Spectral Clustering

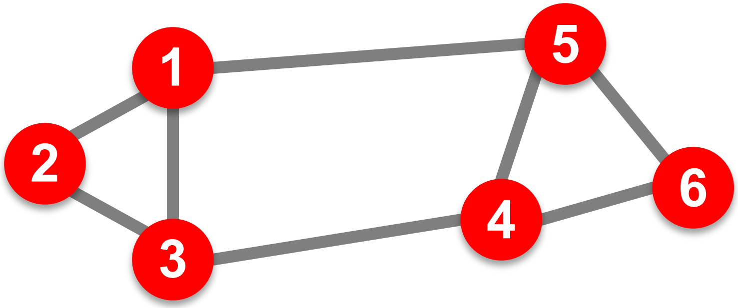
CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

Spectral Clustering Algorithms

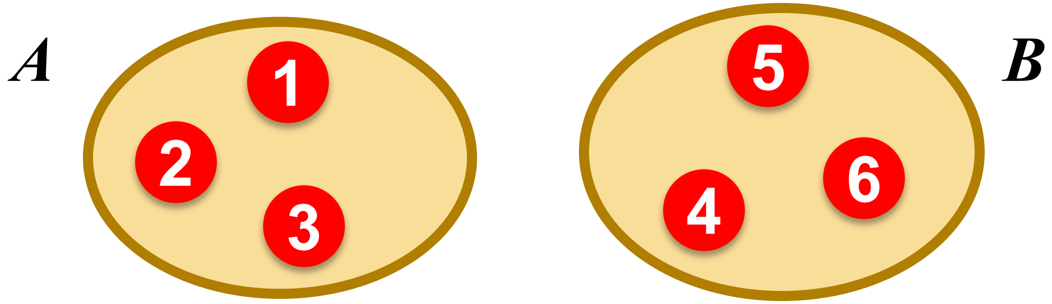
- Three basic stages:
  - 1) Pre-processing
    - Construct a matrix representation of the graph
  - 2) Decomposition
    - Compute eigenvalues and eigenvectors of the matrix
    - Map each point to a lower-dimensional representation based on one or more eigenvectors
  - 3) Grouping
    - Assign points to two or more clusters, based on the new representation
- But first, let's define the problem

Graph Partitioning

- **Undirected graph $G(V, E)$:**



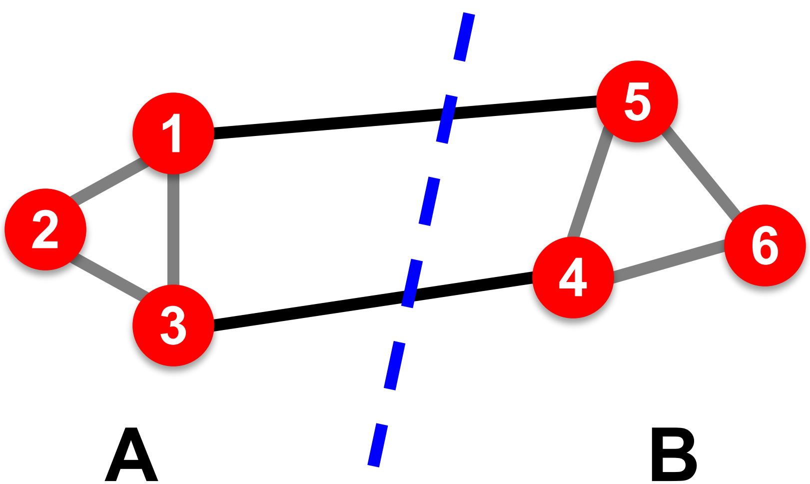
- **Bi-partitioning task:**
  - Divide vertices into two disjoint groups A, B



- **Questions:**
  - How can we define a "good" partition of 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

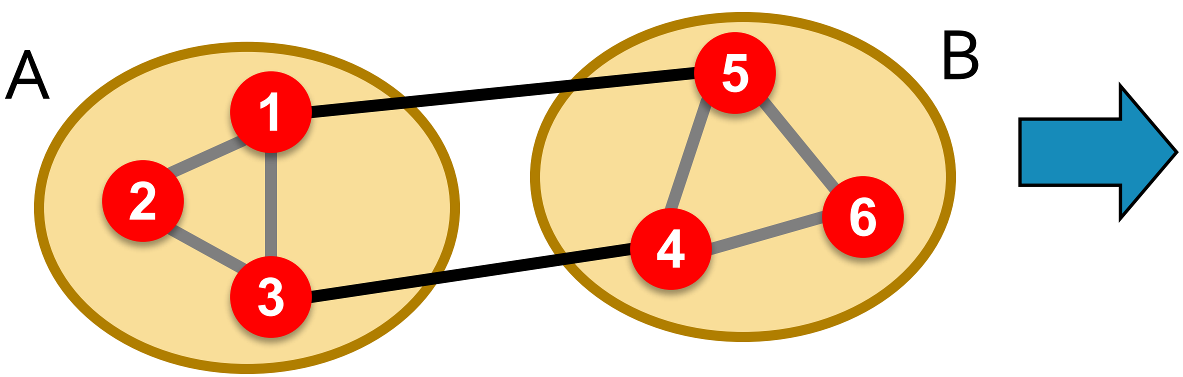


Graph Cuts

- **Express partitioning objectives as a function of the "edge cut" of the partition**
- **Cut:** Set of edges with one endpoint in each group:

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

If the graph is weighted w_{ij} is the weight, otherwise, all $w_{ij} \in \{0,1\}$



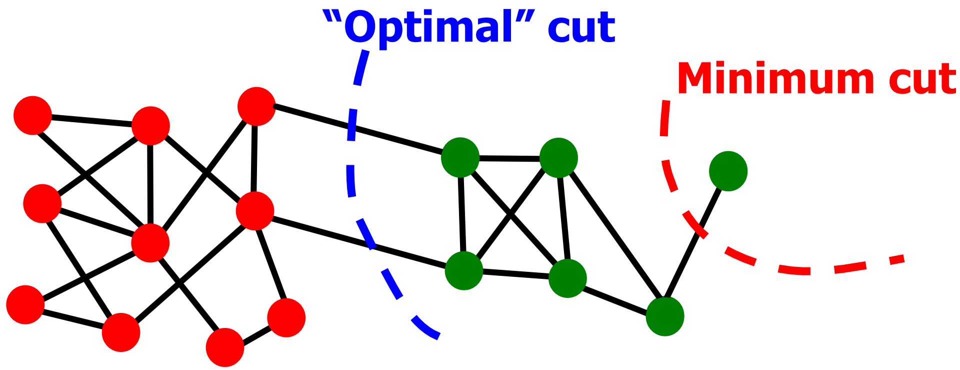
$$cut(A,B) = 2$$

Graph Cut Criterion

- **Criterion: Minimum-cut**
  - Minimize weight of connections between groups

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

- **Degenerate case:**



- **Problem:**
  - Only considers external cluster connections
  - Does not consider internal cluster connectivity

Graph Cut Criterion

- **Criterion: Conductance** [Shi-Malik, '97]
  - Connectivity between groups relative to the density of each group

$$\phi(A,B) = \frac{cut(A,B)}{\min(vol(A), vol(B))}$$

$\boldsymbol{vol(A)}$: total weighted degree of the nodes in $\boldsymbol{A}$:
$\boldsymbol{vol(A)} = \sum_{i \in A} \boldsymbol{k_i}$ (number of edge end points in $\boldsymbol{A}$)

- **Why use this criterion?**
  - Produces more balanced partitions
- **How do we efficiently find a good partition?**
  - **Problem:** Computing best conductance cut is NP-hard

Spectral Graph Partitioning

- A: adjacency matrix of undirected **G**
  - A_{ij} **=1** if (i, j) is an edge, else **0**
- x is a vector in $\Re^n$ with components $(x_1, \ldots, x_n)$
  - Think of it as a label/value of each node of 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**

What is the Meaning of Ax?

- j^{th} **coordinate of $A \cdot x$:**
  - Sum of the x-values of neighbors of j
  - Make this a new x-value at node j

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

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

- **Spectral Graph Theory:**
  - Analyze the "spectrum" of matrix representing $\boldsymbol{G}$
  - **Spectrum:** Eigenvectors $\boldsymbol{x}^{(i)}$ of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues $\boldsymbol{\lambda_i}$: $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$

$$\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$$

Note: We sort λ_i in ascending (not descending) order!

Example: d-Regular Graph

- Suppose all nodes in $\boldsymbol{G}$ have degree $\boldsymbol{d}$ ($\boldsymbol{G}$ is $\boldsymbol{d}$-regular) and $\boldsymbol{G}$ is connected
- **What are some eigenvalues/vectors of $\boldsymbol{G}$?**

  $\boldsymbol{A}\cdot \boldsymbol{x} = \boldsymbol{\lambda} \cdot \boldsymbol{x}$ What is λ? What x?

  - **Let's try: $\boldsymbol{x} = (\mathbf{1}, \mathbf{1}, \dots, \mathbf{1})$**
  - **Then: $\boldsymbol{A} \cdot \boldsymbol{x} = (\boldsymbol{d}, \boldsymbol{d}, \dots, \boldsymbol{d}) = \boldsymbol{\lambda} \cdot \boldsymbol{x}$. So: $\boldsymbol{\lambda} = \boldsymbol{d}$**
  - **We found an eigenpair of $\boldsymbol{G}$:**
    $\boldsymbol{x} = (\mathbf{1}, \mathbf{1}, \dots, \mathbf{1}), \boldsymbol{\lambda} = \boldsymbol{d}$
- $\boldsymbol{d}$ is the largest eigenvalue of $\boldsymbol{A}$ (see next slide)

Remember the meaning of $\boldsymbol{y} = \boldsymbol{A}\cdot \boldsymbol{x}$:

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

Note, this is just one eigenpair.
An *n* by *n* matrix can have up to *n* eigenpairs.

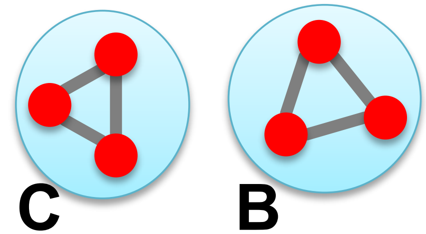
d is the Largest Eigenvalue of A

Details!

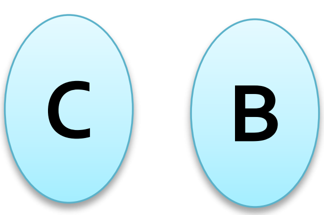
- G is d-regular connected, A is its adjacency matrix
- **Claim:**
  - (1) d has multiplicity of **1** (there is only **1** eigenvector associated with eigenvalue d)
  - (2) **d** is the largest eigenvalue of A
- **Proof:**
  - To obtain value eigval d we needed $x_i = x_j$ for every i, j
  - This means $x = c \cdot (1,1, \ldots, 1)$ for some const. c
  - **Define:** Set S = nodes i with maximum value of x_i
  - Then consider some vector y which is not a multiple of vector $(\mathbf{1}, \ldots, \mathbf{1})$. So not all nodes i (with labels y_i) are in S
  - Consider some node $j \in S$ and a neighbor $i \notin S$ then node j gets a value strictly less than d
  - So y is not eigenvector! And so d is the largest eigenvalue!

