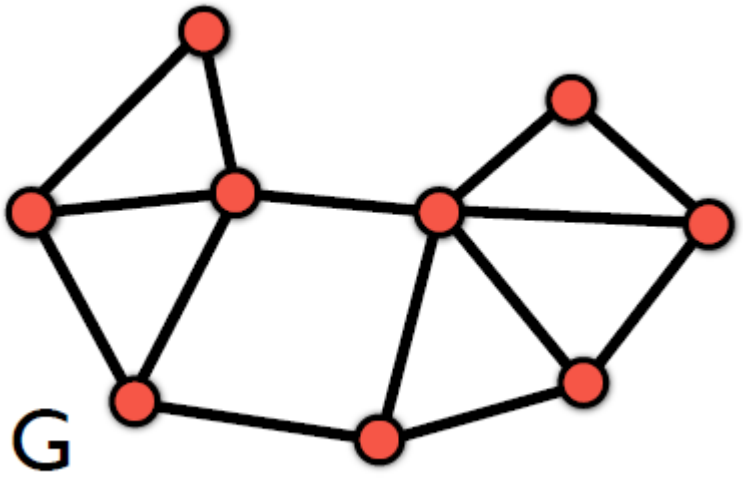
- Consider the decision problem
  - Is there a set S with $d(S) \geq c$?

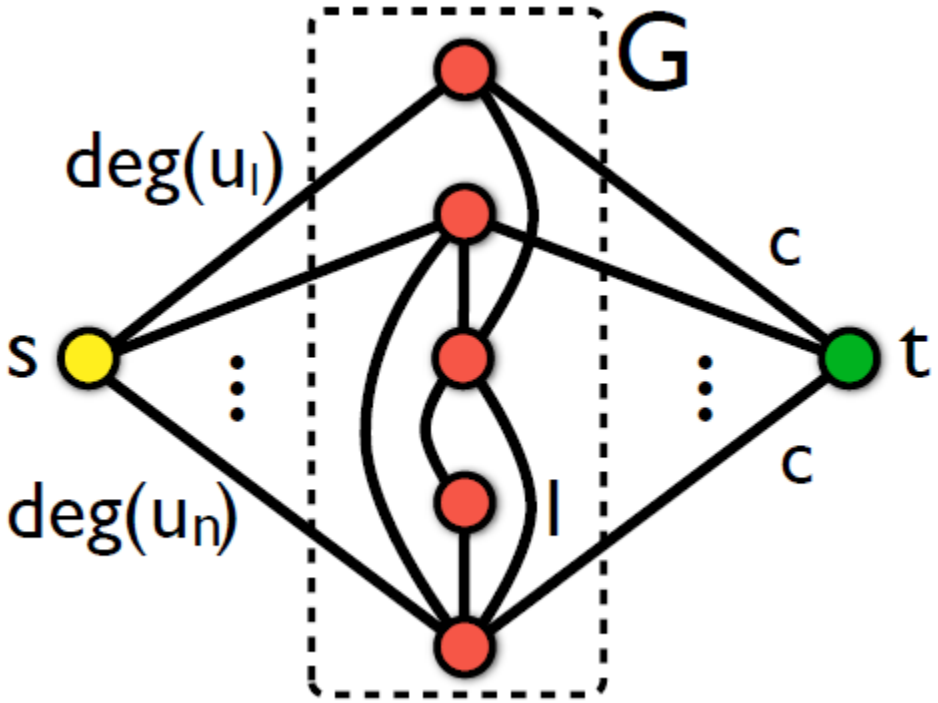


- $d(S) \geq c$
- $2|E(S)| \geq c|S|$
- $\sum_{v \in S} \deg(v) - E(S, \bar{S}) \geq c|S|$
- $2|E| - \sum_{v \in \bar{S}} \deg(v) - E(S, \bar{S}) \geq c|S|$
- $\sum_{v \in \bar{S}} \deg(v) + E(S, \bar{S}) + c|S| \leq 2|E|$

