Dynamic and Social Network Analysis

Lecture 6

Miray Kas
Bilkent University
Computer Engineering Department

Network Properties: Structure, Groups, and Community

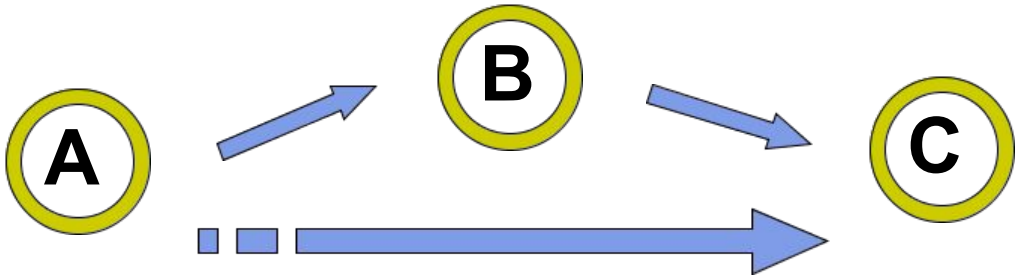
Partially covers "The structure and function of complex networks" by Mark Newman (https://arxiv.org/abs/cond-mat/0303516)

Network Properties

- Transitivity (Clustering)
- Degree Distributions
  - Scale-free networks
  - Maximum Degree
- Network Resilience
- Network Balance
- Mixing Patterns
  - Degree Correlations
- Community Structure (Forming Clusters)
- Network Navigation

Transitivity (1)

- The friend of your friend is likely also to be your friend



Transitivity (2)

- **Clustering Coefficients:**
  - (3 x Num of Triangles) / Num of connected triples
    - Fraction of triples that have their 3rd edge filled in to complete the triangle

$$C_i = \frac{\text{number of triangles connected to vertex } i}{\text{number of triples centered on vertex } i}$$

$$C = \frac{1}{n}\sum_i C_i.$$

    - Weights low-degree vertices heavily, because they have a small denominator

Degree Distributions

Degree = Number of edges connected

- p_k = Fraction of vertices with degree k
  = Probability that a uniform randomly chosen vertex has degree k

- Degree Distribution
  - Usually, right skewed, long tail
  - Power laws tend to dominate

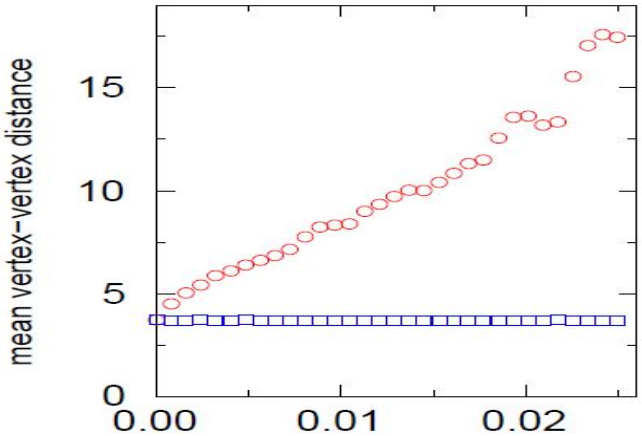
- Tricky for bipartite graphs:
  - $p_k \rightarrow p_{jk}$
  - *j* for in, *k* for out degree

Scale Free Networks & Maximum Degree

- Networks with power law degree distributions
- Only power law degree distributions are scale free
- Other potential distributions:
  - Exponential
  - Power law with exponential cutoff (Power law with an exponential cutoff behaves like a power law for small values of x, but drops more steeply (exponentially) for larger x)
- Maximum Degree:
  - Important for preferential attachment studies

Network Resilience

- Related to degree distributions
- How resilient to removal of its vertices
  - Typically: Paths will get longer
  - Ultimately: Vertices will become disconnected
  - Typical application: Vaccination in epidemiology



RED: Remove in the order of degrees
BLUE: Remove randomly

Internet is:

- Vulnerable to degree based attacks
- Resilient to random failure (Most vertices have low degrees, rarely on others' paths)