Example: Graph on 2 Components

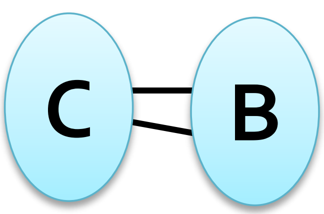
- **What if G is not connected?**
  - G has **2** components, each d-regular



- **What are some eigenvectors?**
  - $x =$ Put all 1s on C and 0s on B or vice versa
    - $x' = (\underbrace{1, \dots, 1}_{|C|}, \underbrace{0, \dots, 0}_{|B|})^T$ **then** $\mathbf{A} \cdot x' = (d, \dots, d, 0, \dots, 0)^T$
    - $x'' = (0, \dots, 0, 1, \dots, 1)^T$ **then** $A \cdot x'' = (0, \dots, 0, d, \dots, d)^T$
    - And so in both cases the corresponding $\lambda = d$
- **A bit of intuition:**



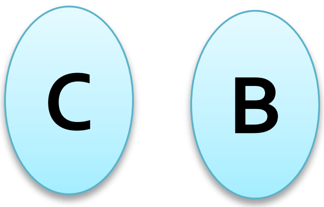
$\lambda_n = \lambda_{n-1}$



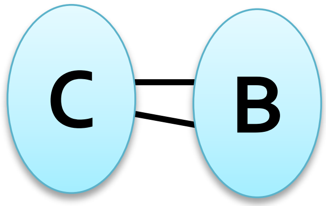
$\lambda_n - \lambda_{n-1} \approx 0$

2nd largest eigval. λ_{n-1} now has value very close to λ_n

More Intuition



$\lambda_n = \lambda_{n-1}$



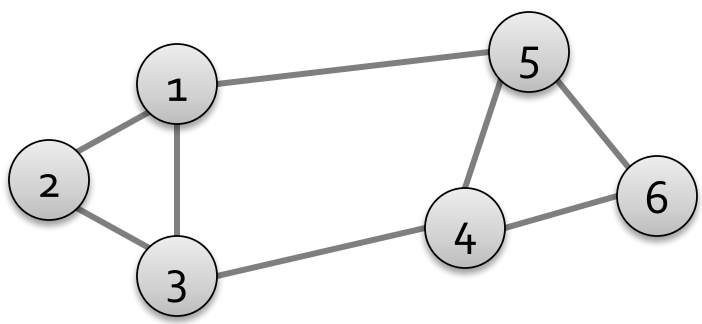
$\lambda_n - \lambda_{n-1} \approx 0$

2nd largest eigval. λ_{n-1} now has value very close to λ_n

- If the d-regular graph is connected (right example) then we already know that $\boldsymbol{x_n} = (\mathbf{1}, \dots \mathbf{1})$ is an eigenvector
- Eigenvectors are orthogonal so then the components of $\boldsymbol{x_{n-1}}$ must sum to **0**
  - Why? $\boldsymbol{x_n} \cdot \boldsymbol{x_{n-1}} = \mathbf{0}$ then $\sum_i \boldsymbol{x_n}[\boldsymbol{i}] \cdot \boldsymbol{x_{n-1}}[\boldsymbol{i}]$ = $\sum_i \boldsymbol{x_{n-1}}\ [\boldsymbol{i}]$ =0
  - $\boldsymbol{x_{n-1}}$ "splits" the nodes into two groups
    - $\boldsymbol{x_{n-1}}[\boldsymbol{i}] > \mathbf{0}$ vs. $\boldsymbol{x_{n-1}}[\boldsymbol{i}] < \mathbf{0}$
  - So we in principle could look at the eigenvector of the 2nd largest eigenvalue and declare nodes with positive label in **C** and negative label in **B.** (but there are still many details for us to figure out here)

Matrix Representations

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

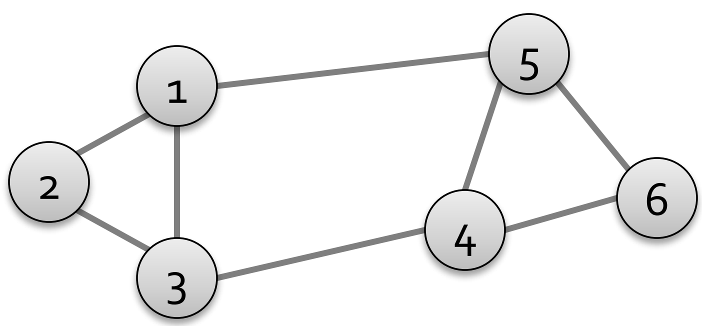


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

- **Important properties:**
  - **Symmetric matrix**
  - **Has n real eigenvalues**
  - **Eigenvectors are real-valued and orthogonal**

Matrix Representations

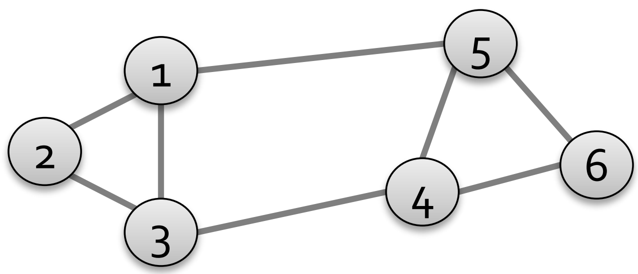
- **Degree matrix (D):**
  - $n \times n$ diagonal matrix
  - $D=[d_{ii}]$, d_{ii} = degree of node i



	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 Representations

- **Laplacian matrix (L):**
  - $n \times n$ symmetric matrix



	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

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

- **What is trivial eigenpair?**
  - $x = (1, \dots, 1)$ then $L \cdot x = 0$ and so $\lambda = \lambda_1 = 0$
- **Important properties of L:**
  - **Eigenvalues** are non-negative real numbers
  - **Eigenvectors** are real (and always orthogonal)

3 Facts About the Laplacian L

Details!

(a) All eigenvalues are ≥ 0

(b) $x^T L x = \sum_{ij} L_{ij} x_i x_j \geq 0$ for every x

(c) L can be written as $L = N^T \cdot N$

- That is, L is positive semi-definite

- **Proof: (the 3 facts are saying the same thing)**
  - **(c)⇒(b):** $x^T L x = x^T N^T N x = (Nx)^T (Nx) \geq 0$
    - As it is just the square of length of Nx
  - **(b)⇒(a):** Let $\boldsymbol{\lambda}$ be an eigenvalue of $\boldsymbol{L}$. Then by **(b)** $x^T L x \geq 0$ so $x^T L x = x^T \lambda x = \lambda x^T x \Rightarrow \boldsymbol{\lambda} \geq \mathbf{0}$
  - **(a)⇒(c):** is also easy! Do it yourself.

λ_2 as an Optimization Problem

- **Fact: For symmetric matrix M:**

$$\lambda_2 = \min_{x\,:\,x^T w_1 = 0} \frac{x^T M x}{x^T x}$$

(**w_1** is eigenvector corresponding to λ_1)

See next slide for the proof. Deriving this is a HW problem.

- **What is the meaning of min $x^T L x$ on G?**
  - $x^T L\, x = \sum_{i,j=1}^{n} L_{ij}\, x_i x_j = \sum_{i,j=1}^{n} \left(D_{ij} - A_{ij}\right) x_i x_j$
  - $= \sum_i D_{ii} x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j$
  - $= \sum_{(i,j)\in E} (\underbrace{x_i^2 + x_j^2} - 2x_i x_j) = \sum_{(i,j)\in E} \left(x_i - x_j\right)^2$

Node i has degree d_i. So, value x_i^2 needs to be summed up d_i times.
But each edge (i,j) has two endpoints so we need $x_i^2 + x_j^2$

Proof: $\lambda_2 = \min_{x\,:\,x^T w_1 = 0} \frac{x^T M x}{x^T x}$

Details!

- Write x in basis of eigenvecs $w_1, w_2, \dots, w_n$ of $\boldsymbol{M}$ and λ_i are corresponding eigenvalues. So, $x = \sum_i^n \alpha_i w_i$
- Then we get: $Mx = \sum_i \alpha_i \underbrace{M w_i}_{\lambda_i w_i} = \sum_i \alpha_i \lambda_i w_i$
- So, what is $\boldsymbol{x^T M x}$?
  - $x^T M x = \underbrace{\left(\sum_i \alpha_i w_i\right)^T}_{x^T} \underbrace{\left(\sum_i \alpha_i \lambda_i w_i\right)}_{Mx} \quad = \sum_{ij} \alpha_i \lambda_j \alpha_j$

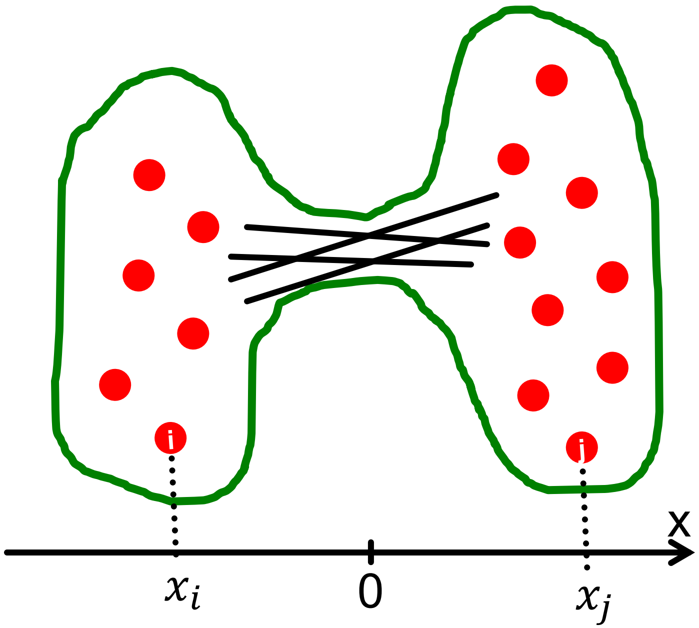
    $= \sum_i \alpha_i^2 \lambda_i \underbrace{w_i^T w_i}_{=\,0 \text{ if } i \neq j,\ 1 \text{ otherwise}} = \boldsymbol{\sum_i \lambda_i \alpha_i^2}$
  - Want minimize this over all unit vectors $\boldsymbol{w}$:
    $\boldsymbol{w}$ = min over choices of $(\alpha_1, \dots \alpha_n)$ so that:
    - $\mathrm{x}^T w_1 = 0$, rewrite it as $\left(\sum_i \alpha_i w_i\right) \cdot w_1 = 0$ and remember that $w_i^T w_j = 0$ (because w are eigenvectors). Then $\alpha_1 = 0$
    - $\sum \alpha_i^2 = 1$ (unit length)
  - So, to minimize this, set $\boldsymbol{\alpha_2 = 1}$ and the rest to 0 $\boldsymbol{\sum_i \lambda_i \alpha_i^2 = \lambda_2}$

Finding x that Solves $\lambda_2 = \min_{x:\, x^T w_1 = 0} \frac{x^T M x}{x^T x}$

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

- **Remember:**

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



Balance to minimize

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)

Find Optimal Cut [Fiedler'73]

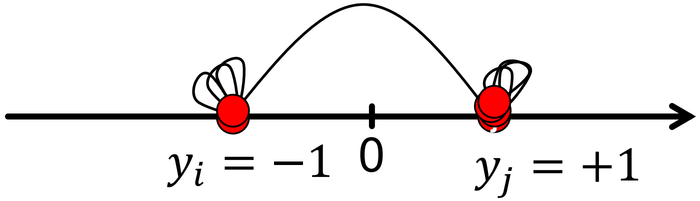
- **Back to finding the optimal cut**
- **Express partition (A,B) as a vector**

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

- Enforce that **|A| = |B| → $\Sigma_i y_i = 0$**
  - Equivalent to being orthogonal to the trivial eigenvector $(\mathbf{1}, \ldots, \mathbf{1})$
- We can minimize the cut of the partition by finding a vector y that **minimizes**:

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

Can't solve exactly. Let's relax y and allow it to take any real value.

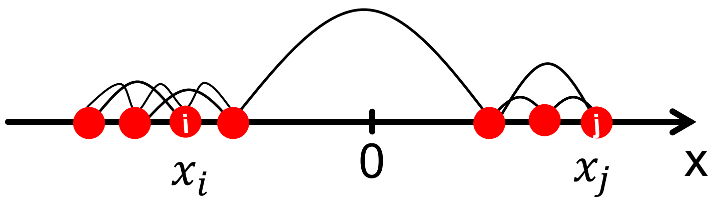


Rayleigh Theorem

$$\lambda_2 = \min_{x\,:\,x^T w_1 = 0} \frac{x^T M x}{x^T x}$$

Slide 18

$$\min_{\substack{y \in \mathbb{R}^n \,:\, \sum_i y_i = 0 \\ \sum_i y_i^2 = 1}} f(y) = \sum_{(i,j)\in E} (y_i - y_j)^2 = y^T L y$$



- $\boldsymbol{\lambda_2 = \min_y f(y)}$: The minimum value of $\boldsymbol{f(y)}$ is given by the 2nd smallest eigenvalue $\boldsymbol{\lambda_2}$ of the Laplacian matrix $\boldsymbol{L}$
- $\mathbf{x} = \arg\min_y f(y)$: The optimal solution for $\boldsymbol{y}$ is given by the eigenvector $\boldsymbol{x}$ corresponding to $\boldsymbol{\lambda_2}$, referred to as the **Fiedler vector**
- Can use sign of x_i to determine cluster assignment of node i

Approx. Guarantee of Spectral

Details!

- Suppose there is a partition of **G** into **A** and **B** where $|A| \leq |B|$, s.t. "conductance" of the cut (A,B) is $\boldsymbol{\beta} = \frac{(\#\, edges\ from\ A\ to\ B)}{|A|}$ then $\boldsymbol{\lambda_2} \leq 2\beta$

  Note: |A|<|B|

  - This is the approximation guarantee of the spectral clustering: Spectral finds a cut that has at most **twice the conductance** as the optimal one of conductance $\boldsymbol{\beta}$.
- **Proof:**
  - Let: $\boldsymbol{a} = |\boldsymbol{A}|, \boldsymbol{b} = |\boldsymbol{B}|$ and $\boldsymbol{e} = \#$ edges from $\boldsymbol{A}$ to $\boldsymbol{B}$
  - Enough to choose some $\boldsymbol{x_i}$ based on $\boldsymbol{A}$ and $\boldsymbol{B}$ such that:

$$\underbrace{\lambda_2 \leq \frac{\sum (x_i - x_j)^2}{\sum_i x_i^2}}_{\lambda_2 \text{ is only smaller}} \leq 2\beta \text{ (while also } \sum_i x_i = 0)$$

Approx. Guarantee of Spectral

Details!