Transform to min-cut

- For a value c we do the following transformation

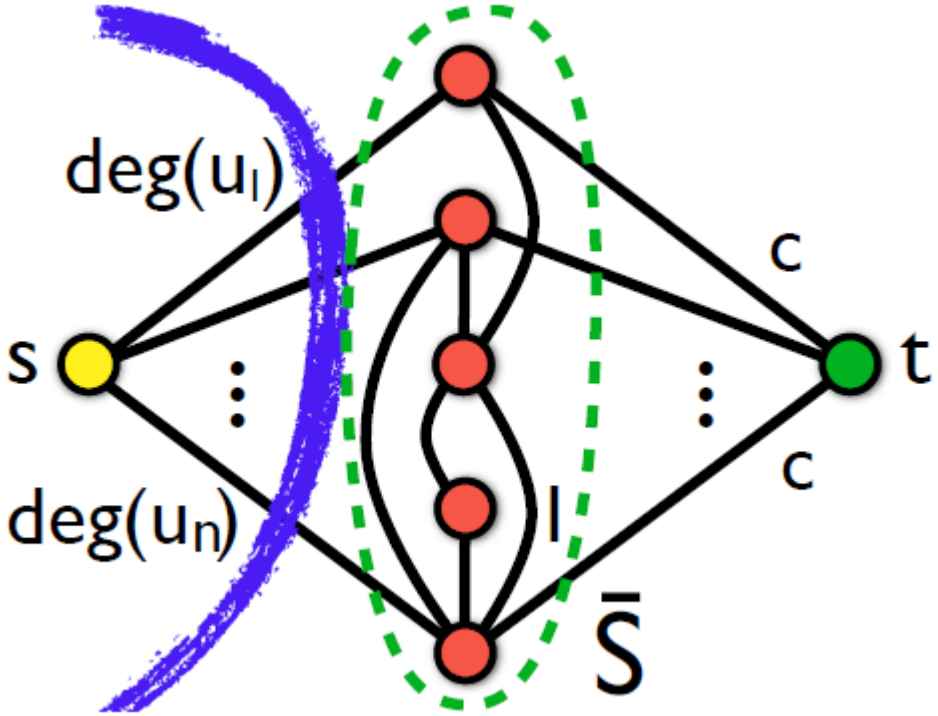




- We ask for a min s-t cut in the new graph

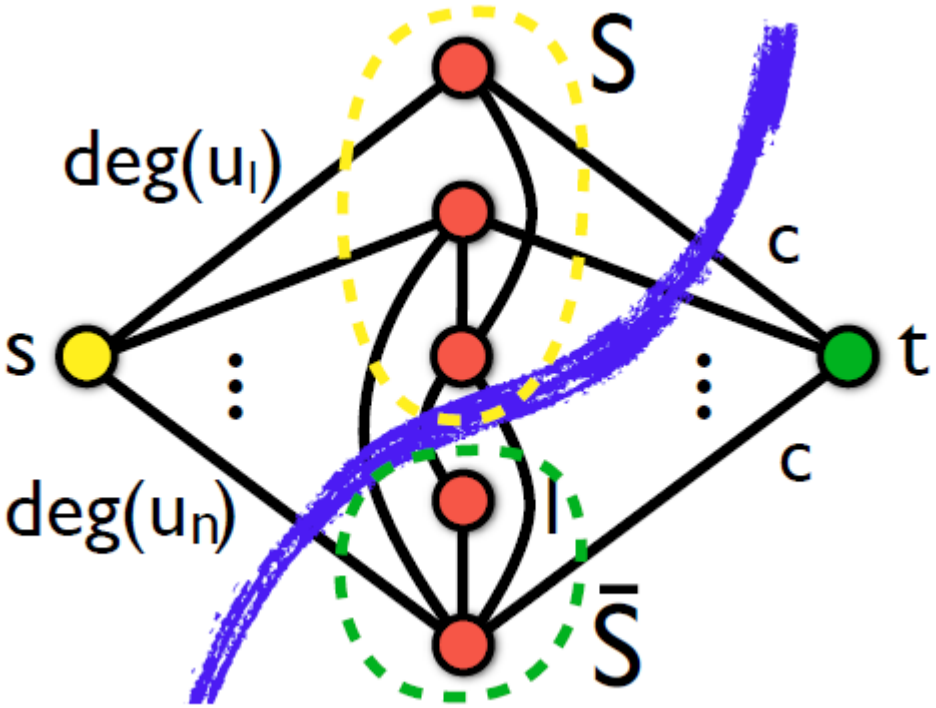
Transformation to min-cut

- There is a cut that has value $2|E|$



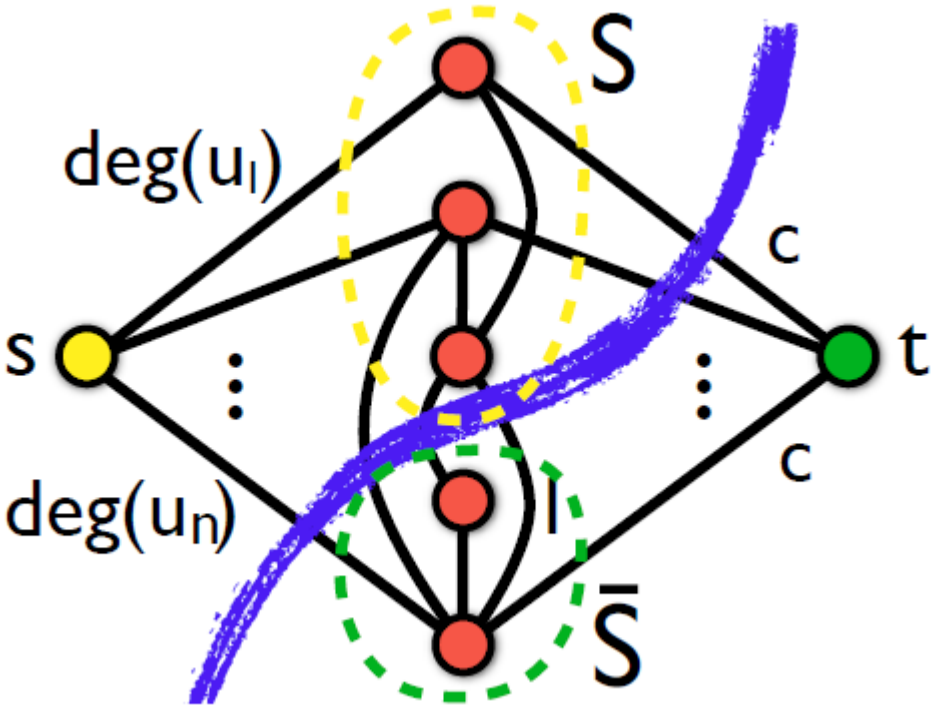
Transformation to min-cut

- Every other cut has value:
- $\sum_{v \in \bar{S}} \deg(v) + E(S, \bar{S}) + c|S|$



Transformation to min-cut

- If $\sum_{v\in\bar{S}} \deg(v) + E(S,\bar{S}) + c|S| \leq 2|E|$ then $S \neq \emptyset$ and $d(S) \geq c$



Algorithm (Goldberg)

Given the input graph G, and value c

1. Create the min-cut instance graph
2. Compute the min-cut
3. If the set S is not empty, return YES
4. Else return NO

How do we find the set with maximum density?

Min-cut algorithm

- The min-cut algorithm finds the optimal solution in polynomial time O(nm), but this is too expensive for real networks.
- We will now describe a simpler approximation algorithm that is very fast
  - Approximation algorithm: the ratio of the density of the set produced by our algorithm and that of the optimal is bounded.
    - We will show that the ratio is at most ½
    - The optimal set is at most twice as dense as that of the approximation algorithm.

