Modelling and Analysis of Complex Networks

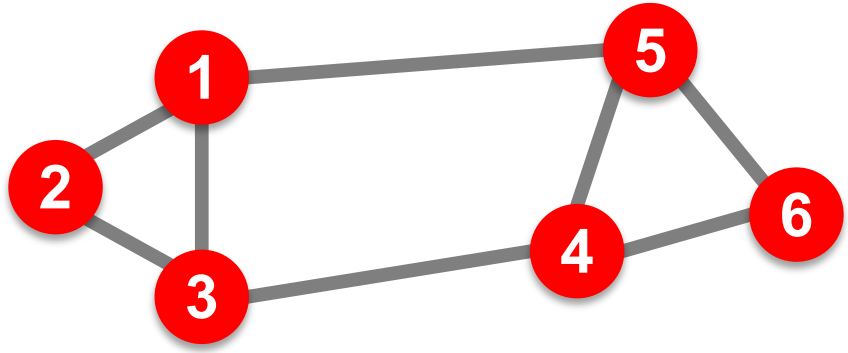
— Semester 3, Master of Data Science —

Jun Pang
University of Luxembourg

Graph Partitioning

Community Detection vs. Graph Partitioning

Undirected graph $G(V, E)$

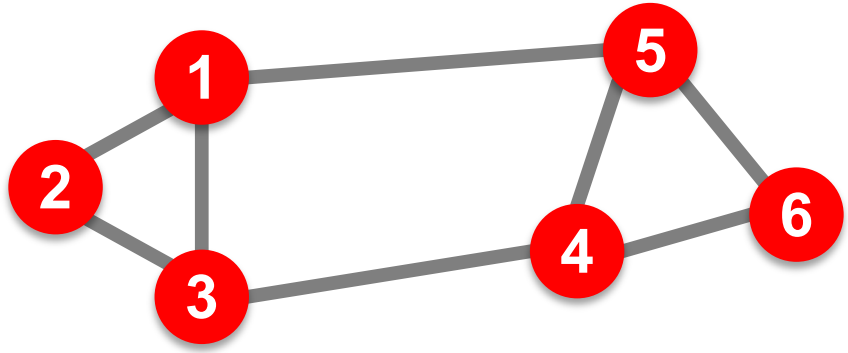


- Bi-partitioning task: divide vertices into two disjoint groups

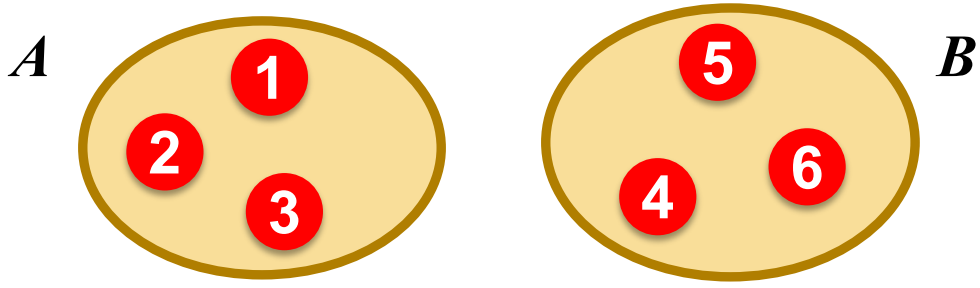
- How can we define a "good" partition of G?
- How can we efficiently identify such a partition?

Community Detection vs. Graph Partitioning

Undirected graph $G(V, E)$



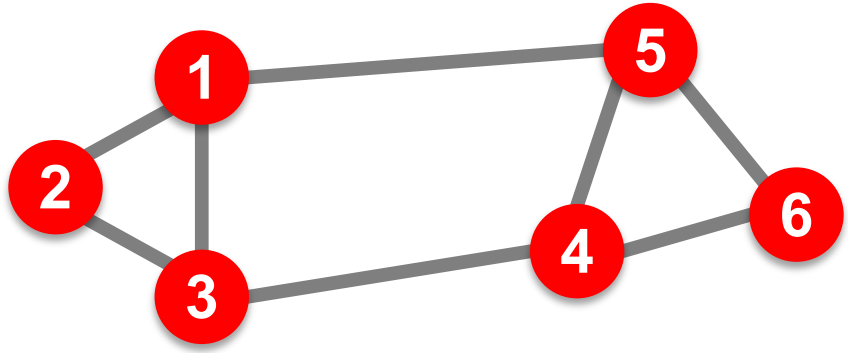
- Bi-partitioning task: divide vertices into two disjoint groups



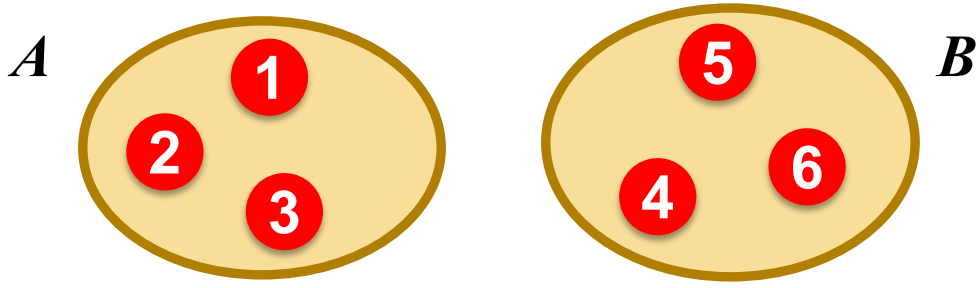
- How can we define a "good" partition of G?
- How can we efficiently identify such a partition?

Community Detection vs. Graph Partitioning

Undirected graph $G(V, E)$



- Bi-partitioning task: divide vertices into two disjoint groups

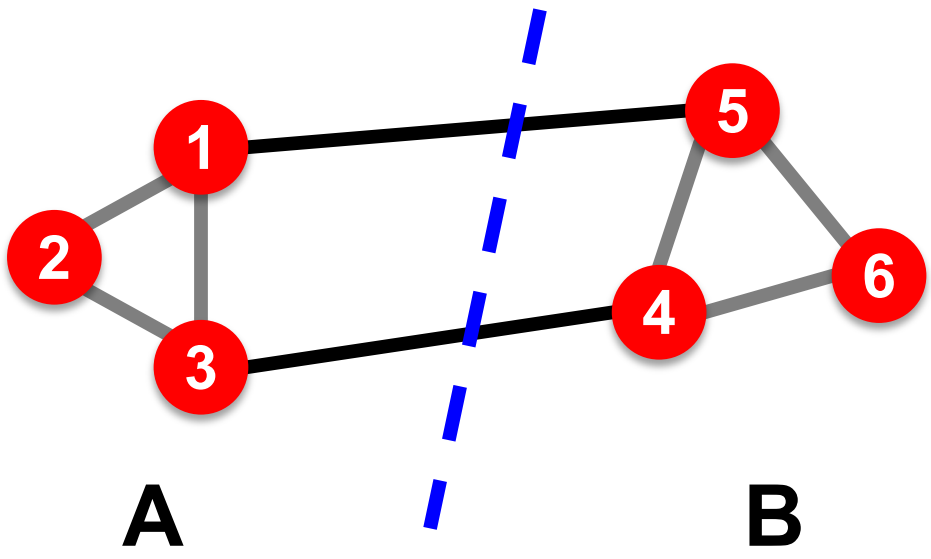


- How can we define a "good" partition of G?
- How can we efficiently identify such a partition?

Community Detection vs. Graph Partitioning

What makes a good partition?

- Maximise the number of within-group connections
- Minimise the number of between-group connections



How to define a formula to capture these?

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

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

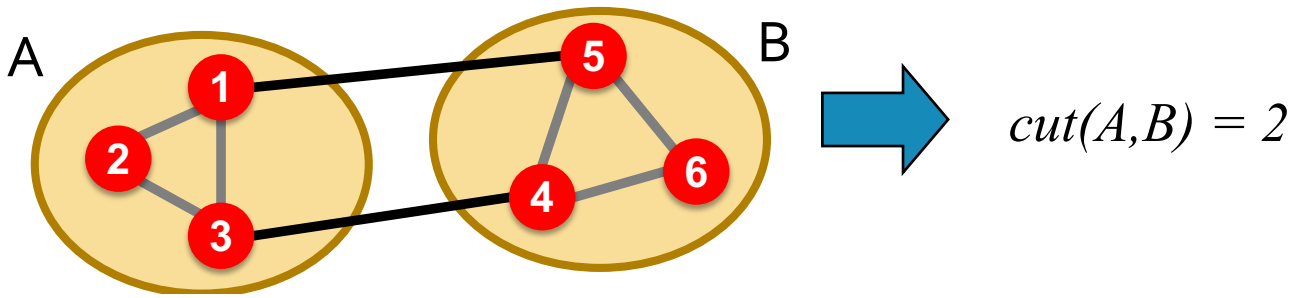
If the graph is weighted, w_{ij} is the weight. Otherwise, $w_{ij} = \{0, 1\}$.