- **Proof (continued):**
  - **1) Let's set:** $x_i = \begin{cases} -\frac{1}{a} & \text{if } i \in A \\ +\frac{1}{b} & \text{if } i \in B \end{cases}$

    Note: |A|<|B|

    - Let's quickly verify that $\sum_i x_i = 0$: $a\left(-\frac{1}{a}\right) + b\left(\frac{1}{b}\right) = \mathbf{0}$
  - **2) Then:** $\frac{\sum (x_i - x_j)^2}{\sum_i x_i^2} = \frac{\sum_{i \in A, j \in B}\left(\frac{1}{b}+\frac{1}{a}\right)^2}{a\left(-\frac{1}{a}\right)^2 + b\left(\frac{1}{b}\right)^2} = \frac{e \cdot \left(\frac{1}{a}+\frac{1}{b}\right)^2}{\frac{1}{a}+\frac{1}{b}} =$

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

e … number of edges between A and B

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

Approx. Guarantee of Spectral

Details!

- **Putting it all together: The Cheeger inequality**

$$\frac{\beta^2}{2k_{max}} \leq \lambda_2 \leq 2\beta$$

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

So far...

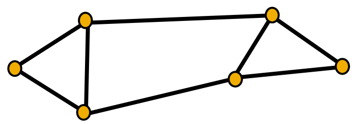
- **How to define a "good" partition of a graph?**
  - Minimize a given graph cut criterion
- **How to efficiently identify such a partition?**
  - Approximate using information provided by the eigenvalues and eigenvectors of a graph
- **Spectral Clustering**

Spectral Clustering Algorithm

- **Three basic stages:**
  - **1) Pre-processing**
    - Construct a matrix representation of the graph
  - **2) Decomposition**
    - Compute eigenvalues and eigenvectors of the matrix
    - Map each point to a lower-dimensional representation based on one or more eigenvectors
  - **3) Grouping**
    - Assign points to two or more clusters, based on the new representation

Spectral Partitioning Algorithm

- **1) 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

- **2) Decomposition:**
  - Find eigenvalues λ and eigenvectors x of the matrix L
  - Map vertices to corresponding components of x_2

λ=

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

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

- 3) **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

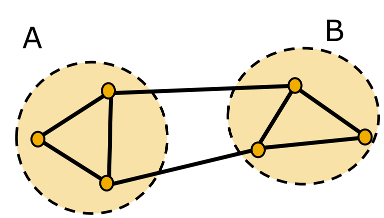
Split at 0:

Cluster A: Positive points

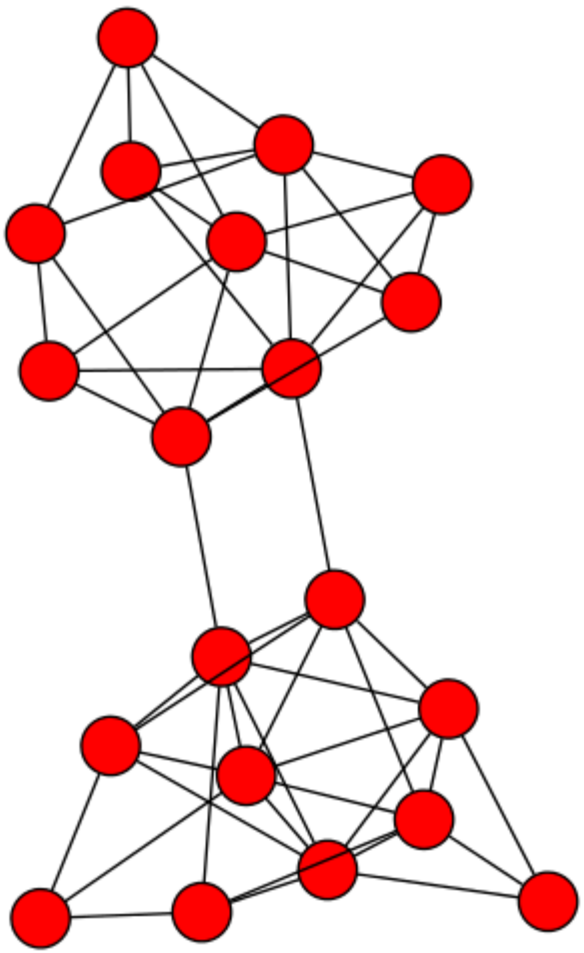
Cluster B: Negative points

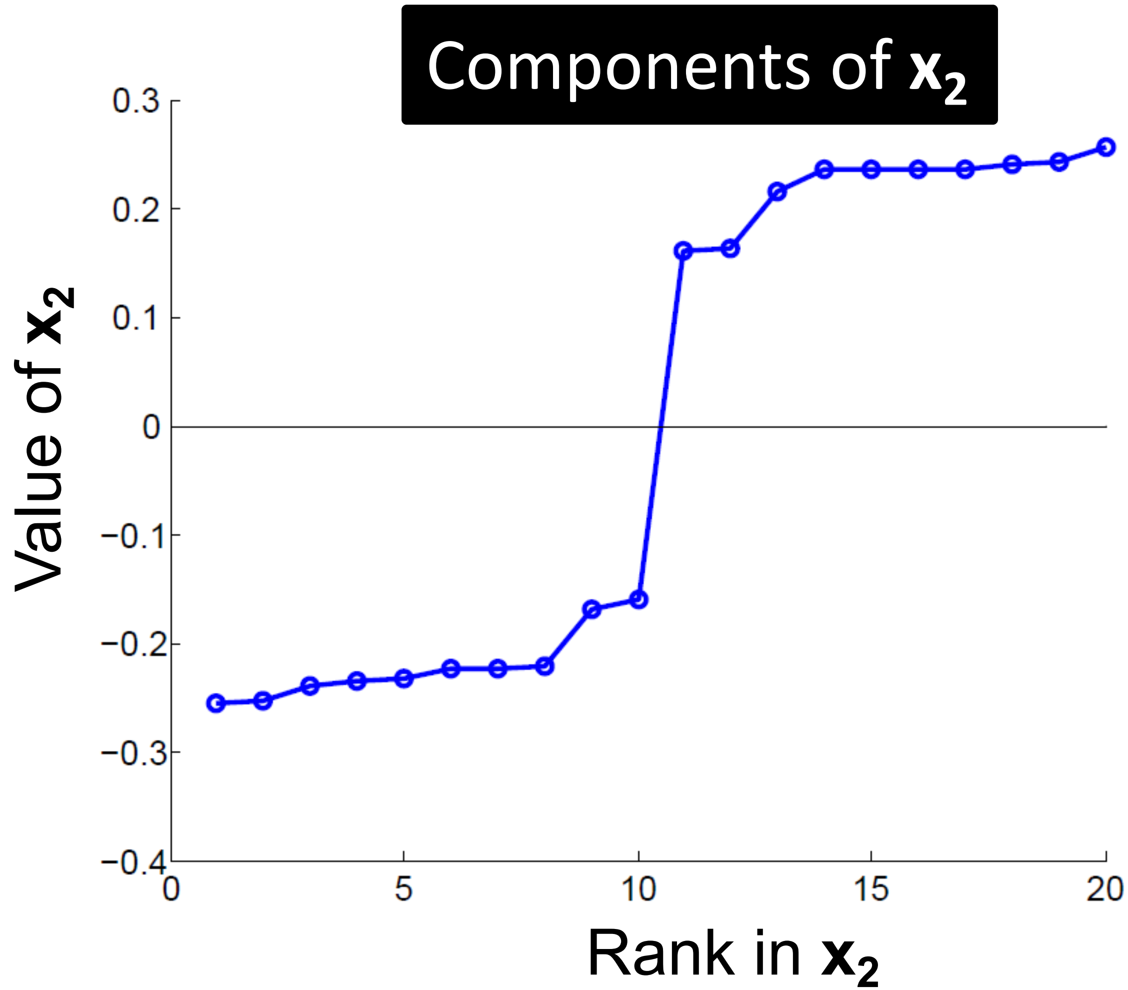
1	0.3
2	0.6
3	0.3

4	-0.3
5	-0.3
6	-0.6

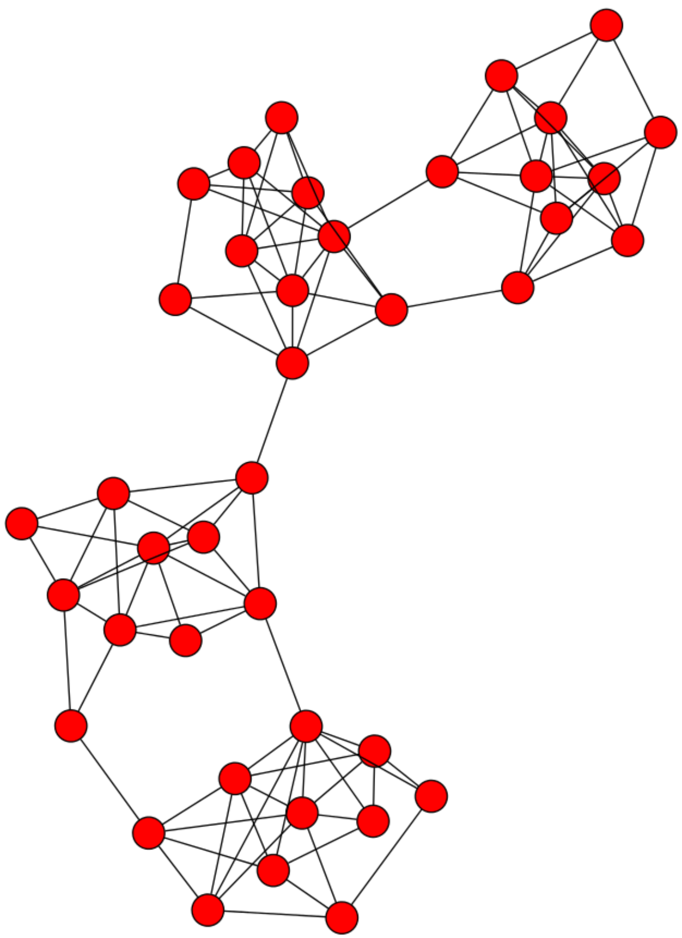


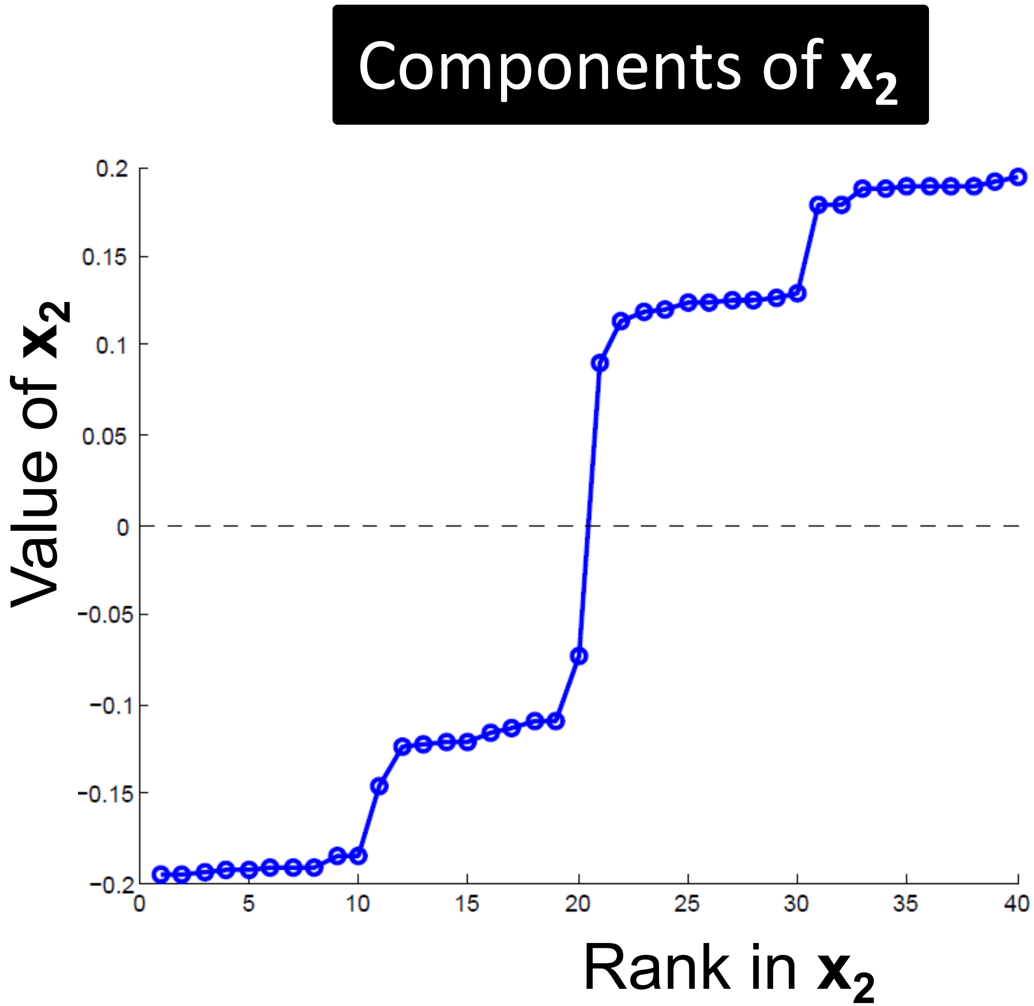
Example: Spectral Partitioning



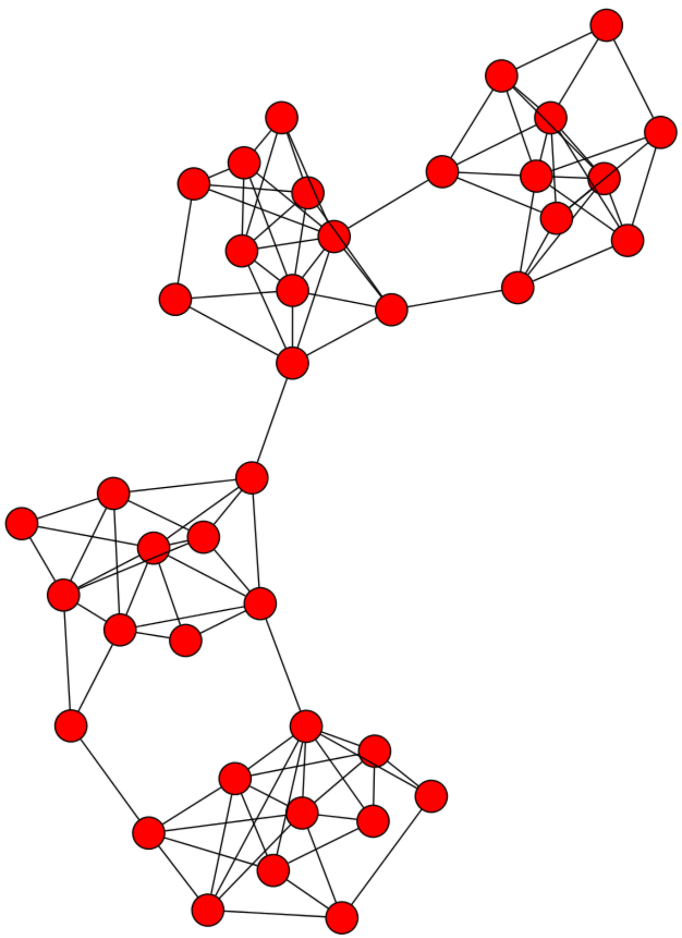


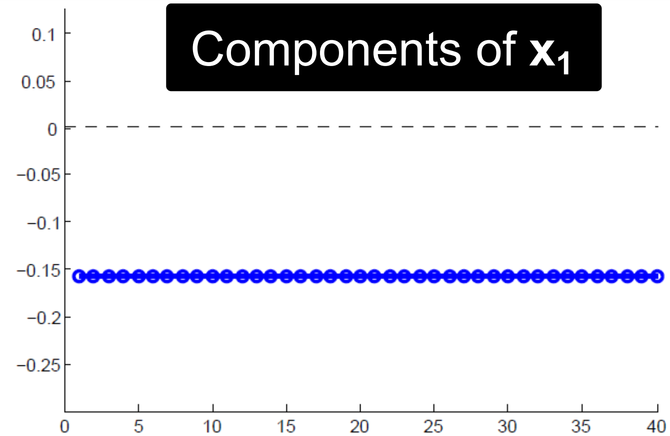
Example: Spectral Partitioning

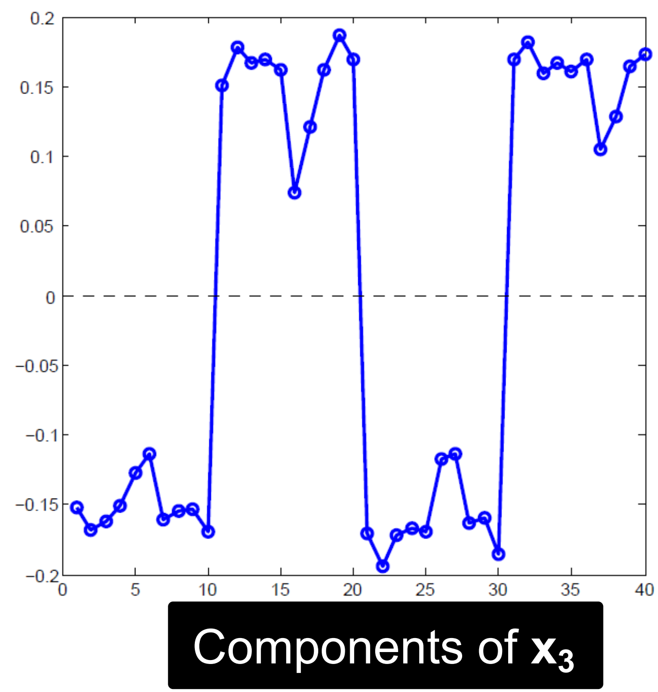




Example: Spectral Partitioning







k-Way Spectral Clustering

- **How do we partition a graph into k clusters?**
- **Two basic approaches:**
  - **Recursive bi-partitioning** [Hagen et al., ’92]
    - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
    - Disadvantages: Inefficient, unstable
  - **Cluster multiple eigenvectors** [Shi-Malik, ’00]
    - Build a reduced space from multiple eigenvectors
      - Each node is now represented by $\boldsymbol{k}$ numbers
      - We then cluster (apply k-means) the nodes based on their $\boldsymbol{k}$-dim representation
    - Commonly used in recent papers
    - A preferable approach…

Why Use Multiple Eigenvectors?

- **Approximates the optimal cut** [Shi-Malik, ’00]
  - Can be used to approximate optimal k-way normalized cut
- **Emphasizes cohesive clusters**
  - Increases the unevenness in the distribution of the data
  - Associations between similar points are amplified, associations between dissimilar points are attenuated
  - The data begins to “approximate a clustering”
- **Well-separated space**
  - Transforms data to a new “embedded space”, consisting of k orthogonal basis vectors
- Multiple eigenvectors prevent instability due to information loss

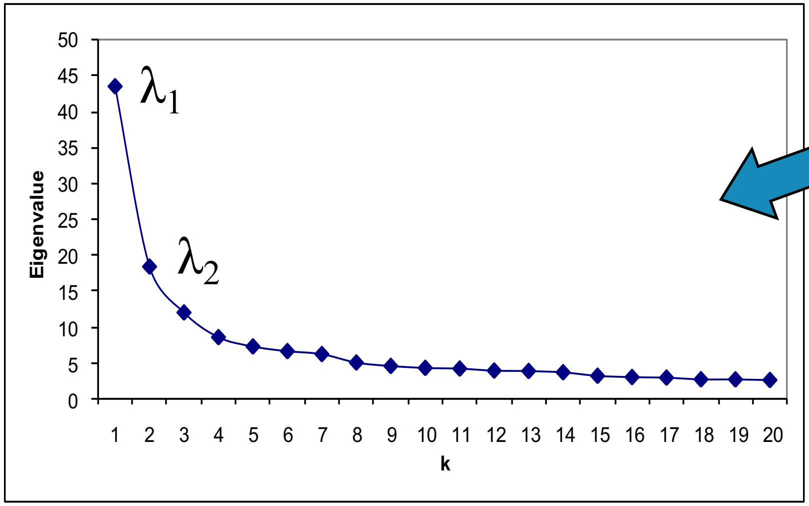
How to Select k?

- **Eigengap:**
  - The difference between two consecutive eigenvalues
- **Most stable clustering is generally given by the value k that maximizes eigengap Δ_k:**

$$\Delta_k = |\lambda_k - \lambda_{k-1}|$$

Note eigenvalues are sorted in descending order

- **Example:**



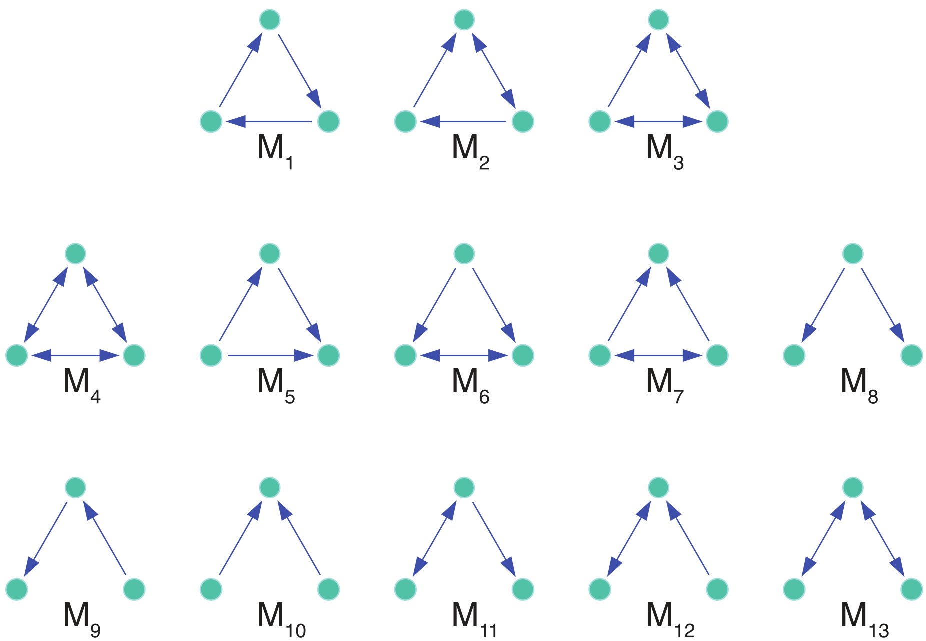
$$\max \Delta_k = |\lambda_2 - \lambda_1|$$

$\Rightarrow$ **Choose $k = 2$**

Motif-Based Spectral Clustering

Motif-based spectral clustering

- **What if we want our clustering based on other patterns (not edges)?**

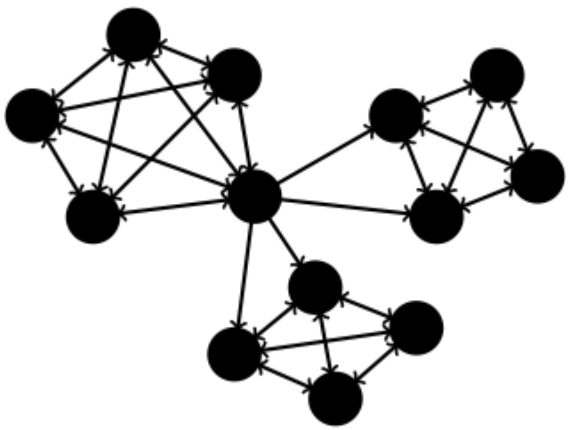


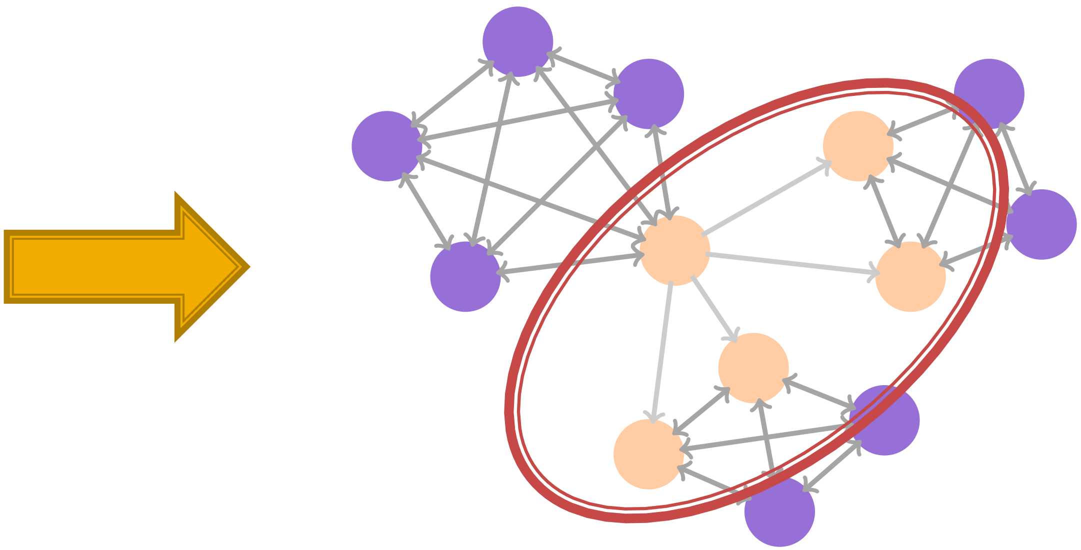
Small subgraphs (motifs, graphlets) are building blocks of networks [Milo et al., ’02]

Topic 1: Modules of Motifs

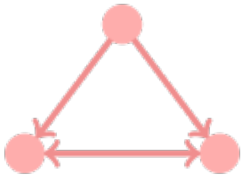
Find modules based on motifs!