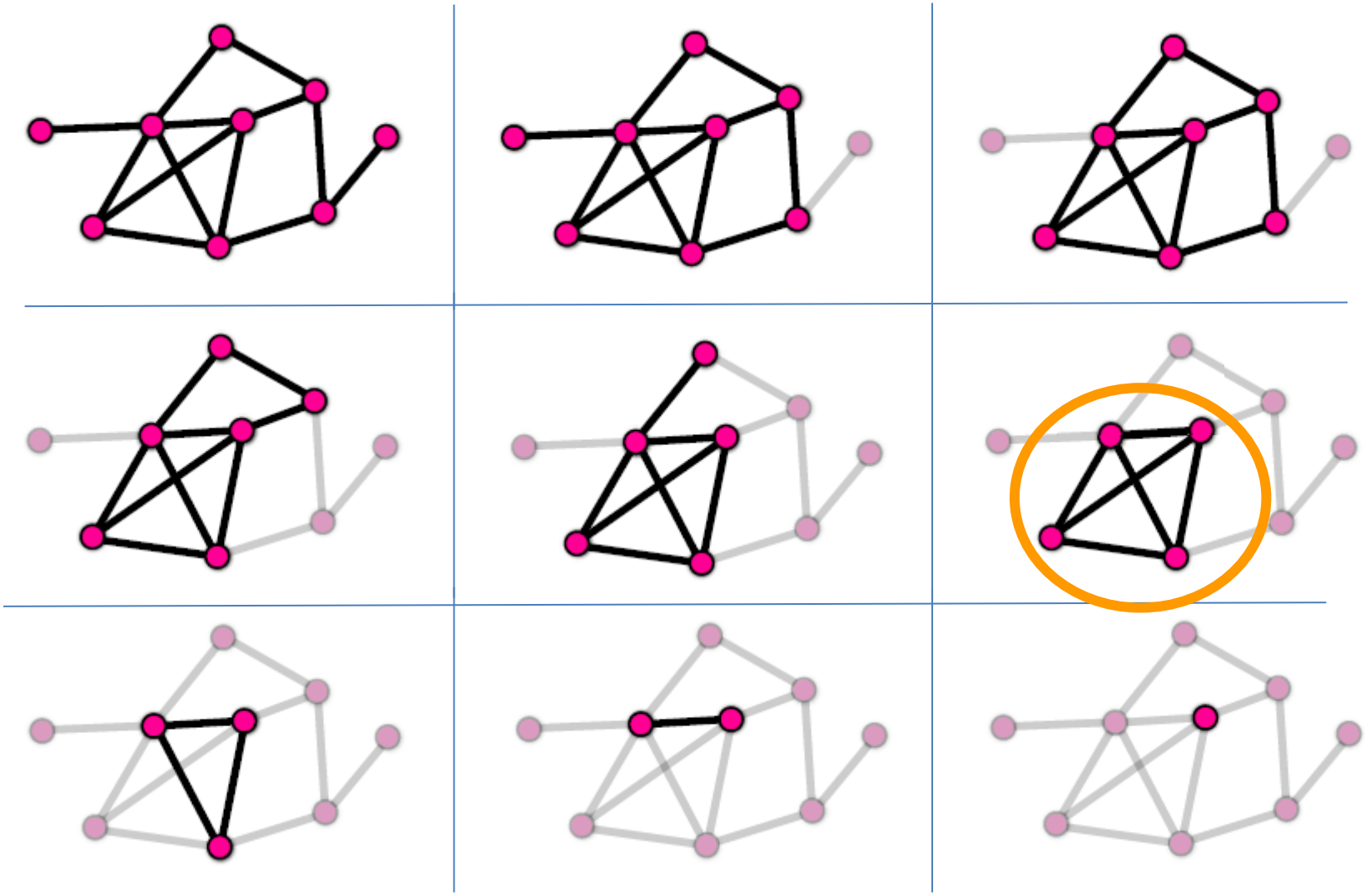
- Any ideas for the algorithm?

Greedy Algorithm

Given the graph $G = (V, E)$

1. $S_0 = V$
2. For $i = 1 \ldots |V|$
   a. Find node $v \in S$ with the minimum degree
   b. $S_i = S_{i-1} \setminus \{v\}$
3. Output the densest set S_i

Example



Analysis

- We will prove that the optimal set has density at most 2 times that of the set produced by the Greedy algorithm.

- Density of optimal set: $d_{opt} = \max_{S \subseteq V} d(S)$
- Density of greedy algorithm d_g

- We want to show that $d_{opt} \leq 2 \cdot d_g$

Upper bound

- We will first upper-bound the solution of optimal
- Assume an arbitrary assignment of an edge (u, v) to either u or v

- Define:
  - $IN(u)$= # edges assigned to u
  - $\Delta = \max_{u \in V} IN(u)$
- We can prove that
  - $d_{opt} \leq 2 \cdot \Delta$

This is true for any assignment of the edges!

Lower bound

- We will now prove a lower bound for the density of the set produced by the greedy algorithm.
- For the lower bound we consider a specific assignment of the edges that we create as the greedy algorithm progresses:
  - When removing node u from S, assign all the edges to u
- So: $IN(u) = \text{degree of } u \text{ in } S \leq d(S) \leq d_g$
- This is true for all u so $\Delta \leq d_g$

- It follows that $d_{opt} \leq 2 \cdot d_g$

The k-densest subgraph

- The k-densest subgraph problem: Find the set of k nodes S, such that the density $d(S)$ is maximized.
  - The k-densest subgraph problem is NP-hard!