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, $w_{ij} = \{0, 1\}$.



Graph Cut Criterion

- Minimum-cut: minimise weight of connections between groups

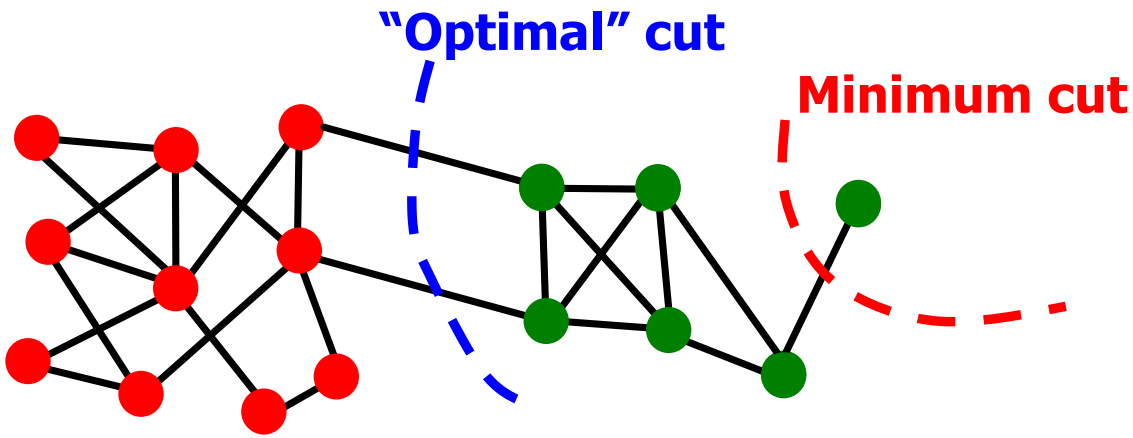
$$\arg\min_{A,B} cut(A, B)$$

Graph Cut Criterion

- Minimum-cut: minimise weight of connections between groups

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

- Degenerate case:

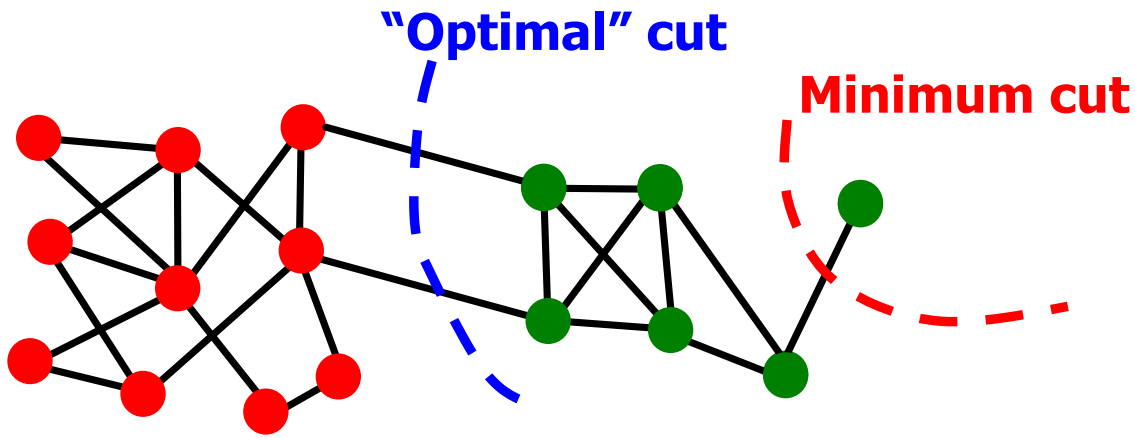


Graph Cut Criterion

- Minimum-cut: minimise weight of connections between groups

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

- Degenerate case:



- Problem: only consider external cluster connections, but not internal ones

Graph Cut Criterion

- Conductance: connectivity between groups relative to the density of each group

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

$vol(A) = \sum_{i \in A} k_i$: total (weighted) degree of the nodes in A.

- This produces more balanced partitions (i.e., $\min(vol(A), vol(B))$)

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

- How can we efficiently identify a good partition?
  (Computing the best conductance cut is NP-hard.)
- An algorithm (i.e., spectral graph partitioning) to approximate

Graph Cut Criterion

- Conductance: connectivity between groups relative to the density of each group

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

$vol(A) = \sum_{i \in A} k_i$: total (weighted) degree of the nodes in A.

- This produces more balanced partitions (i.e., $\min(vol(A), vol(B))$)

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

- How can we efficiently identify a good partition?
  (Computing the best conductance cut is NP-hard.)
- An algorithm (i.e., spectral graph partitioning) to approximate

Graph Cut Criterion

- Conductance: connectivity between groups relative to the density of each group

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

$vol(A) = \sum_{i \in A} k_i$: total (weighted) degree of the nodes in A.

- This produces more balanced partitions (i.e., $\min(vol(A), vol(B))$)

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

- How can we efficiently identify a good partition?
  (Computing the best conductance cut is NP-hard.)
- An algorithm (i.e., spectral graph partitioning) to approximate

Spectral Graph Partitioning

- A, the adjacency matrix of undirected graph G:
  $A_{ij} = 1$ if (i, j) is an edge of G, $A_{ij} = 0$ otherwise
- x is a vector in $\mathcal{R}^n$ with components $(x_1, \ldots, x_n)$
  (taking it as a label/value of each node in G)
- What is the meaning of $A \cdot x$?

- Each entry y_i is a sum of labels x_j of neighbours of i

Spectral Graph Partitioning

- A, the adjacency matrix of undirected graph G:
  $A_{ij} = 1$ if (i, j) is an edge of G, $A_{ij} = 0$ otherwise
- x is a vector in $\mathcal{R}^n$ with components $(x_1, \ldots, x_n)$
  (taking it as a label/value of each node in G)
- What is the meaning of $A \cdot x$?

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

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

- Each entry y_i is a sum of labels x_j of neighbours of i

Spectral Graph Partitioning

- A, the adjacency matrix of undirected graph G:
  $A_{ij} = 1$ if (i, j) is an edge of G, $A_{ij} = 0$ otherwise
- x is a vector in $\mathcal{R}^n$ with components $(x_1, \ldots, x_n)$
  (taking it as a label/value of each node in G)
- What is the meaning of $A \cdot x$?

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

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

- Each entry y_i is a sum of labels x_j of neighbours of i

Spectral Graph Partitioning

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

- j^{th} coordinate of $A \cdot x$: sum of the x-values of neighbours of j, and make this a new x-value at node j

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

$$A \cdot x = \lambda \cdot x$$

- Spectral graph theory:
  - Analyse the "spectrum" of matrix representing G
  - Spectrum: eigenvectors $x^{(i)}$ of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues λ_i

$$\{\lambda_1 \leq \lambda_2 \leq \lambda_3 \cdots \leq \lambda_n\}$$

Spectral Graph Partitioning

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

- j^{th} coordinate of $A \cdot x$: sum of the x-values of neighbours of j, and make this a new x-value at node j

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

$$A \cdot x = \lambda \cdot x$$

- Spectral graph theory:
  - Analyse the "spectrum" of matrix representing G
  - Spectrum: eigenvectors $x^{(i)}$ of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues λ_i

$$\{\lambda_1 \leq \lambda_2 \leq \lambda_3 \cdots \leq \lambda_n\}$$

Spectral Graph Partitioning: More Intuition

Motivating example: d-regular graph

- Suppose all nodes in G have degree d
  (G is d-regular), and G is connected
- What are eigenvalues/vectors of G?

$$A \cdot x = \lambda \cdot x, \text{ what are } \lambda \text{ and } x?$$

- Let us try: $x = (1, 1, \ldots, 1)$
- Then, $A \cdot x = (d, d, \cdots, d) = \lambda \cdot x$. So, we have $\lambda = d$.
- We found one eigenpair of G: $x = (1, 1, \ldots, 1)$, $\lambda = d$
- d is the largest eigenvalue of A (search for a proof!)
- An $n \times n$ matrix can have up to n eigenpairs.

Spectral Graph Partitioning: More Intuition

Motivating example: d-regular graph

- Suppose all nodes in G have degree d
  (G is d-regular), and G is connected
- What are eigenvalues/vectors of G?

$$A \cdot x = \lambda \cdot x, \text{ what are } \lambda \text{ and } x?$$

  - Let us try: $x = (1, 1, \ldots, 1)$
  - Then, $A \cdot x = (d, d, \cdots, d) = \lambda \cdot x$. So, we have $\lambda = d$.
  - We found one eigenpair of G: $x = (1, 1, \ldots, 1)$, $\lambda = d$
  - d is the largest eigenvalue of A (search for a proof!)
  - An $n \times n$ matrix can have up to n eigenpairs.