Network:

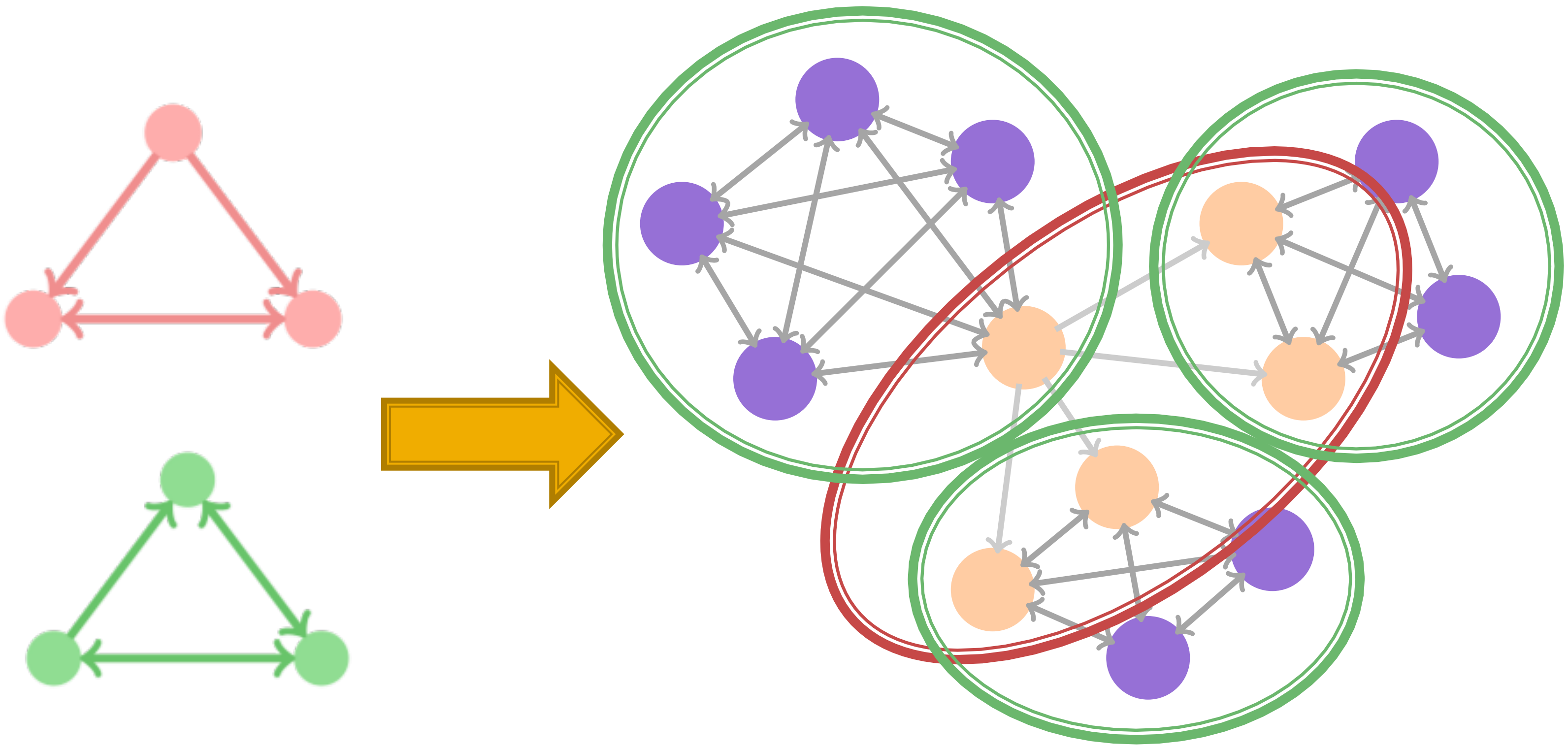




Motif:



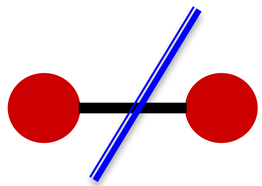
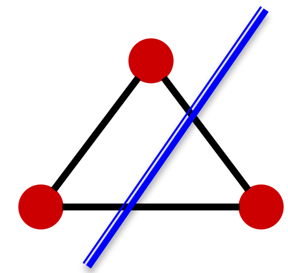
Modules of Motifs



Different motifs reveal different modular structures!

Defining Motif Conductance

Generalize Cut and Volume to motifs:

$edges\ cut$  → $motifs\ cut$ 

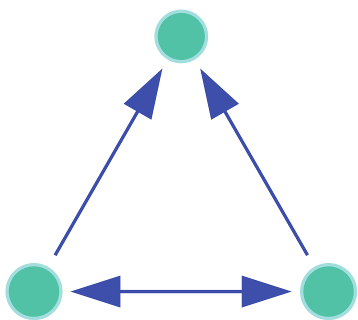
$vol(S) = \#(\text{edge end-points in S})$ → $vol_M(S) = \#(\text{motif end-points in S})$

$$\phi(S) = \frac{\#(edges\ cut)}{vol(S)}$$ → $$\boxed{\phi(S) = \frac{\#(motifs\ cut)}{vol_M(S)}}$$

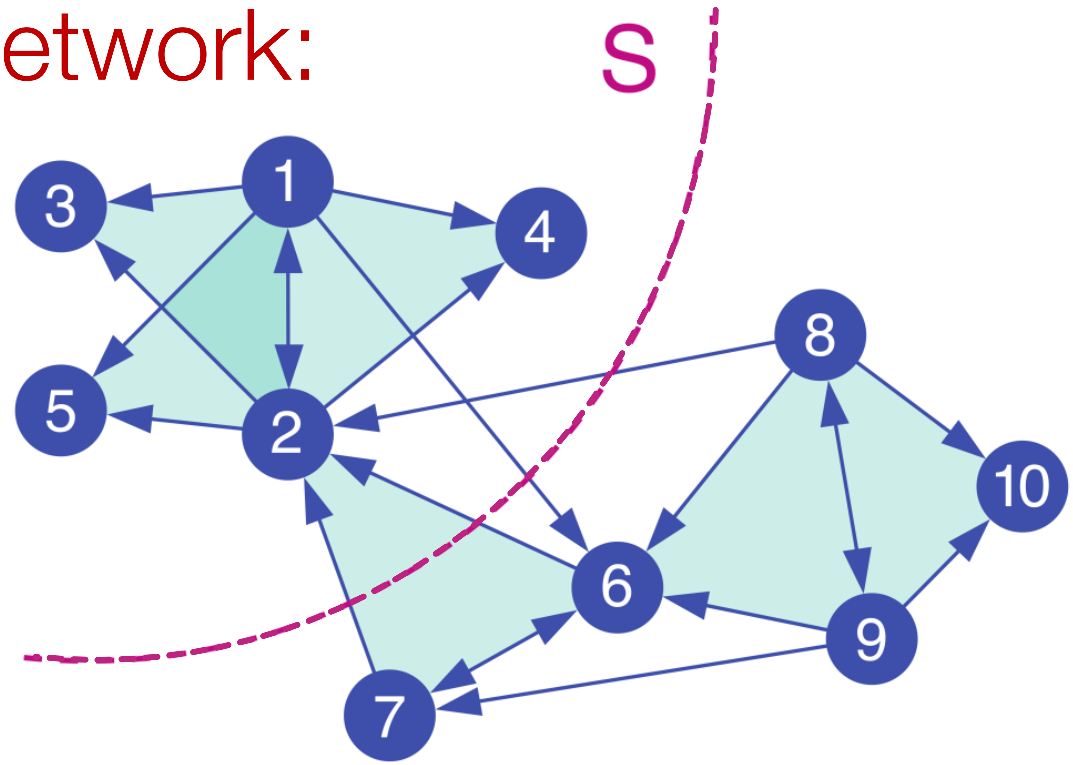
[Benson, Gleich, Leskovec, Science, 2016]

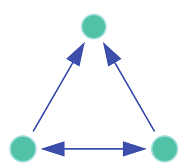
Motif Conductance: Example

Motif:



Network:



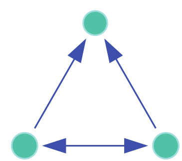


$$\phi_M(S) = \frac{\text{motifs cut}}{\text{motif volume}}$$

Higher-order Partitioning

How do we find clusters of motifs?

- Given a motif M and a graph G
- Find a set of nodes S that minimizes motif conductance



$$\phi_M(S) = \frac{\text{motifs cut}}{\text{motif volume}}$$

Bad news: Finding set S with the minimal motif conductance is NP-hard!

Motif Spectral Clustering

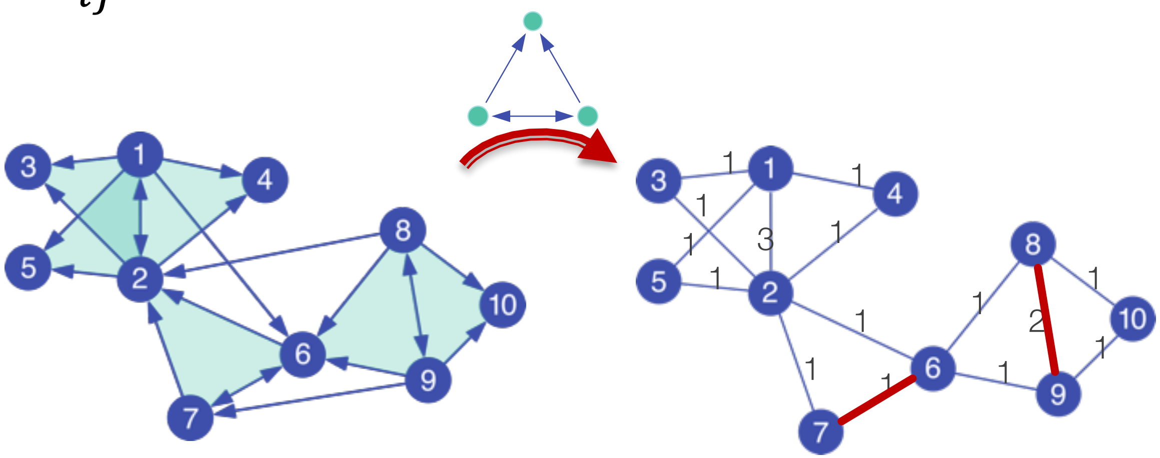
Solution: Motif Spectral Clustering

- Input: Graph $\boldsymbol{G}$ and motif $\boldsymbol{M}$
- Using $\boldsymbol{G}$ form a new weighted graph $\boldsymbol{W^{(M)}}$
- Apply spectral clustering on $\boldsymbol{W^{(M)}}$
- Output the clusters

Theorem: Resulting clusters will obtain near optimal motif conductance

Optimizing Motif Conductance

- **Three steps:**
  - 1) **Pre-processing**
    - $W_{ij}^{(M)}$ = # times edge (i, j) participates in the motif M



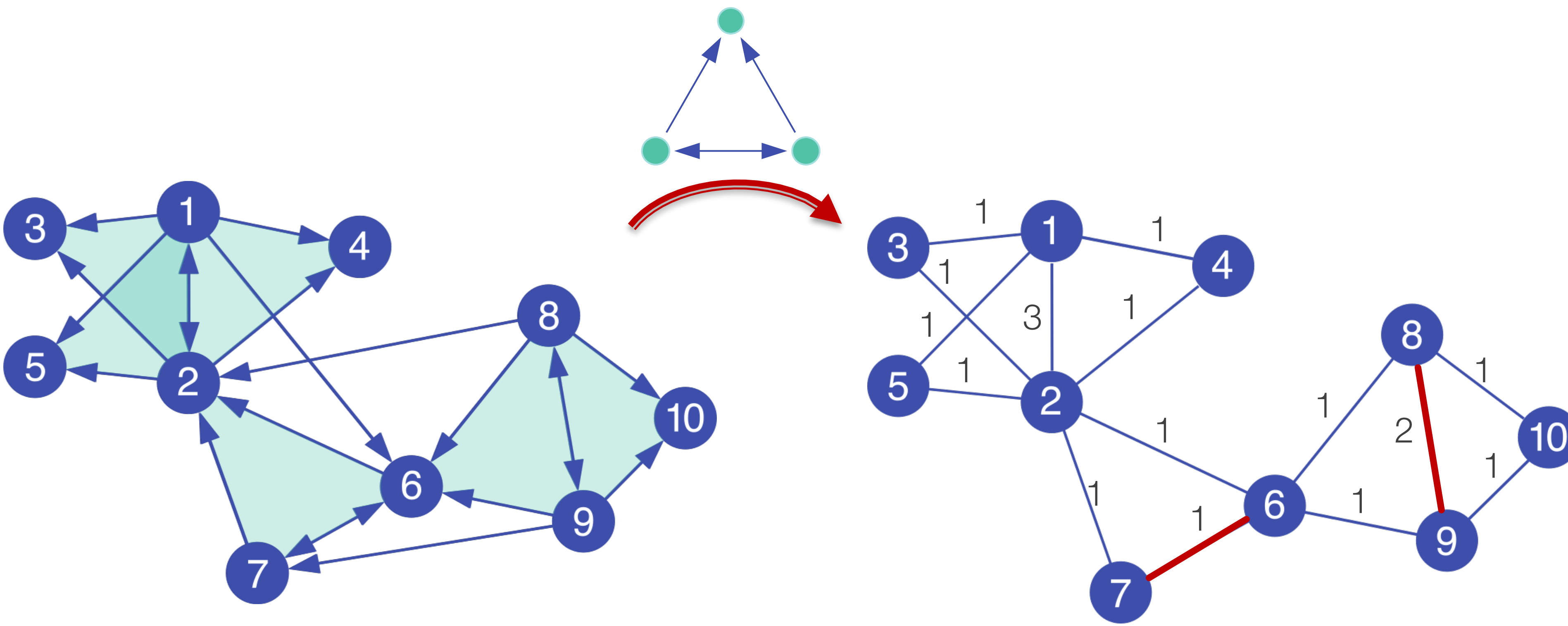
Graph G

Weighted graph $W^{(M)}$

See lecture 5 on motifs and the ESU algorithm for enumerating them

  - 2) **Decomposition**
    - Use standard spectral clustering (but on $W^{(M)}$)
  - 3) **Grouping**
    - Same as standard spectral clustering