QUANTIFYING SOCIAL GROUP EVOLUTION

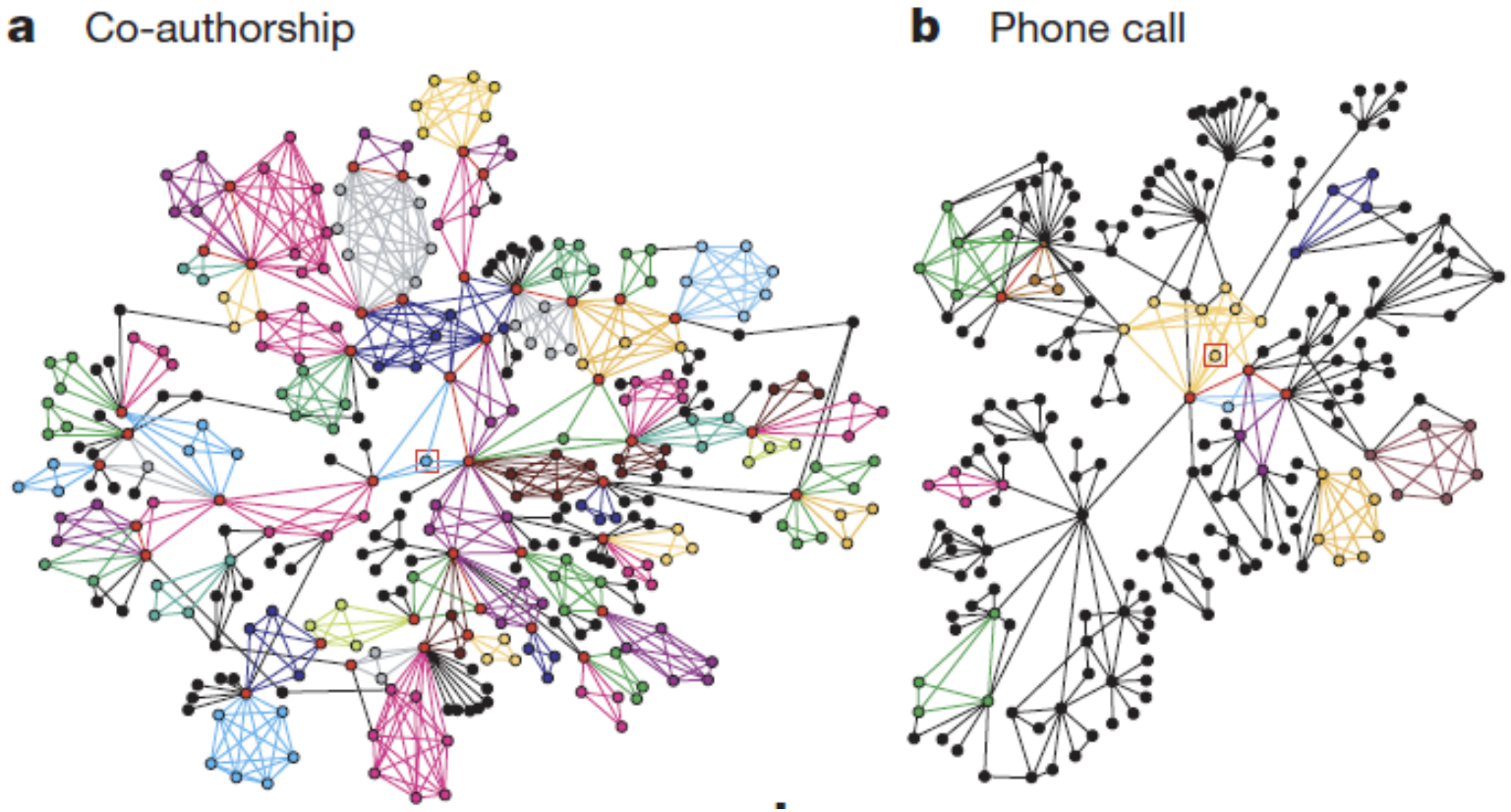
G Palla, AL Barabási, T Vicsek*, Nature* 446 (7136), 664-667

Datasets

- monthly *list of articles* in the Cornell University Library e-print condensed matter (cond-mat) archive spanning *142 months*, with over *30,000 authors*,

- *phone calls* between the customers of a mobile phone company spanning *52 weeks* (accumulated over two-week-long periods) containing the communication patterns of over *4 million users.*

Datasets



black nodes/edges do not belong to any community,
red nodes belong to two or more communities are shown in red

Datasets

Different local structure:

- Co-authorship: dense network with significant overlap among communities (co-authors of an article form cliques) -- Phone-call: communities less interconnected, often separated by one or more inter-community node/edge
- Phone-call: the links correspond to instant communication events, whereas in co-authorship long-term collaborations.

Fundamental differences suggest that any common features represent potentially generic characteristics

Approach

- Communities at each time step extracted using the clique percolation method (CPM)

- Why CPM?