Spectral Graph Partitioning: More Intuition

Motivating example: d-regular graph

- Suppose all nodes in G have degree d
  (G is d-regular), and G is connected
- What are eigenvalues/vectors of G?

$$A \cdot x = \lambda \cdot x, \text{ what are } \lambda \text{ and } x?$$

  - Let us try: $x = (1, 1, \ldots, 1)$
  - Then, $A \cdot x = (d, d, \cdots, d) = \lambda \cdot x$. So, we have $\lambda = d$.
  - We found one eigenpair of G: $x = (1, 1, \ldots, 1)$, $\lambda = d$
  - d is the largest eigenvalue of A (search for a proof!)
  - An $n \times n$ matrix can have up to n eigenpairs.

Spectral Graph Partitioning: More Intuition

Motivating example: d-regular graph

- Suppose all nodes in G have degree d
  (G is d-regular), and G is connected
- What are eigenvalues/vectors of G?

$$A \cdot x = \lambda \cdot x, \text{ what are } \lambda \text{ and } x?$$

  - Let us try: $x = (1, 1, \ldots, 1)$
  - Then, $A \cdot x = (d, d, \cdots, d) = \lambda \cdot x$. So, we have $\lambda = d$.
  - We found one eigenpair of G: $x = (1, 1, \ldots, 1)$, $\lambda = d$
  - d is the largest eigenvalue of A (search for a proof!)
  - An $n \times n$ matrix can have up to n eigenpairs.

Spectral Graph Partitioning: More Intuition

Motivating example: d-regular graph

- Suppose all nodes in G have degree d
  (G is d-regular), and G is connected
- What are eigenvalues/vectors of G?

$$A \cdot x = \lambda \cdot x, \text{ what are } \lambda \text{ and } x?$$

  - Let us try: $x = (1, 1, \ldots, 1)$
  - Then, $A \cdot x = (d, d, \cdots, d) = \lambda \cdot x$. So, we have $\lambda = d$.
  - We found one eigenpair of G: $x = (1, 1, \ldots, 1)$, $\lambda = d$
  - d is the largest eigenvalue of A (search for a proof!)
  - An $n \times n$ matrix can have up to n eigenpairs.

Spectral Graph Partitioning: More Intuition

Motivating example: d-regular graph

- Suppose all nodes in G have degree d
  (G is d-regular), and G is connected
- What are eigenvalues/vectors of G?

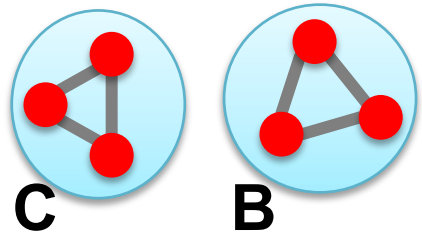
$$A \cdot x = \lambda \cdot x, \text{ what are } \lambda \text{ and } x?$$

  - Let us try: $x = (1, 1, \ldots, 1)$
  - Then, $A \cdot x = (d, d, \cdots, d) = \lambda \cdot x$. So, we have $\lambda = d$.
  - We found one eigenpair of G: $x = (1, 1, \ldots, 1)$, $\lambda = d$
  - d is the largest eigenvalue of A (search for a proof!)
  - An $n \times n$ matrix can have up to n eigenpairs.

Spectral Graph Partitioning: More Intuition

Motivating example: What if G is not connected?

G has two components, each is d-regular.



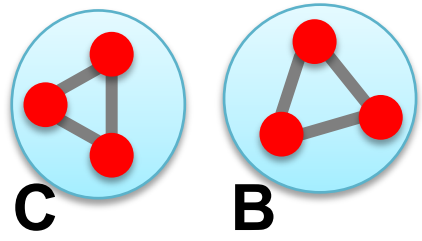
What are eigenvalues/vectors of G?

- x = Put all 1s on C and 0s on B, or vice versa
- $x' = (1, \ldots, 1, 0, \ldots, 0)$, then $A \cdot x' = (d, \ldots, d, 0, \ldots, 0)$
- $x'' = (0, \ldots, 0, 1, \ldots, 1)$, then $A \cdot x'' = (0, \ldots, 0, d, \ldots, d)$
- In both cases, the corresponding $\lambda = d$ $(\lambda_n = \lambda_{n-1} = d)$

Spectral Graph Partitioning: More Intuition

Motivating example: What if G is not connected?

G has two components, each is d-regular.



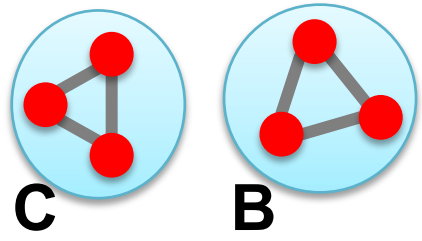
What are eigenvalues/vectors of G?

- x = Put all 1s on C and 0s on B, or vice versa
- $x' = (1, \ldots, 1, 0, \ldots, 0)$, then $A \cdot x' = (d, \ldots, d, 0, \ldots, 0)$
- $x'' = (0, \ldots, 0, 1, \ldots, 1)$, then $A \cdot x'' = (0, \ldots, 0, d, \ldots, d)$
- In both cases, the corresponding $\lambda = d$ $(\lambda_n = \lambda_{n-1} = d)$

Spectral Graph Partitioning: More Intuition

Motivating example: What if G is not connected?

G has two components, each is d-regular.



What are eigenvalues/vectors of G?

- x = Put all 1s on C and 0s on B, or vice versa
- $x' = (1, \ldots, 1, 0, \ldots, 0)$, then $A \cdot x' = (d, \ldots, d, 0, \ldots, 0)$
- $x'' = (0, \ldots, 0, 1, \ldots, 1)$, then $A \cdot x'' = (0, \ldots, 0, d, \ldots, d)$
- In both cases, the corresponding $\lambda = d$ $(\lambda_n = \lambda_{n-1} = d)$

Spectral Graph Partitioning: More Intuition

- If the graph is unconnected (left example), we have $\lambda_n = \lambda_{n-1}$
- If the graph is badly connected (right example), the 2nd largest eigenvalue λ_{n-1} has a value very close to λ_n.
- Eigenvectors are orthogonal, so the components of 2nd eigenvector x_{n-1} must sum to 0.

$$x_n \cdot x_{n-1} = 0, \text{ then } \sum_i (x_n[i] \cdot x_{n-1}[i]) = \sum_i x_{n-1}[j] = 0$$

- x_{n-1} "splits" the nodes into two groups ($x_{n-1}[i] > 0$ vs. $x_{n-1}[i] < 0$)
- We could look at the eigenvector of the 2nd largest eigenvalue and declare nodes with positive (negative) values in C (B).

Spectral Graph Partitioning: More Intuition

- If the graph is unconnected (left example), we have $\lambda_n = \lambda_{n-1}$
- If the graph is badly connected (right example), the 2nd largest eigenvalue λ_{n-1} has a value very close to λ_n.
- Eigenvectors are orthogonal, so the components of 2nd eigenvector x_{n-1} must sum to 0.

$$x_n \cdot x_{n-1} = 0, \text{ then } \sum_i (x_n[i] \cdot x_{n-1}[i]) = \sum_i x_{n-1}[j] = 0$$

- x_{n-1} "splits" the nodes into two groups ($x_{n-1}[i] > 0$ vs. $x_{n-1}[i] < 0$)
- We could look at the eigenvector of the 2nd largest eigenvalue and declare nodes with positive (negative) values in C (B).

Spectral Graph Partitioning: More Intuition

- If the graph is unconnected (left example), we have $\lambda_n = \lambda_{n-1}$
- If the graph is badly connected (right example), the 2nd largest eigenvalue λ_{n-1} has a value very close to λ_n.
- Eigenvectors are orthogonal, so the components of 2nd eigenvector x_{n-1} must sum to 0.

$$x_n \cdot x_{n-1} = 0, \text{ then } \sum_i (x_n[i] \cdot x_{n-1}[i]) = \sum_i x_{n-1}[j] = 0$$

- x_{n-1} "splits" the nodes into two groups ($x_{n-1}[i] > 0$ vs. $x_{n-1}[i] < 0$)
- We could look at the eigenvector of the 2nd largest eigenvalue and declare nodes with positive (negative) values in C (B).

Spectral Graph Partitioning: More Intuition

- If the graph is unconnected (left example), we have $\lambda_n = \lambda_{n-1}$
- If the graph is badly connected (right example), the 2nd largest eigenvalue λ_{n-1} has a value very close to λ_n.
- Eigenvectors are orthogonal, so the components of 2nd eigenvector x_{n-1} must sum to 0.

$$x_n \cdot x_{n-1} = 0, \text{ then } \sum_i (x_n[i] \cdot x_{n-1}[i]) = \sum_i x_{n-1}[j] = 0$$

- x_{n-1} "splits" the nodes into two groups ($x_{n-1}[i] > 0$ vs. $x_{n-1}[i] < 0$)
- We could look at the eigenvector of the 2nd largest eigenvalue and declare nodes with positive (negative) values in C (B).