Structural Balance

Structural Balance

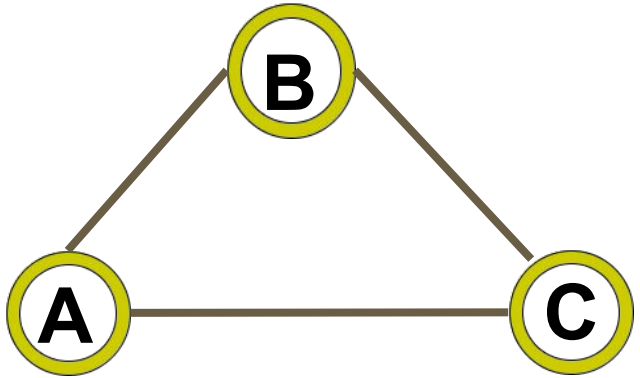
- Comes from social psychology theory
- Only considers signed graph or signed digraph
- ***Signed graph***: Each edge has a positive or a negative sign
  - The combination of connections and signs will define the balance

- **Generalization:** structural balance < clusterability < transitivity
- Examine two connected node in isolation
  - Label the edge **+** if they are friends
  - Label the edge **–** if they are enemies

- Take product of the signs to determine balance:
  - (**+**) (**+**) = (**+**)
  - (**-**) (**+**) = (**-**)
  - (**-**) (**-**) = (**+**)

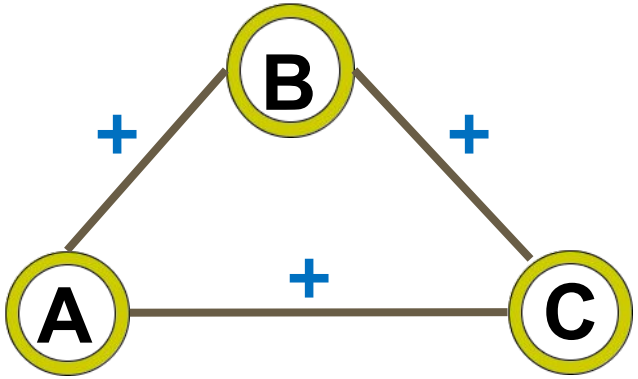
Balance between 3 Nodes

- Take 3 connected nodes
- ***4 possible situations:***
  - 3 pluses
  - 2 pluses, 1 minus
  - 1 plus, 2 minuses
  - 3 minuses

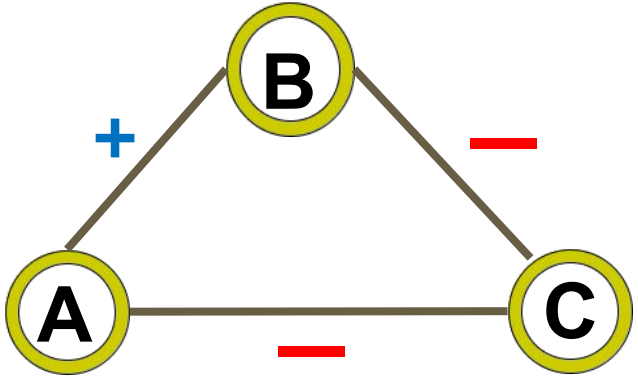


- ***Sign of a cycle:*** product of the signs in a cycle
- ***Balance of a cycle:*** A cycle is balanced if its sign is positive
- ***Balance of a graph:*** A graph is balanced if all cycles are positive

Balance between 3 Nodes

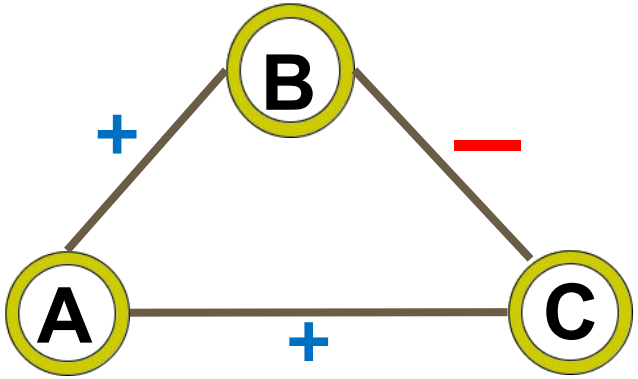


- Balanced
- A, B, and C are mutual friends

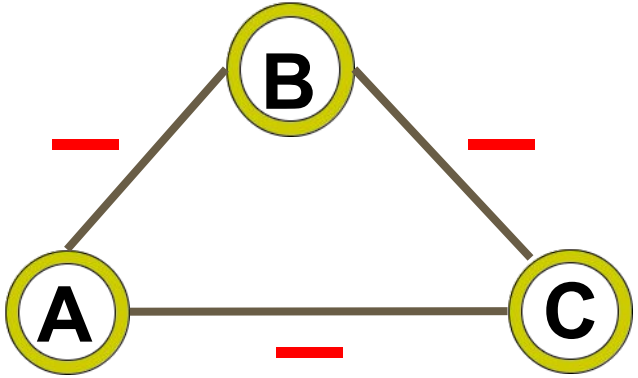


- Balanced
- A and B are friends.
- A and B have a common enemy: C
- A and B can be motivated to team up against C