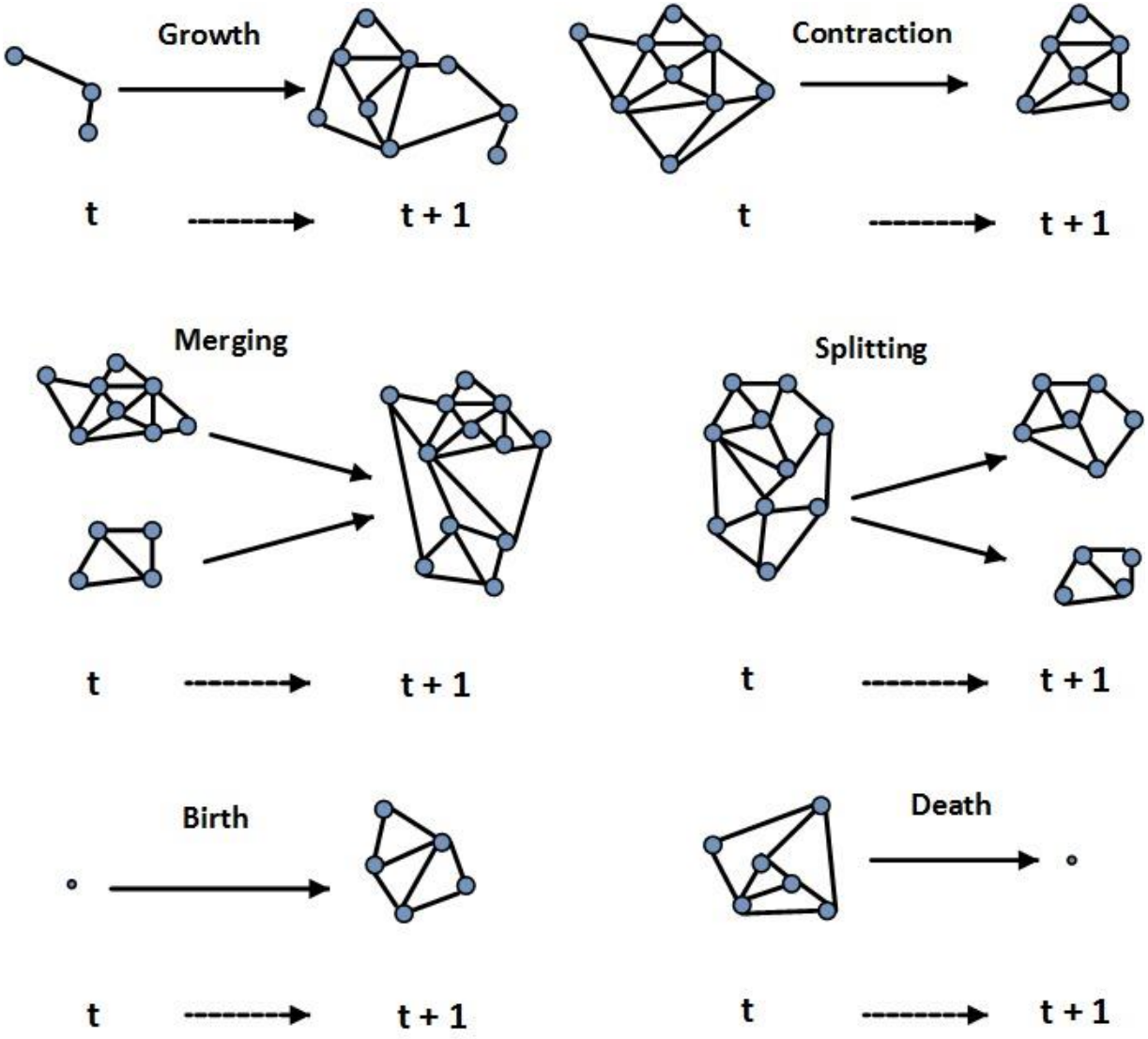
their members can be reached through well connected subsets of nodes, and communities may overlap

- Parameters

k = 4

Weighted graph – use a weight threshold w* (links weaker than w* are ignored)

Basic Events



Identifying Events

For each pair of consecutive time steps t and t+1, construct a joint graph consisting of the union of links from the corresponding two networks, and extract the CPM community structure of this joint network

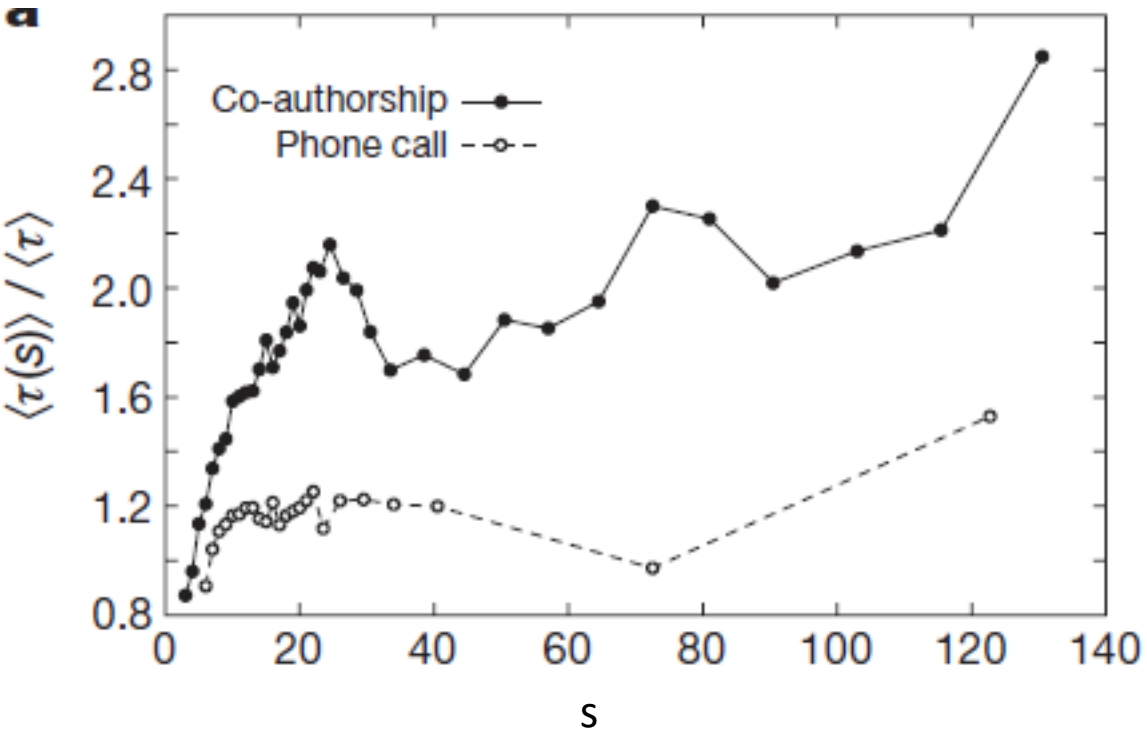
- Any community from either the t or the t+1 snapshot is contained in exactly one community in the joint graph
- If a community in the joint graph contains a single community from t and a single community from t+1, then they are matched.
- If the joint group contains more than one community from either time steps, the communities are matched in descending order of their relative node overlap

Results

s: size

t: age

s and t are positively correlated: larger communities are on average older

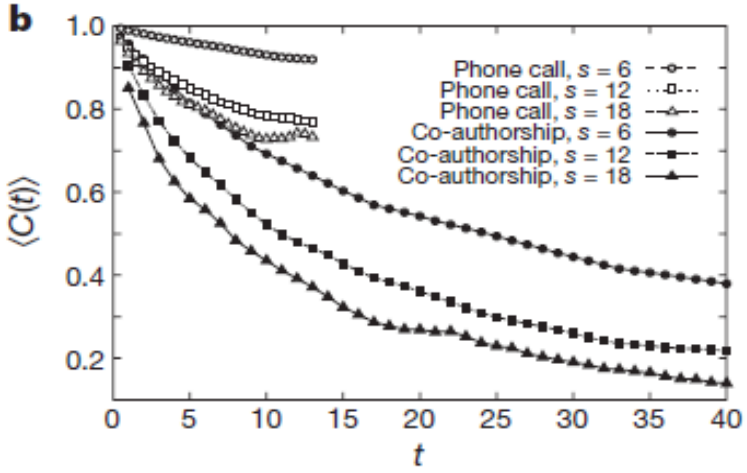


Results

Auto-correlation function

$$C(t) \equiv \frac{|A(t_0) \cap A(t_0 + t)|}{|A(t_0) \cup A(t_0 + t)|}$$

where A(t) members of community A at t



- the collaboration network is more “dynamic” (decays faster)
- in both networks, the auto-correlation function decays faster for the larger communities, showing that the membership of the larger communities is changing at a higher rate.