Spectral Graph Partitioning: More Intuition

- If the graph is unconnected (left example), we have $\lambda_n = \lambda_{n-1}$
- If the graph is badly connected (right example), the 2nd largest eigenvalue λ_{n-1} has a value very close to λ_n.
- Eigenvectors are orthogonal, so the components of 2nd eigenvector x_{n-1} must sum to 0.

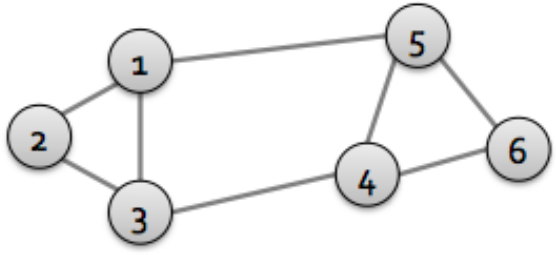
$$x_n \cdot x_{n-1} = 0, \text{ then } \sum_i (x_n[i] \cdot x_{n-1}[i]) = \sum_i x_{n-1}[j] = 0$$

- x_{n-1} "splits" the nodes into two groups ($x_{n-1}[i] > 0$ vs. $x_{n-1}[i] < 0$)
- We could look at the eigenvector of the 2nd largest eigenvalue and declare nodes with positive (negative) values in C (B).

Matrix Representations

Adjacency matrix A:

- $n \times n$ matrix
- $A = [A_{ij}]$, $A_{ij} = 1$ if there is an edge between nodes i and j



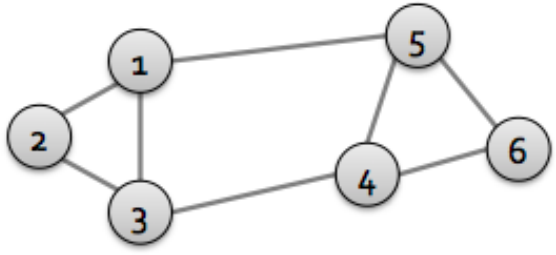
	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

- Important properties: symmetric matrix, n real eigenvalues, eigenvectors are real-valued and orthogonal

Matrix Representations

Adjacency matrix A:

- $n \times n$ matrix
- $A = [A_{ij}]$, $A_{ij} = 1$ if there is an edge between nodes i and j



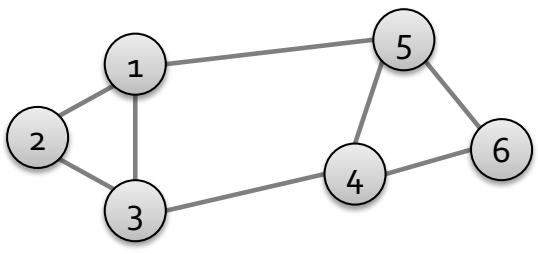
	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

- Important properties: symmetric matrix, 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

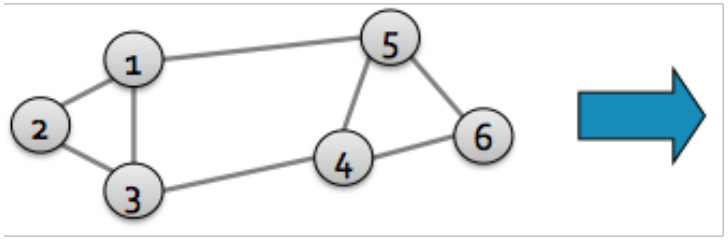


	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

Matrix Representations

Laplacian matrix $L = D - A$:

- $n \times n$ 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

- What is trivial eigenpair?

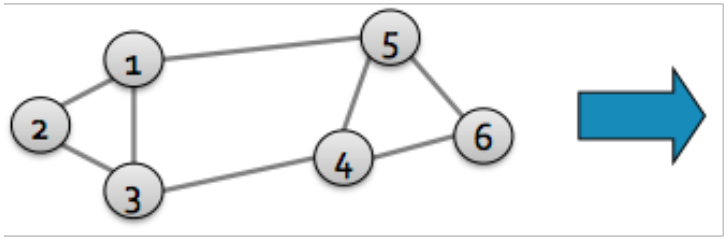
  $x = (1, \ldots, 1)$, then $L \cdot x = 0$ and so $\lambda = \lambda_1 = 0$ (smallest)

- Important properties: eigenvalues are non-negative real numbers, eigenvectors are real-valued and orthogonal

Matrix Representations

Laplacian matrix $L = D - A$:

- $n \times n$ 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

- What is trivial eigenpair?

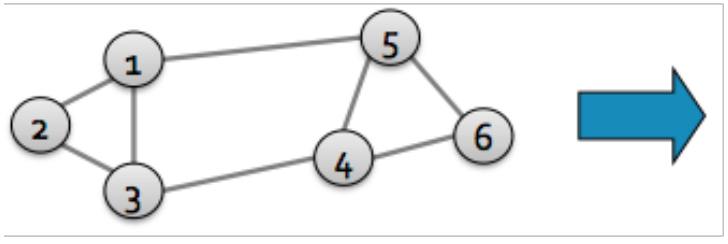
  $x = (1, \ldots, 1)$, then $L \cdot x = 0$ and so $\lambda = \lambda_1 = 0$ (smallest)

- Important properties: eigenvalues are non-negative real numbers, eigenvectors are real-valued and orthogonal

Matrix Representations

Laplacian matrix $L = D - A$:

- $n \times n$ 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

- What is trivial eigenpair?

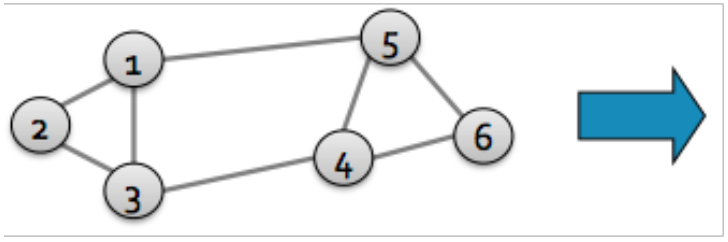
  $x = (1, \ldots, 1)$, then $L \cdot x = 0$ and so $\lambda = \lambda_1 = 0$ (smallest)

- Important properties: eigenvalues are non-negative real numbers, eigenvectors are real-valued and orthogonal

Matrix Representations

Laplacian matrix $L = D - A$:

- $n \times n$ 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

- What is trivial eigenpair?

  $x = (1, \ldots, 1)$, then $L \cdot x = 0$ and so $\lambda = \lambda_1 = 0$ (smallest)

- Important properties: eigenvalues are non-negative real numbers, eigenvectors are real-valued and orthogonal

λ_2 as Optimisation Problem

For any symmetric matrix M:

the second smallest eigenvalue is a solution to the following optimisation problem

$$\lambda_2 = \min_x \frac{x^T \cdot M \cdot x}{x^T x}$$

What is the meaning of $\min\ x^T \cdot L \cdot x$ on G?

$$\begin{aligned} x^T \cdot L \cdot x &= \sum_{i,j}^n L_{ij} x_i x_j = \sum_{i,j}^n (D_{ij} - A_{ij}) x_i x_j \\ &= \sum_i D_{ii} x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j \\ &= \sum_{(i,j)\in E} (x_i^2 + x_j^2 - 2 x_i x_j) \\ &= \sum_{(i,j)\in E} (x_i - x_j)^2 \end{aligned}$$

x_i^2 needs to be summed up D_{ii} times. Edge (i,j) has two end points, so $x_i^2 + x_j^2$.

λ_2 as Optimisation Problem

For any symmetric matrix M:

the second smallest eigenvalue is a solution to the following optimisation problem

$$\lambda_2 = \min_x \frac{x^T \cdot M \cdot x}{x^T x}$$

What is the meaning of $\min x^T \cdot L \cdot x$ on G?

$$\begin{aligned} x^T \cdot L \cdot x &= \sum_{i,j}^n L_{ij} x_i x_j = \sum_{i,j}^n (D_{ij} - A_{ij}) x_i x_j \\ &= \sum_i D_{ii} x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j \\ &= \sum_{(i,j)\in E} (x_i^2 + x_j^2 - 2 x_i x_j) \\ &= \sum_{(i,j)\in E} (x_i - x_j)^2 \end{aligned}$$

x_i^2 needs to be summed up D_{ii} times. Edge (i,j) has two end points, so $x_i^2 + x_j^2$.