Motif Spectral Clustering



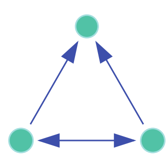
Graph G Weighted graph $W^{(M)}$

$W_{ij}^{(M)}$ = # of times edge (i,j) participates in motif M

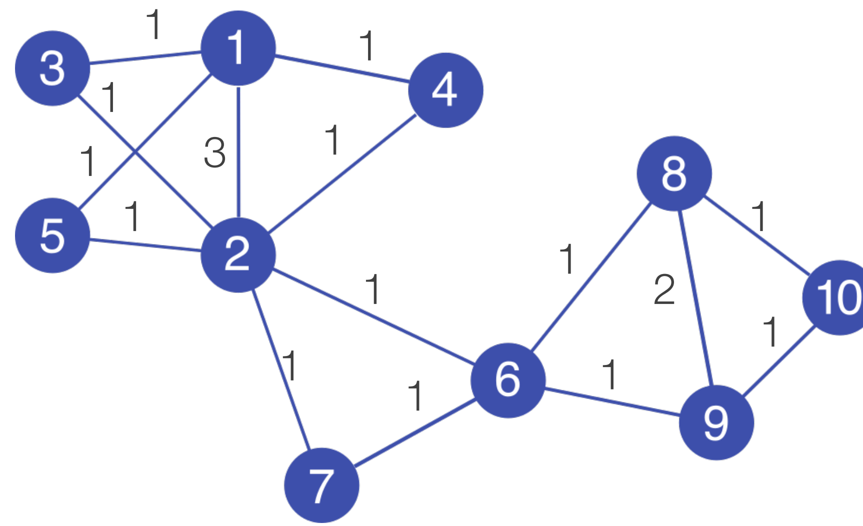
Motif Spectral Clustering

Insight:

Spectral clustering on weighted graph $W^{(M)}$ finds clusters of low motif conductance:



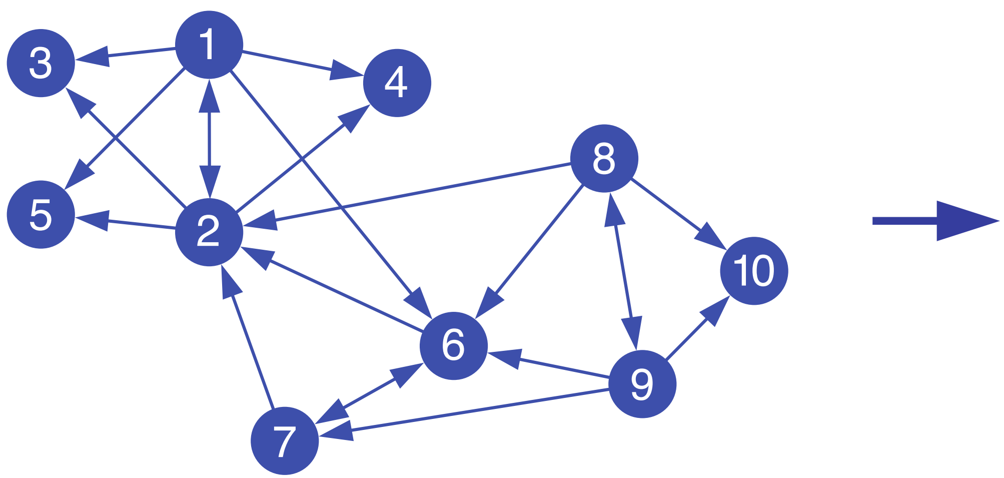
$$\phi_M(S) = \frac{\text{motifs cut}}{\text{motif volume}}$$



Weighted graph $W^{(M)}$

$W_{ij}^{(M)}$ = # of times edge (i,j) participates in motif M

Step 1: Create W



Nodes

$$
\text{Nodes}\quad
\begin{array}{c}
\\ 1\\ 2\\ 3\\ 4\\ 5\\ 6\\ 7\\ 8\\ 9\\ 10
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array}\\
\begin{bmatrix}
0 & \mathbf{3} & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0\\
\mathbf{3} & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0\\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0\\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{2} & 1\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & \mathbf{2} & 0 & 1\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{bmatrix}
\end{array}
$$

Step 1: Given motif M. Form weighted graph $W^{(M)}$

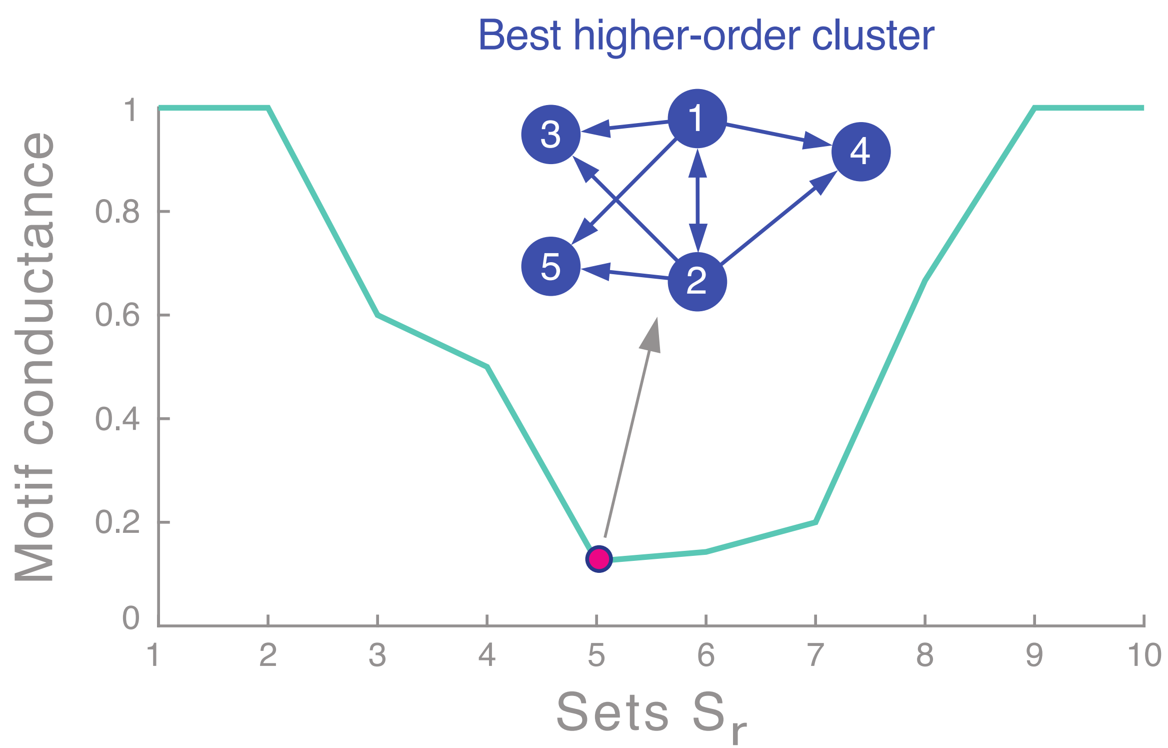
Step 2: Apply Spectral Clust to $W^{(M)}$

Step 2: Apply spectral clustering:
Compute Fiedler vector x
associated with λ_2 of the Laplacian of $L^{(M)}$

Set: $L^{(M)} = D^{(M)} - W^{(M)}$
Compute: $L^{(M)}x = \lambda_2 x$
Use x to identify communities

Degree matrix
$$D_{ii}^{(M)} = \sum_j W_{ij}^{(M)}$$

Step 3: Grouping (Sweep Procedure)



Step 3: Sort nodes by their values in x: $x_1, x_2, \ldots x_n$

Let $S_r = \{x_1, \ldots, x_r\}$ and compute the motif conductance of each S_r

Motif Cheeger Inequality

Theorem: The algorithm finds a set of nodes S for which

$$\phi_M(S) \leq 4\sqrt{\phi_M^*}$$

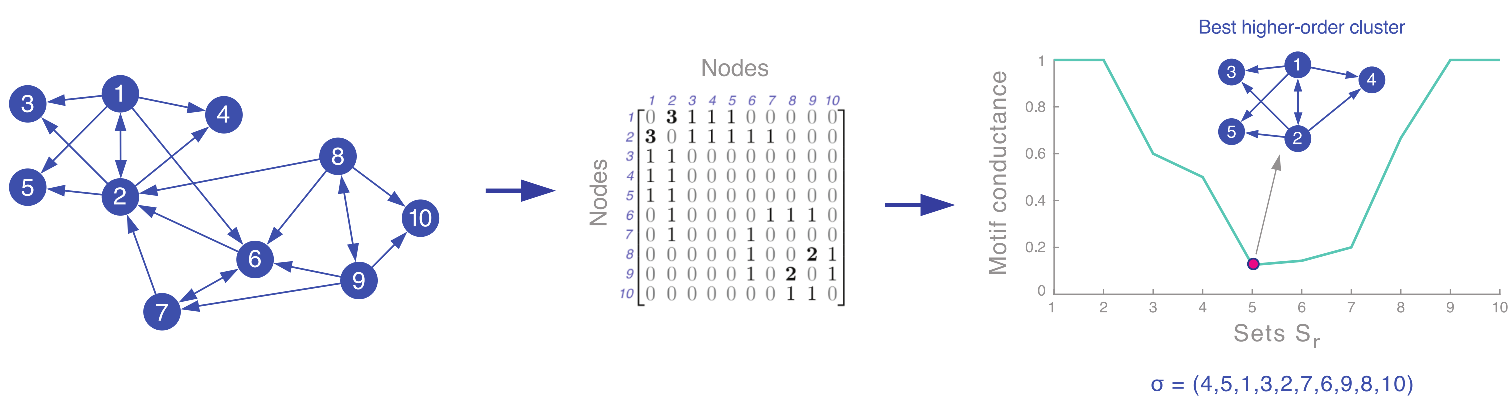
$\phi_M(S)$… motif conductance of S found by our algorithm

ϕ_M^* … motif conductance of optimal set S*

In other words: Clusters S found by the method are provably near optimal

Summary

- Generalization of community detection to higher-order structures
- Motif-conductance objective admits a motif Cheeger inequality
- Simple, fast, and scalable:



Two Examples

1) We don't know a motif of interest

- Food webs and new applications

2) We know the motif of interest

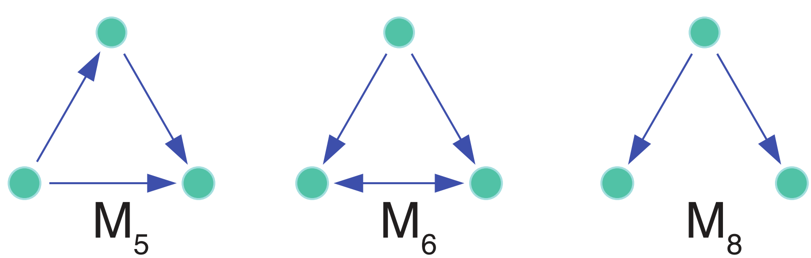
- Regulatory transcription networks, connectome, social networks