Results

$$\zeta \equiv \frac{\sum_{t=t_0}^{t_{\max}-1} C(t,t+1)}{t_{\max}-t_0-1}$$

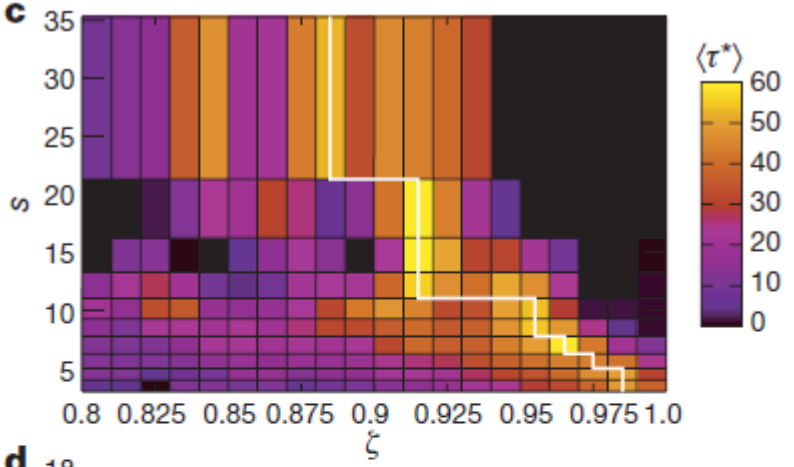
1-ζ: the average ratio of members changed in one step

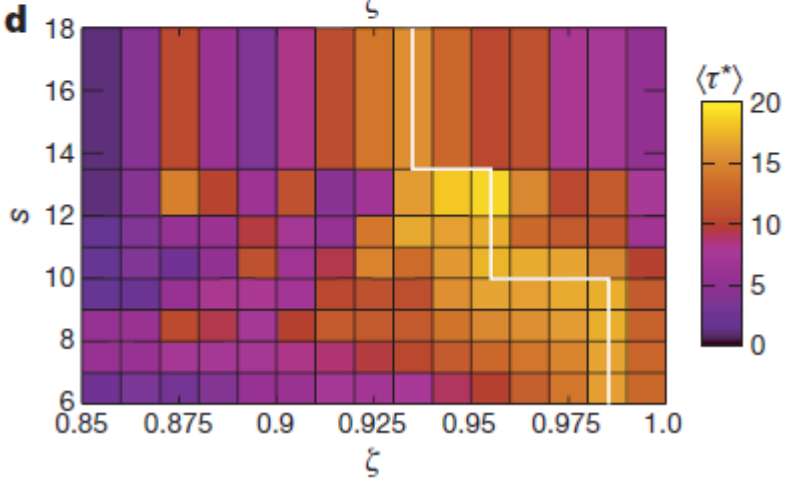
τ*: lifetime, stationarity ζ

the average life-span <t*> (colour coded) as a function of ζ and s

- for *small communities* optimal ζ near 1, better to have static, time-independent
- For *large communities*, the peak is shifted towards low f values, better to have acontinually changing membership