λ_2 as Optimisation Problem

For any symmetric matrix M:

the second smallest eigenvalue is a solution to the following optimisation problem

$$\lambda_2 = \min_x \frac{x^T \cdot M \cdot x}{x^T x}$$

What is the meaning of min $x^T \cdot L \cdot x$ on G?

$$\begin{aligned} x^T \cdot L \cdot x &= \sum_{i,j}^n L_{ij} x_i x_j = \sum_{i,j}^n (D_{ij} - A_{ij}) x_i x_j \\ &= \sum_i D_{ii} x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j \\ &= \sum_{(i,j)\in E} (x_i^2 + x_j^2 - 2 x_i x_j) \\ &= \sum_{(i,j)\in E} (x_i - x_j)^2 \end{aligned}$$

x_i^2 needs to be summed up D_{ii} times. Edge (i,j) has two end points, so $x_i^2 + x_j^2$.

λ_2 as Optimisation Problem

We also know that

- x is (chosen to be) a unit vector, i.e., $\sum_i x_i^2 = 1$
- x is orthogonal to $(1, \ldots, 1)$, $\sum_i x_i \cdot 1 = \sum_i x_i = 0$

$$\lambda_2 = \min_{x:\sum_i x_i=0} \frac{\sum_{(i,j)\in E}(x_i - x_j)^2}{\sum_i x_i^2} = \min_{x:\sum_i x_i=0} \sum_{(i,j)\in E} (x_i - x_j)^2$$

λ_2 as Optimisation Problem

We also know that

- x is (chosen to be) a unit vector, i.e., $\sum_i x_i^2 = 1$
- x is orthogonal to $(1, \ldots, 1)$, $\sum_i x_i \cdot 1 = \sum_i x_i = 0$

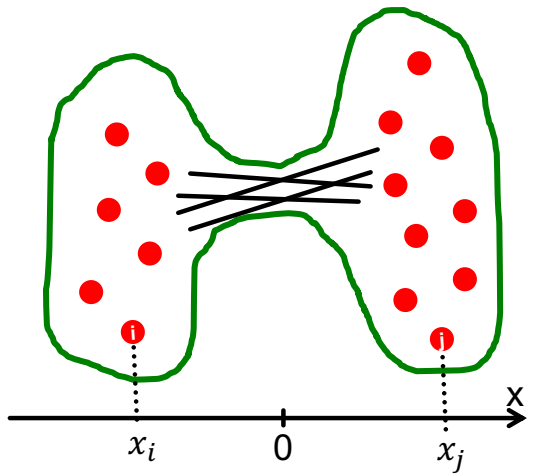
$$\lambda_2 = \min_{x:\sum_i x_i=0} \frac{\sum_{(i,j)\in E}(x_i - x_j)^2}{\sum_i x_i^2} = \min_{x:\sum_i x_i=0} \sum_{(i,j)\in E} (x_i - x_j)^2$$

λ_2 as Optimisation Problem

We also know that

- x is (chosen to be) a unit vector, i.e., $\sum_i x_i^2 = 1$
- x is orthogonal to $(1, \ldots, 1)$, $\sum_i x_i \cdot 1 = \sum_i x_i = 0$

$$\lambda_2 = \min_{x:\sum_i x_i=0} \frac{\sum_{(i,j)\in E}(x_i - x_j)^2}{\sum_i x_i^2} = \min_{x:\sum_i x_i=0} \sum_{(i,j)\in E} (x_i - x_j)^2$$



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]

- Optimal cut: expressing partition (A, B) as a vector $y_i = +1$ if $i \in A$, $y_i = -1$ if $i \in B$
- We can minimise the cut of the partition by finding a vector y that minimises:

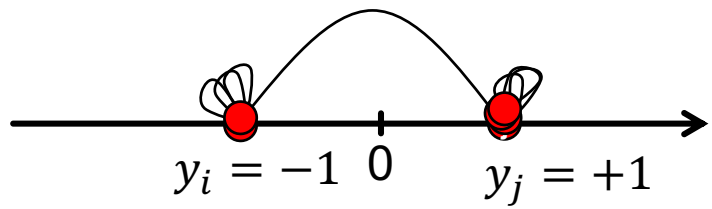
$$\arg\min_{y \in \{1,+1\}^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2$$

- Cannot solve exactly, we need to relax y and allow it to take any real value.

Find Optimal Cut [Fiedler'73]

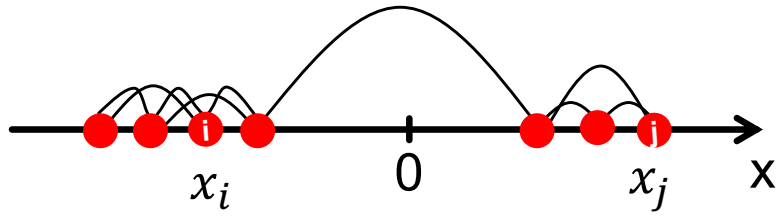
- Optimal cut: expressing partition (A, B) as a vector $y_i = +1$ if $i \in A$, $y_i = -1$ if $i \in B$
- We can minimise the cut of the partition by finding a vector y that minimises:

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



- Cannot solve exactly, we need to relax y and allow it to take any real value.

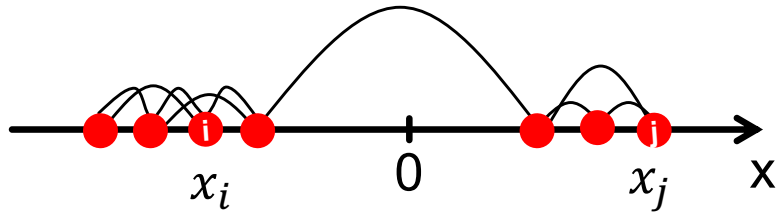
Find Optimal Cut [Fiedler'73] & Rayleigh Theorem



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

- The minimum value of $f(y)$ is given by the 2nd smallest eigenvalue λ_2 of the Laplacian matrix
- The optimal solution for y is given by the corresponding eigenvector x, referred to as the Fiedler vector
- Use sign of x_i to determine cluster assignment of node i

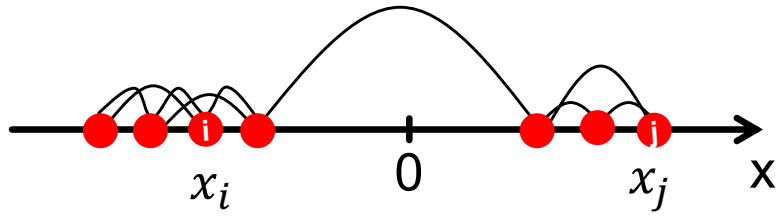
Find Optimal Cut [Fiedler'73] & Rayleigh Theorem



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

- The minimum value of $f(y)$ is given by the 2nd smallest eigenvalue λ_2 of the Laplacian matrix
- The optimal solution for y is given by the corresponding eigenvector x, referred to as the Fiedler vector
- Use sign of x_i to determine cluster assignment of node i

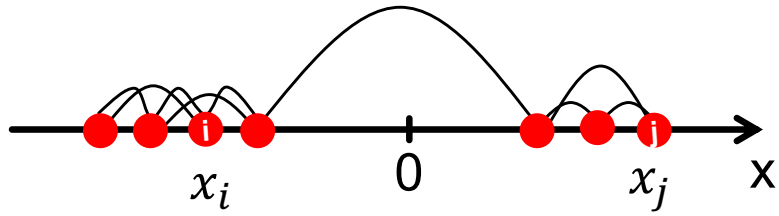
Find Optimal Cut [Fiedler'73] & Rayleigh Theorem



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

- The minimum value of $f(y)$ is given by the 2nd smallest eigenvalue λ_2 of the Laplacian matrix
- The optimal solution for y is given by the corresponding eigenvector x, referred to as the Fiedler vector
- Use sign of x_i to determine cluster assignment of node i

Find Optimal Cut [Fiedler'73] & Rayleigh Theorem



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

- The minimum value of $f(y)$ is given by the 2nd smallest eigenvalue λ_2 of the Laplacian matrix
- The optimal solution for y is given by the corresponding eigenvector x, referred to as the Fiedler vector
- Use sign of x_i to determine cluster assignment of node i

Graph Partitioning

- Question: how to define a "good" partition of a graph?

  Answer: minimise a graph cut criterion (e.g., "conductance").

- Question: how to efficiently identify such a "good" partition?

  Answer: approximate using information provided by the eigenvalues and eigenvectors of a graph.

- Spectral clustering!

Graph Partitioning

- Question: how to define a "good" partition of a graph?

  Answer: minimise a graph cut criterion (e.g., "conductance").

- Question: how to efficiently identify such a "good" partition?

  Answer: approximate using information provided by the eigenvalues and eigenvectors of a graph.

- Spectral clustering!

Graph Partitioning

- Question: how to define a "good" partition of a graph?

  Answer: minimise a graph cut criterion (e.g., "conductance").

- Question: how to efficiently identify such a "good" partition?

  Answer: approximate using information provided by the eigenvalues and eigenvectors of a graph.