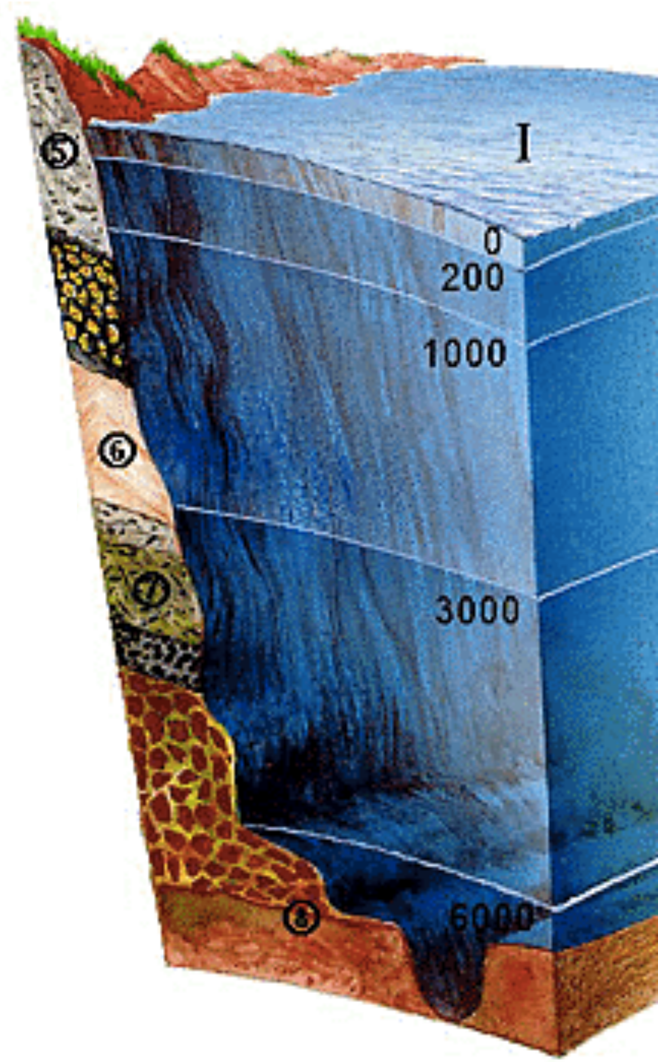
Application 1: Food webs

Florida Bay food web:

- **Nodes:** species in the ecosystem
- **Edges:** carbon exchange (who eats whom)

Different motifs capture different energy flow patterns:





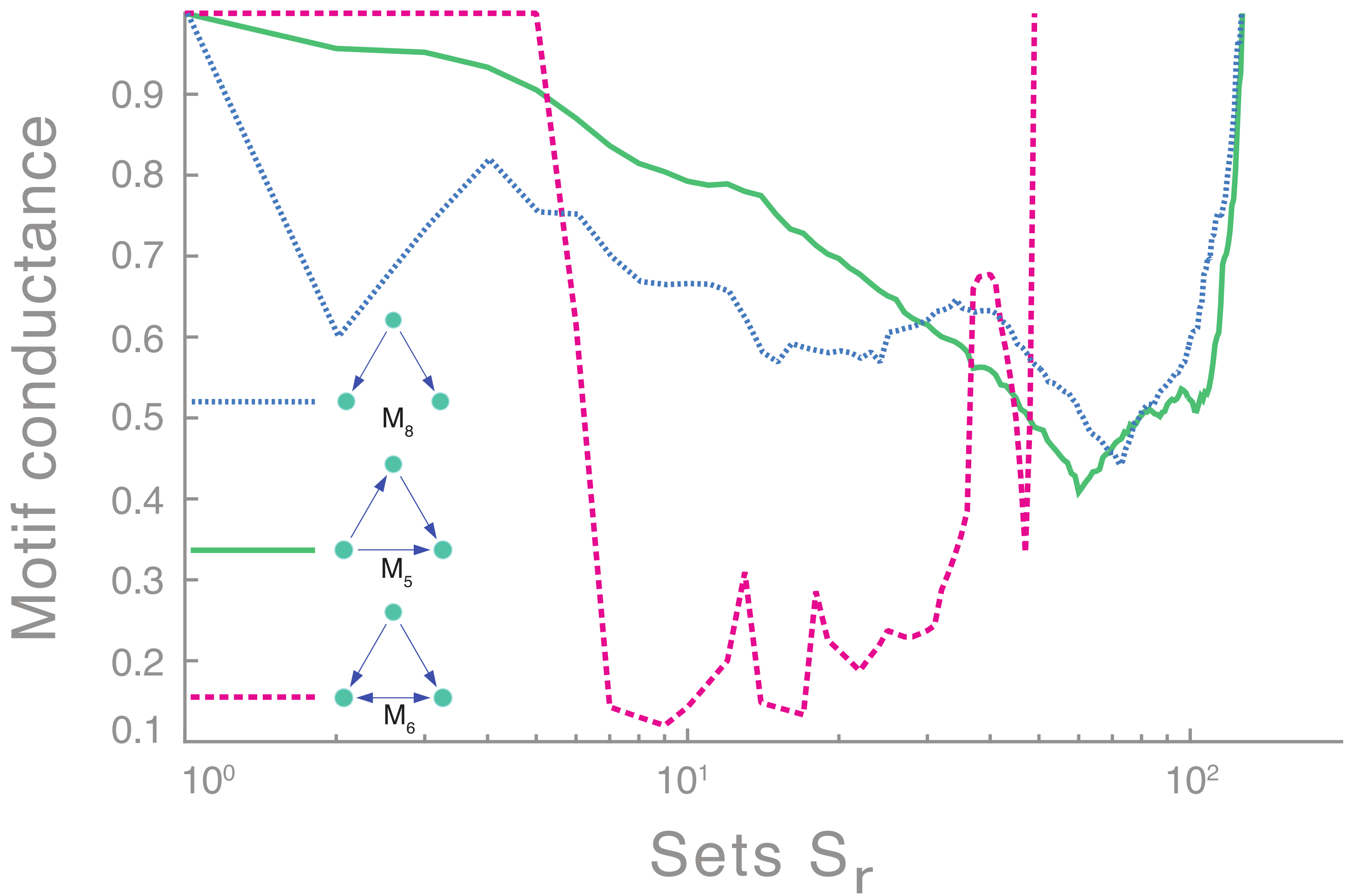
Florida Bay Food Web

Which motif organizes the food web?

Approach:

- Run motif spectral clustering separately for each of the 13 motifs
- Examine the **Sweep profile (next slide)** to see which motif gives best clusters

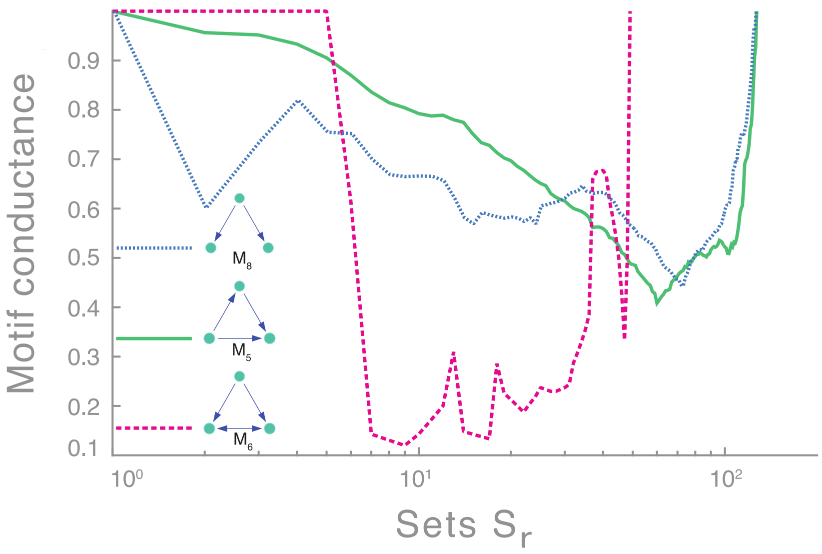
Florida Bay Food Web

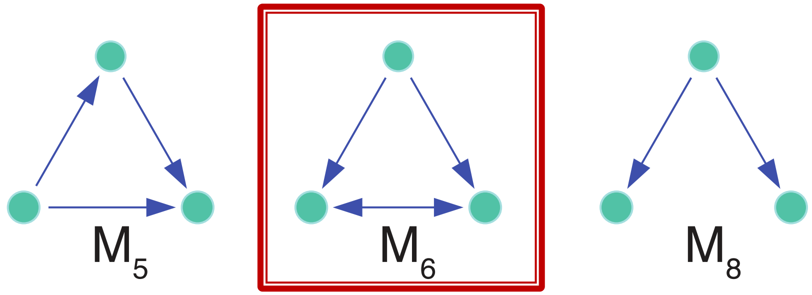


Food Web: Observations

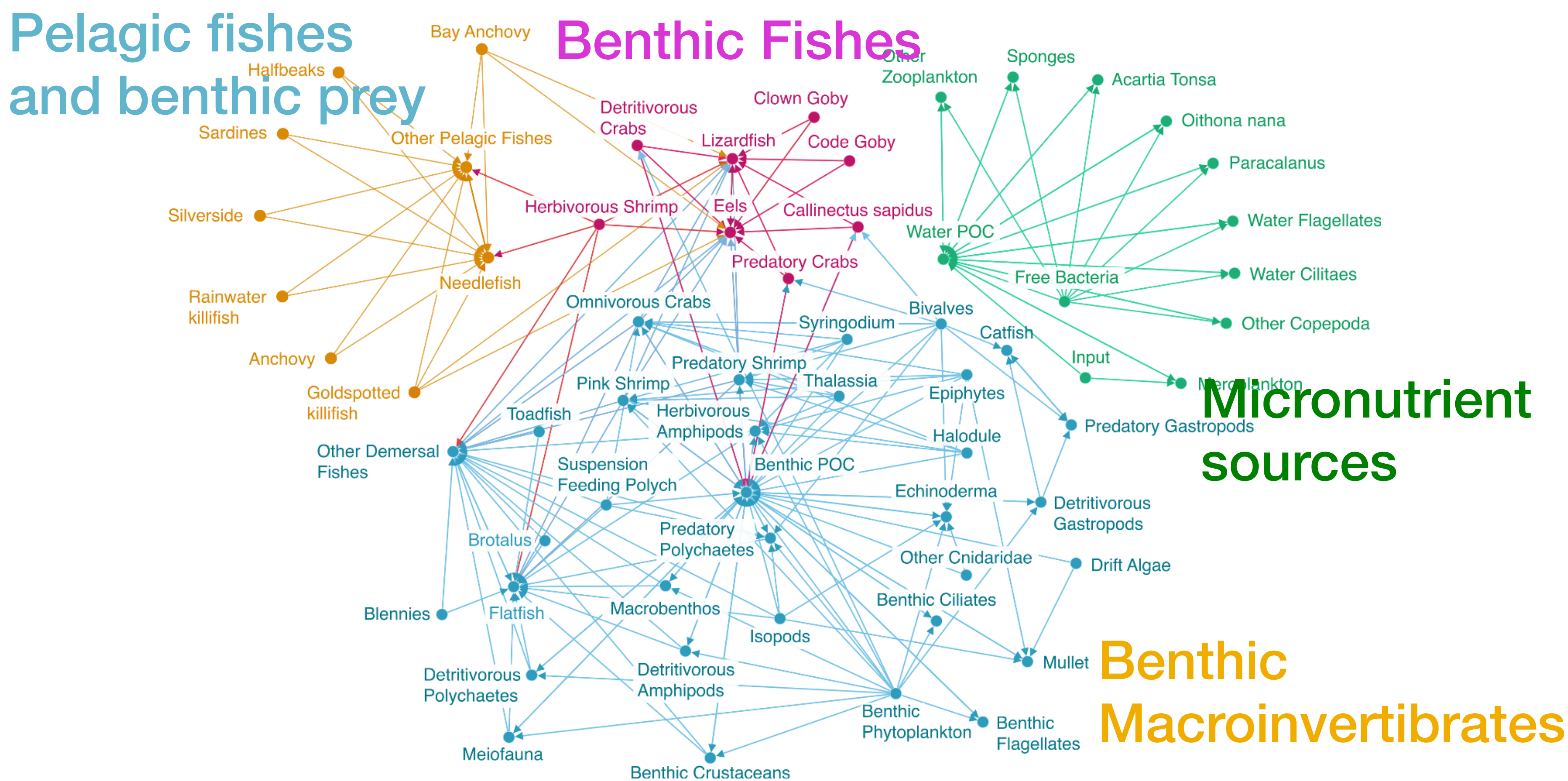
Observation:
Network organizes based on motif M_6 (but not M_5 or M_8)