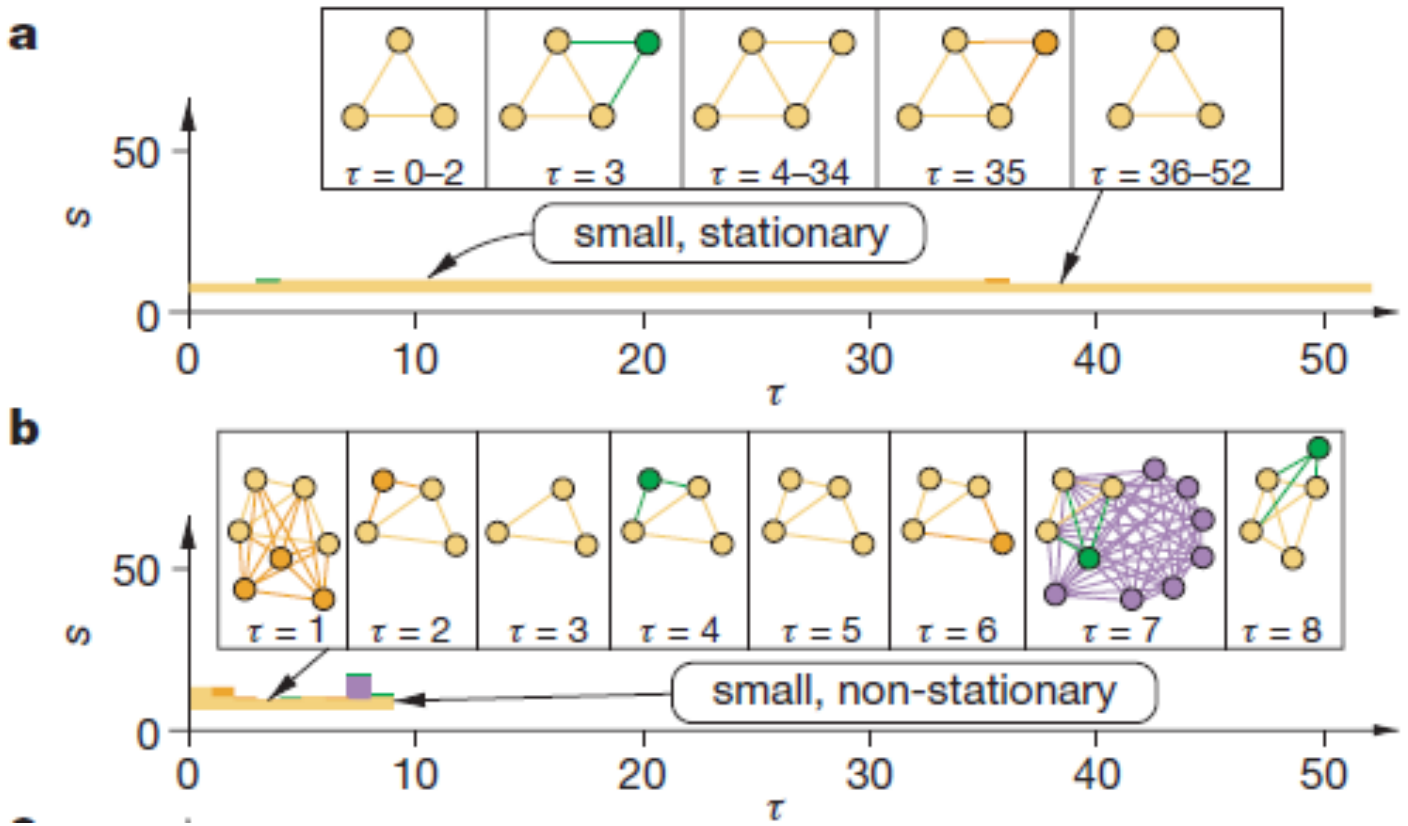
co-authorship



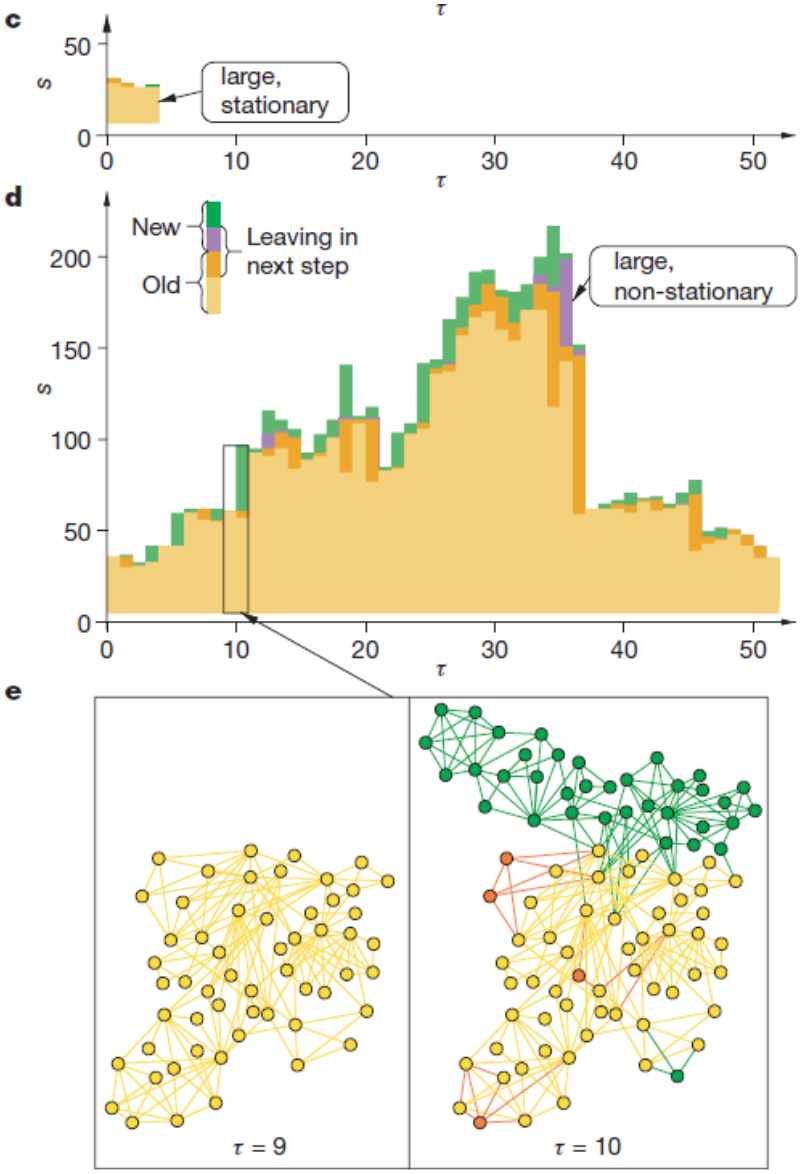


phone-call

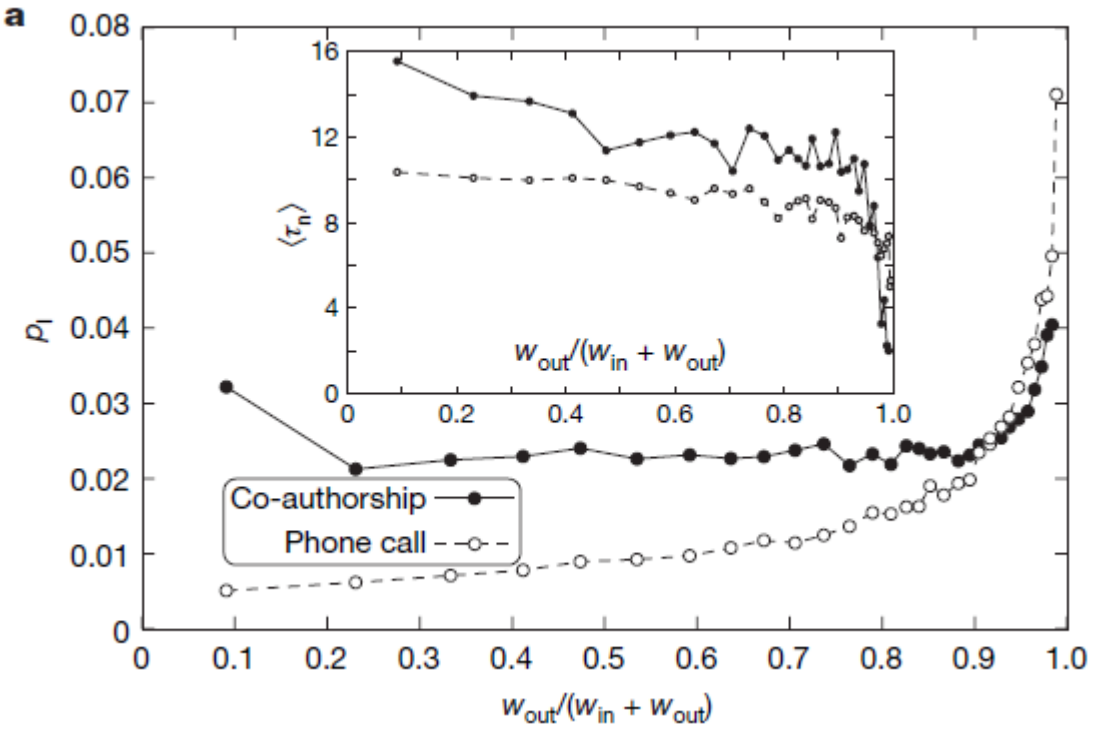
Results


a
50
s
0
τ = 0–2
τ = 3
τ = 4–34
τ = 35
τ = 36–52
small, stationary
0
10
20
30
40
50
τ
b
50
s
0
τ = 1
τ = 2
τ = 3
τ = 4
τ = 5
τ = 6
τ = 7
τ = 8
small, non-stationary
0
10
20
30
40
50
τ

Results

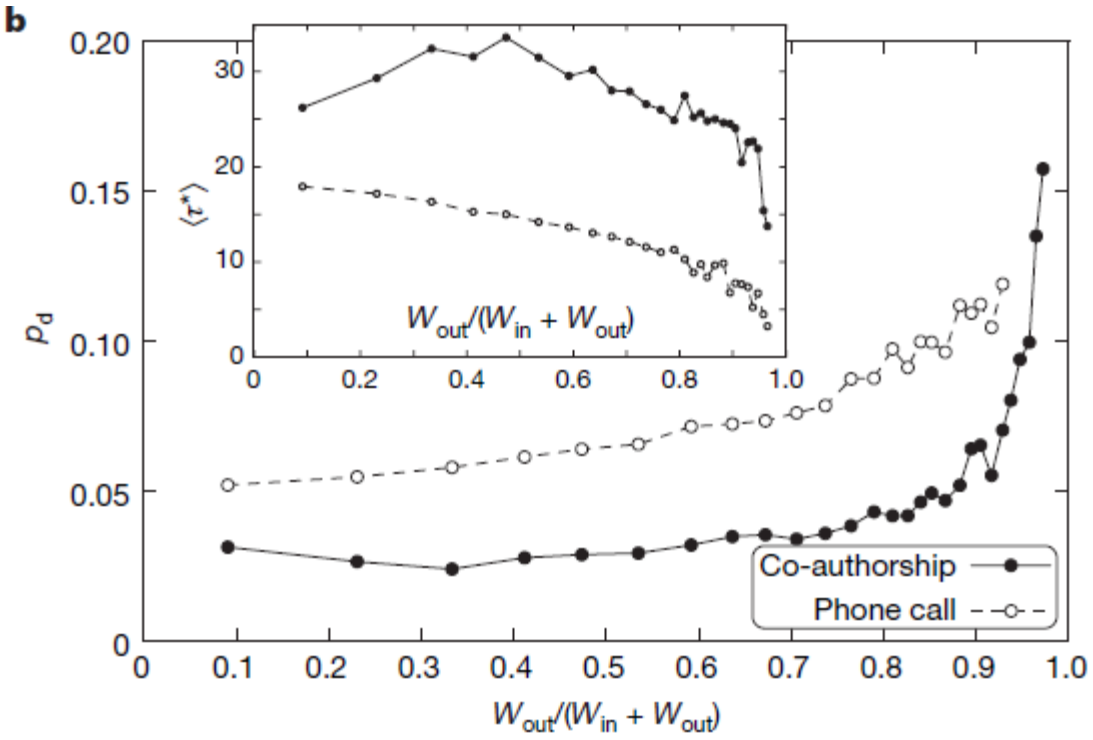

c
d
e
s
τ
large,
stationary
New
Old
Leaving in
next step
large,
non-stationary
τ = 9
τ = 10

Can we predict the evolution?



w_{out}: individual commitment to outside the community
w_{in}: individual commitment inside the community
p: probability to abandon the community

Can we predict the evolution?



W_{out}: total weight of links to nodes outside the community
W_{in}: total weight of links inside the community
p: probability of a community to disintegrate in the next step
for co-authorship max lifetime at intermediate values

Conclusions

Significant *difference between smaller collaborative or friendship circles and institutions*.

- At the heart of small communities are a few strong relationships, and as long as these persist, the community around them is stable.
- The condition for stability of large communities is continuous change, so that after some time practically all members are exchanged.
- Loose, rapidly changing communities reminiscent of institutions, which can continue to exist even after all members have been replaced by new members (e.g., members of a school).

Basic References

- Jure Leskovec, Anand Rajaraman, Jeff Ullman, Mining of Massive Datasets, Chapter 10, http://www.mmds.org/
- Reza Zafarani, Mohammad Ali Abbasi, Huan Liu, Social Media Mining: An Introduction, Chapter 6, http://dmml.asu.edu/smm/
- Santo Fortunato: Community detection in graphs. CoRR abs/0906.0612v2 (2010)
- Ulrike von Luxburg: A Tutorial on Spectral Clustering. CoRR abs/0711.0189 (2007)
- G Palla, A. L. Barabási, T Vicsek, Quantyfying Social Group Evolution. *Nature* 446 (7136), 664-667

Questions?