- Spectral clustering!

Approximate Guarantee of Spectral

Connecting λ_2 with graph partitioning:

- Suppose there is a partition of G into A and B with $|A| \leq |B|$, such that "conductance" of the cut (A, B) is α, then

$$\lambda_2 \leq 2\alpha$$

- 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 α.

Approximate Guarantee of Spectral

Connecting λ_2 with graph partitioning:

- Suppose there is a partition of G into A and B with $|A| \leq |B|$, such that "conductance" of the cut (A, B) is α, then

$$\lambda_2 \leq 2\alpha$$

- 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 α.

Spectral Clustering

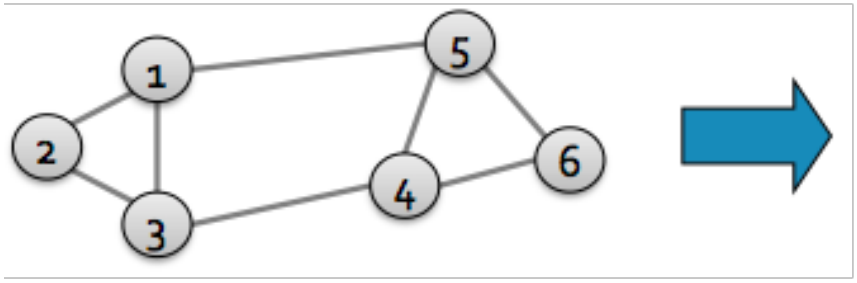
Spectral Clustering Algorithm

Three basic steps:

1. Pre-processing: construct a matrix representation of the graph
2. Decomposition:
   - compute eigenvalues and eigenvectors of the matrix
   - map each node to a lower-dimensional representation based on one (or more) eigenvector(s)
3. Grouping: assign nodes to two (or more) clusters, based on the new representation representation

Spectral Clustering 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

Spectral Clustering Algorithm

Decomposition: find eigenvalues λ and eigenvectors x of the matrix L (especially λ_2 and x_2)

$\lambda =$

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

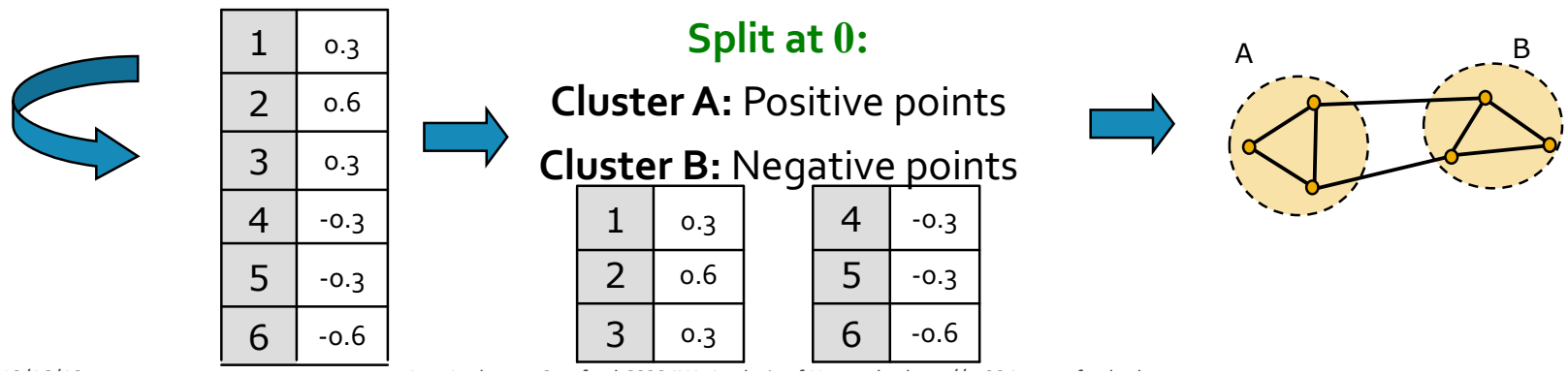
Decomposition: map vertices to corresponding components of x_2

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

Spectral Clustering Algorithm

Grouping:

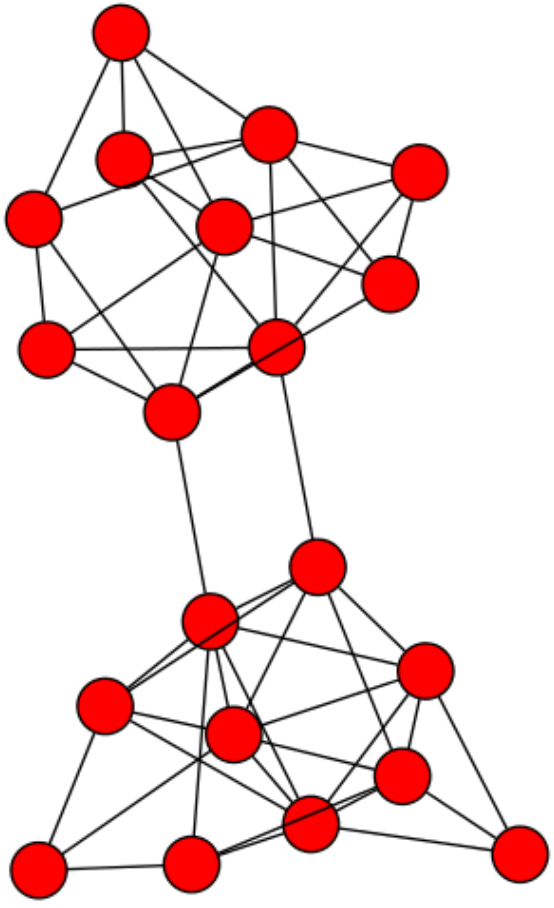
- sort components of reduced 1-dimensional vector
- identify clusters by splitting the sorted vector into two