- There exist good cuts for M_6 but not for M_5 or M_8



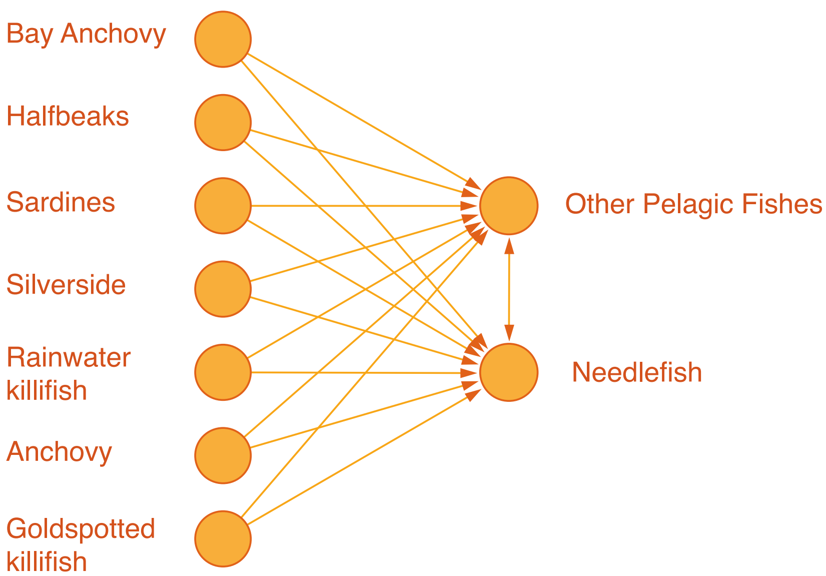


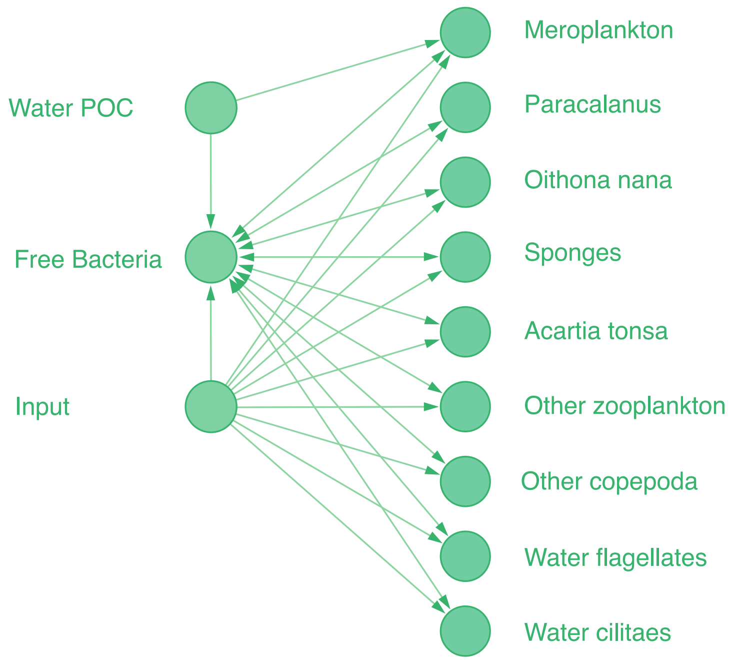
Food Web: Clusters



M_6 reveals known aquatic layers with higher accuracy (84% vs. 65%)

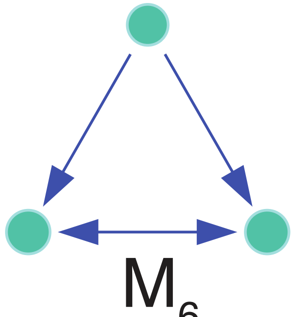
Structure of Aquatic Layers





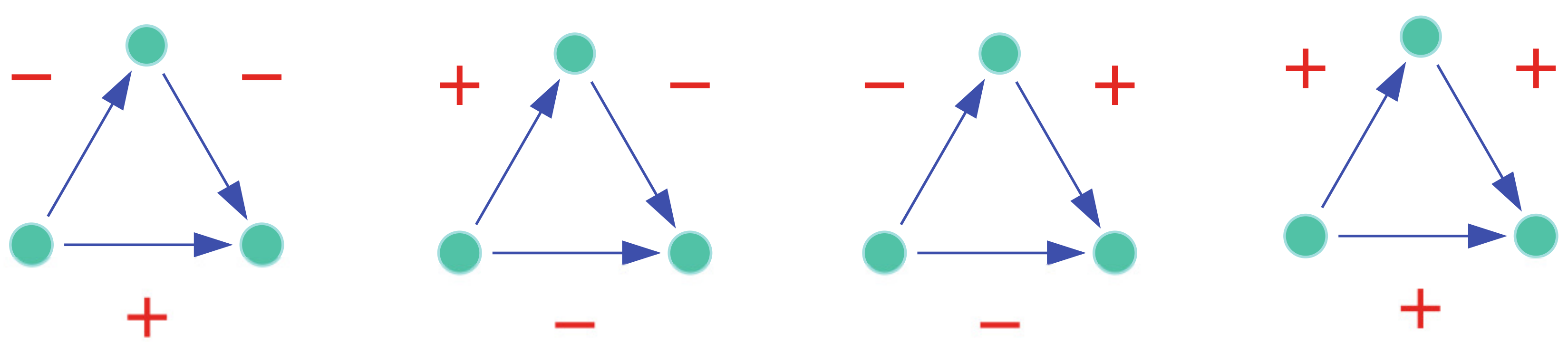
Aquatic layers organize based on M_6

- Many instances of M_6 inside
- Few instances of M_6 cross



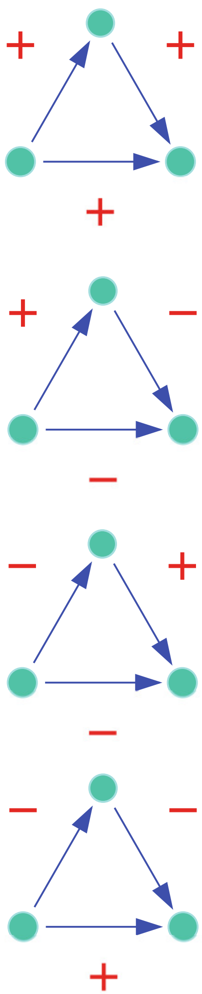
(2) Gene Regulatory Networks

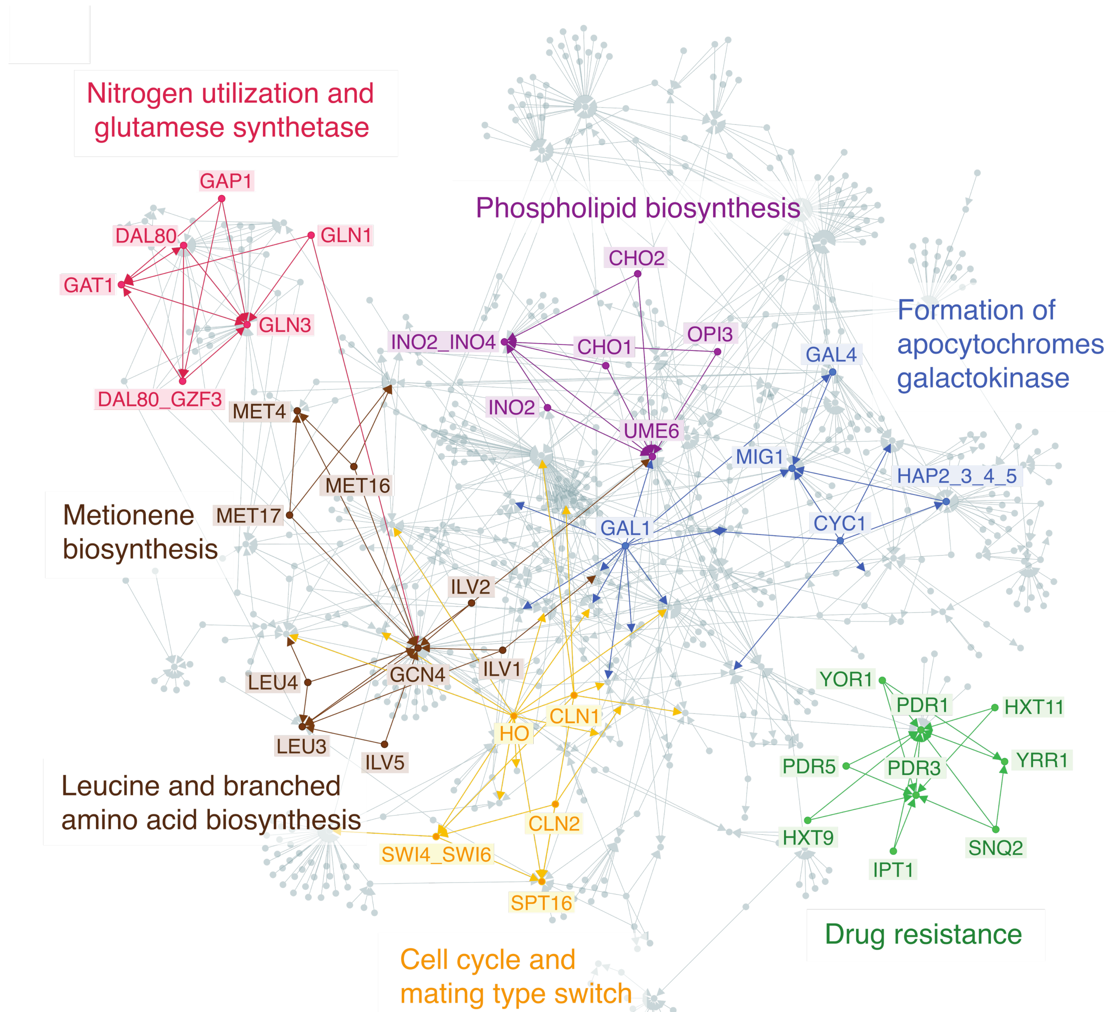
- Nodes are groups of genes in mRNA
- Edges are directed transcriptional regulation relationships



- The "feedforward loop" motif represents biological function [Alon '07]

Yeast Regulatory Network

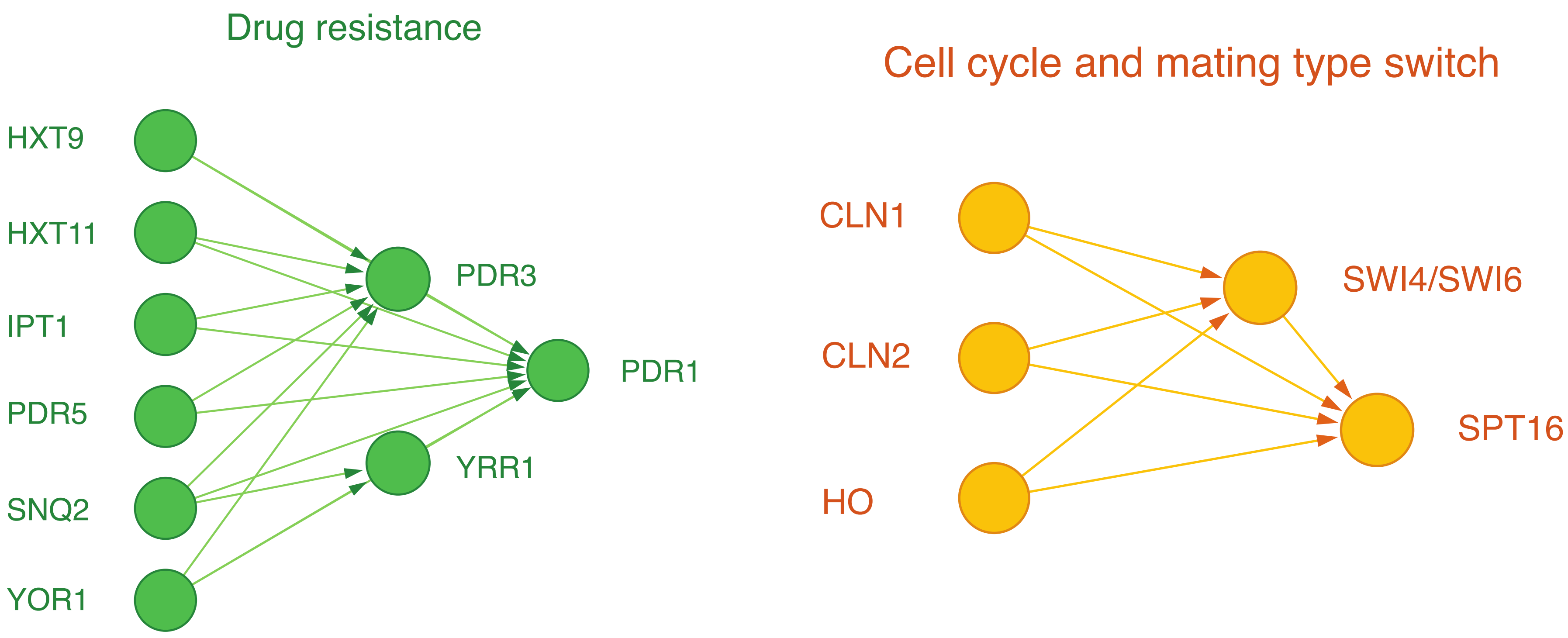




97% detection accuracy (vs. 68-82%)

Structure of Modules

- Feed forward loops:



Many other partitioning methods

- **METIS:**
  - Heuristic but works really well in practice
  - http://glaros.dtc.umn.edu/gkhome/views/metis
- **Graclus:**
  - Based on kernel k-means
  - http://www.cs.utexas.edu/users/dml/Software/graclus.html
- **Louvain:**
  - Based on Modularity optimization
  - http://perso.uclouvain.be/vincent.blondel/research/louvain.html
- **Clique percorlation method:**
  - For finding overlapping clusters
  - http://angel.elte.hu/cfinder/