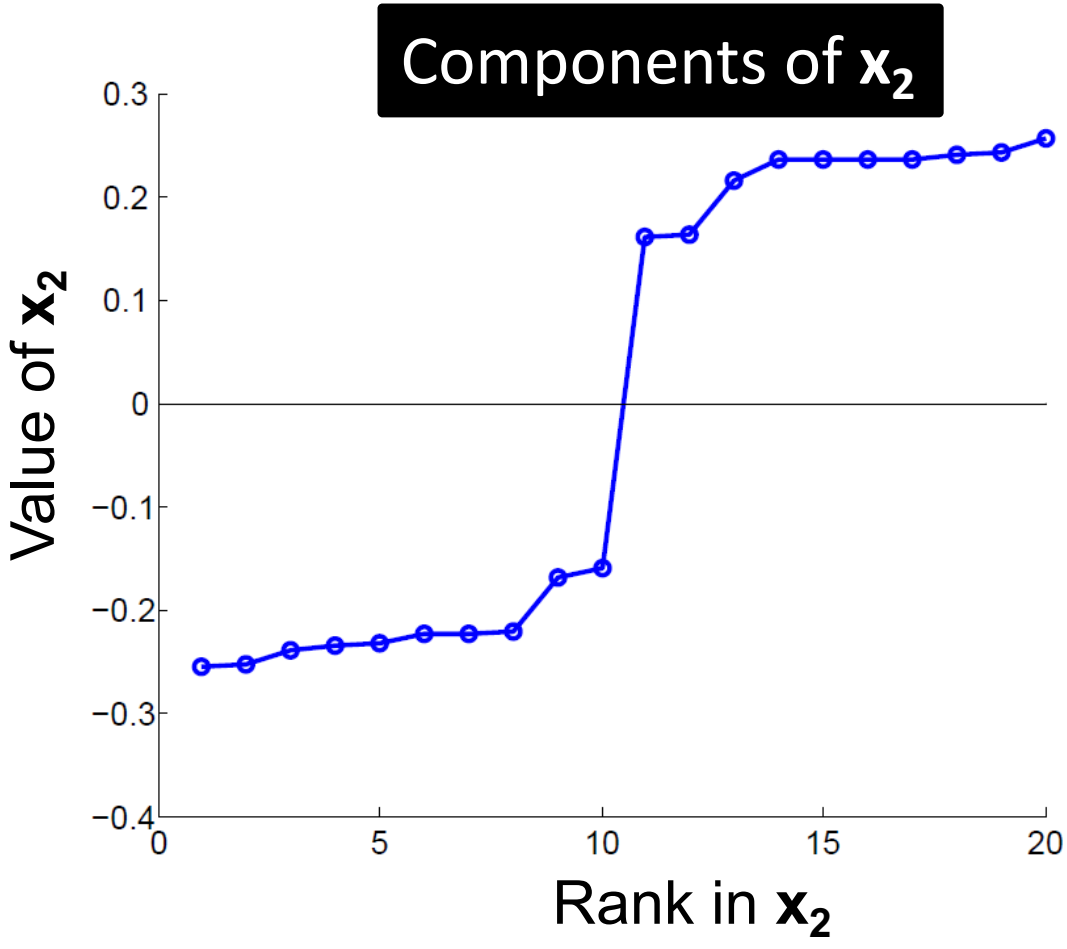


- Naïve approach: split at 0 or median value

Spectral Clustering Algorithm

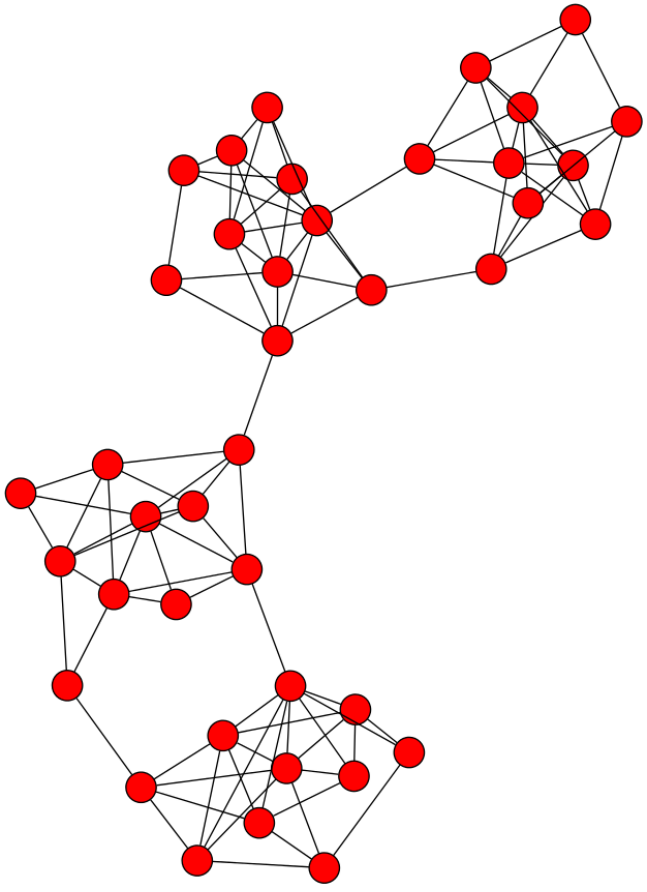
Magic works!

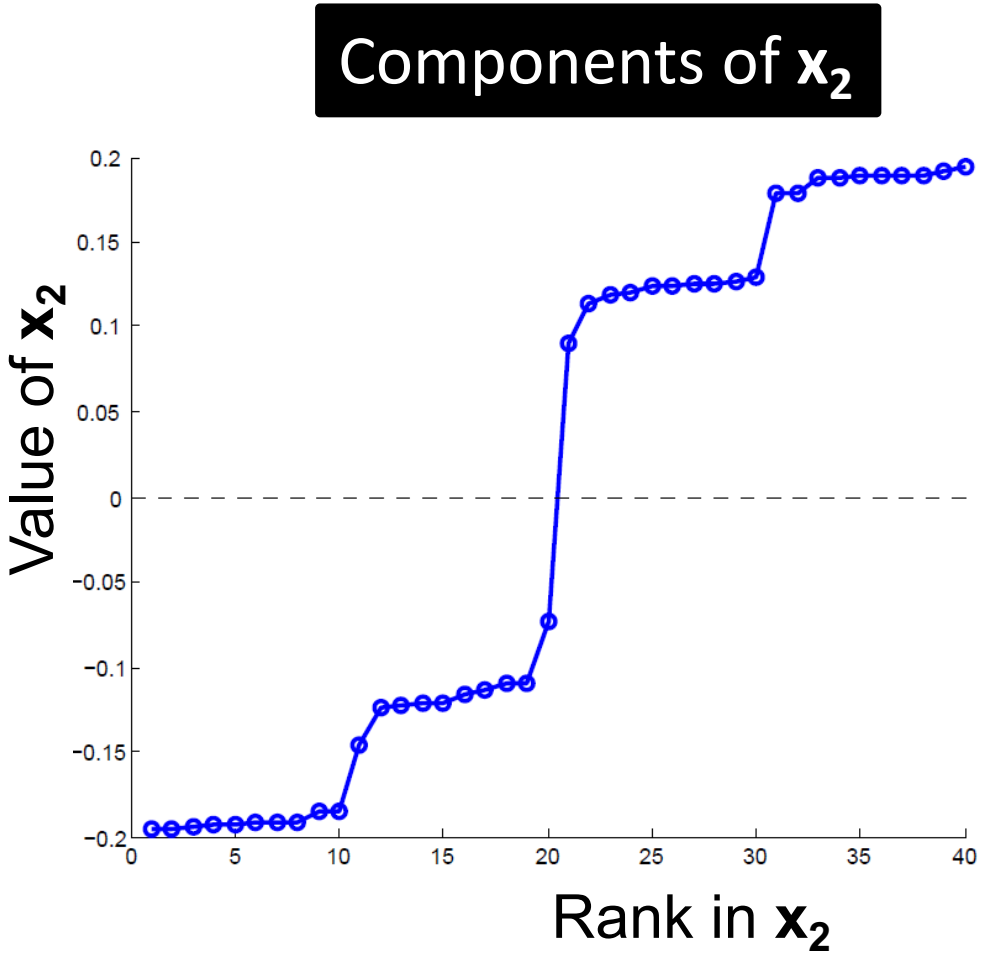




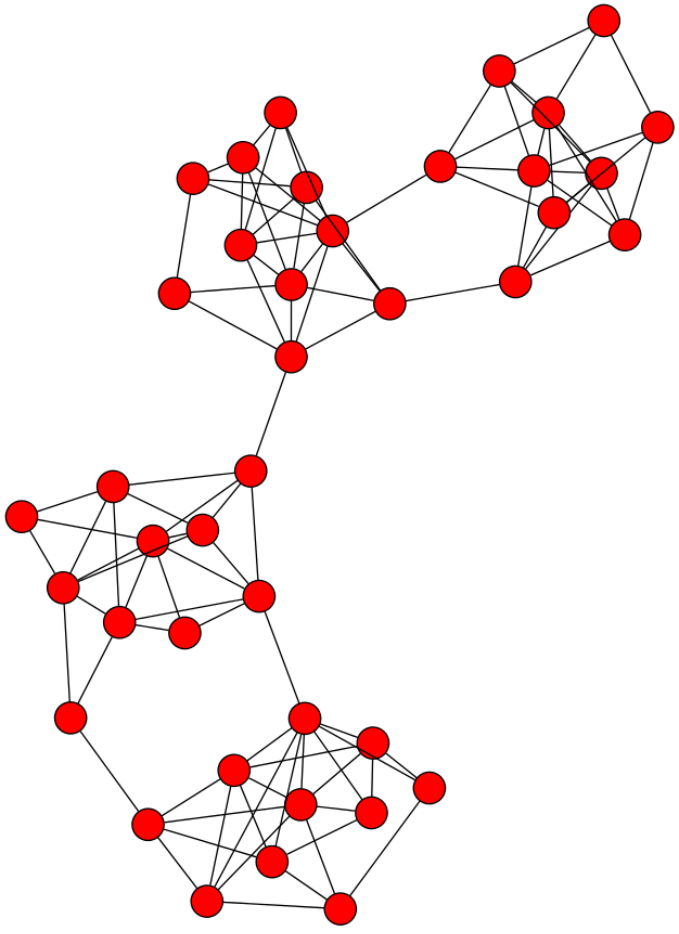
Spectral Clustering Algorithm

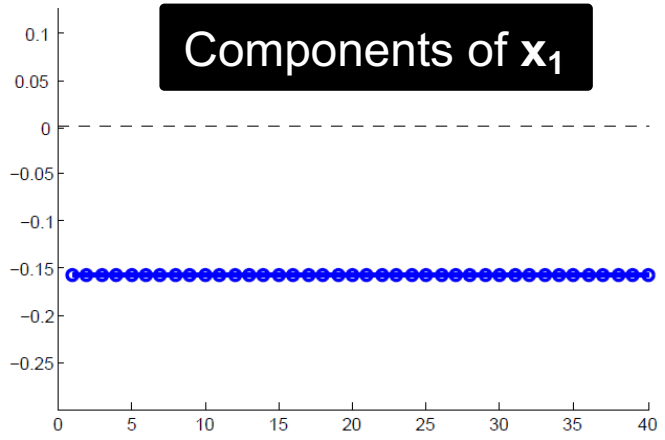
Magic works!

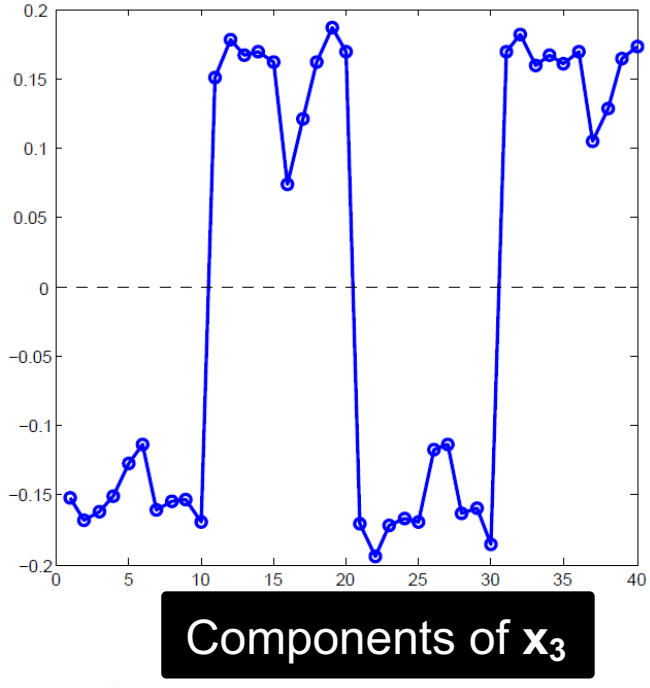




Spectral Clustering Algorithm







Spectral Clustering Algorithm

How do we partition a graph into k clusters?

- Recursive bi-partitioning: apply the bi-partitioning algorithm in a hierarchical divisive manner, but inefficient and unstable
- Cluster multiple eigenvectors: build a reduced space from multiple eigenvectors.

  Each node is now represented by k numbers, then cluster the nodes based on their k-dimension representation.

Spectral Clustering Algorithm

How do we partition a graph into k clusters?

- Recursive bi-partitioning: apply the bi-partitioning algorithm in a hierarchical divisive manner, but inefficient and unstable
- Cluster multiple eigenvectors: build a reduced space from multiple eigenvectors.

  Each node is now represented by k numbers, then cluster the nodes based on their k-dimension representation.

Spectral Clustering Algorithm

How do we partition a graph into k clusters?

- Recursive bi-partitioning: apply the bi-partitioning algorithm in a hierarchical divisive manner, but inefficient and unstable
- Cluster multiple eigenvectors: build a reduced space from multiple eigenvectors.

  Each node is now represented by k numbers, then cluster the nodes based on their k-dimension representation.

Clique Percolation Method

Overlapping Communities

- In real networks, communities are often overlapping
  - Some of high-school friends might be also university friends
  - A colleague might be a family member
- Overlapping community detection is considered much harder
- Many algorithms: adaptations of modularity
- Many local methods, unlike global optimisation for non-overlapping methods

Overlapping Communities

- In real networks, communities are often overlapping
  - Some of high-school friends might be also university friends
  - A colleague might be a family member
- Overlapping community detection is considered much harder
- Many algorithms: adaptations of modularity
- Many local methods, unlike global optimisation for non-overlapping methods

Clique Percolation Method [Palla et al., 2005]

Main idea:

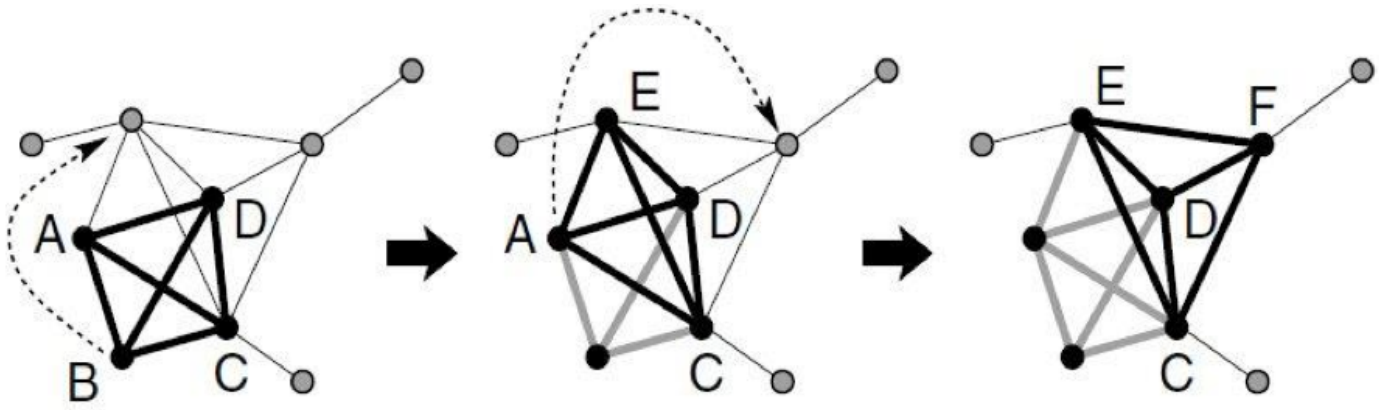
- Internal edges of community likely to form clique
- Inter-community edges unlikely to form clique

Clique Percolation Method [Palla et al., 2005]

- k-clique: complete subgraph on k nodes
- Adjacent k-cliques: two k-cliques that share $k-1$ nodes
- Module: union of adjacent k-cliques

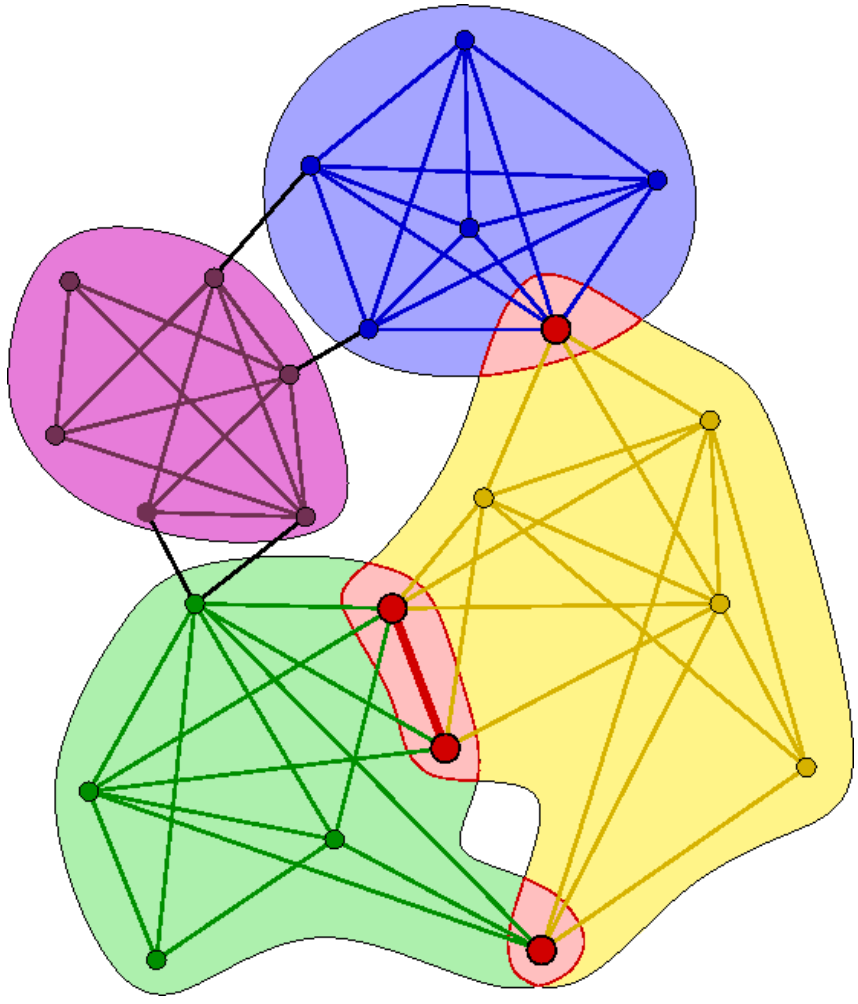
Clique Percolation Method [Palla et al., 2005]

- k-clique: complete subgraph on k nodes
- Adjacent k-cliques: two k-cliques that share $k-1$ nodes
- Module: union of adjacent k-cliques



1. Find all k-cliques
2. Merge iteratively all adjacent k-cliques

Clique Percolation Method [Palla et al., 2005]



Generates overlapping clusters!

Clique Percolation Method [Palla et al., 2005]

Pros:

- Believed to be non-polynomial
- But claimed to be efficient on real networks
- Widely used for detecting overlapping communities

Cons:

- Failed for graph with few cliques
- Give a trivial community structure if too many cliques
- Left out nodes
- What value of k to choose to give a meaningful structure?

Clique Percolation Method [Palla et al., 2005]

Pros:

- Believed to be non-polynomial
- But claimed to be efficient on real networks
- Widely used for detecting overlapping communities

Cons:

- Failed for graph with few cliques
- Give a trivial community structure if too many cliques
- Left out nodes
- What value of k to choose to give a meaningful structure?