Balance between 3 Nodes

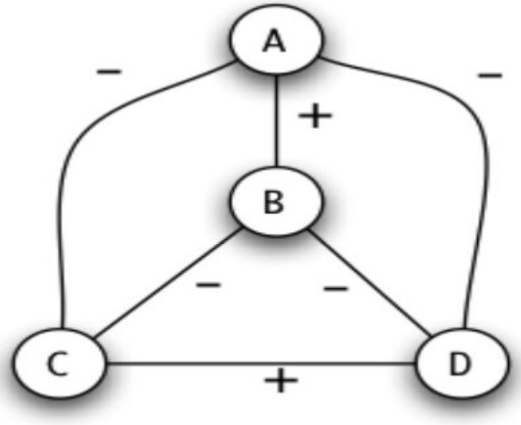


- Not balanced
- A is friends with both B and C, but B and C are enemies.
- A will have stress often because B and C cannot get on well

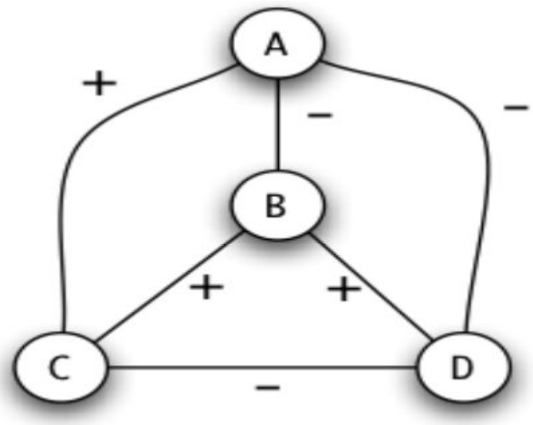


- Not balanced
- Everybody is enemies with one another
- Hard to motivate, no one trust or teams up with another

Example: Structural Balance in a 4-node Graph



Balanced



Not balanced

All cycles are balanced:

- A-B-C = (+) * (-) * (-) = +
- A-B-D = (+) * (-) * (-) = +
- B-C-D = (-) * (-) * (+) = +
- A-C-D = (-) * (-) * (+) = +

NOT all cycles are balanced:

- **A-B-C = (-) * (+) * (+) = -**
- **B-C-D = (+) * (+) * (-) = -**
- A-B-D = (-) * (-) * (+) = +
- A-C-D = (+) * (-) * (-) = +

https://gramoli.redbellyblockchain.io/largescalenetworks/lectures/04a-structural-balance.pdf

Structural Balance in Real Life

- Lack of structural balance creates stress for people
- People want to minimize such situations in their lives
    - Unbalanced settings are less common in real life settings

Balance Theorem

If a graph is balanced, one of the two cases is observed:

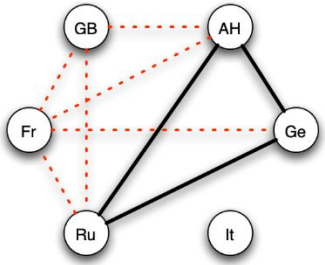
1. Everybody likes one another
2. Graph can be divided into 2 groups X and Y:
    a. Everyone in X likes one another
    b. Everyone in Y likes one another
    c. Everyone in X is enemies with everyone in Y
    d. Everyone in Y is enemies with everyone in X

- We are skipping the mathematical proof
- Use Case Example: International Relations

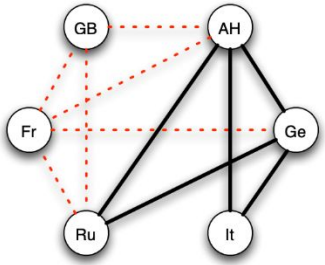
Structural Balance Use Case: International Relations

- Nodes are nations
  - Edges labeled **+** if two nations are in alliance
  - Edges labeled **-** if two nations are in animosity
- When a war is brewing, the nodes slide into alliances over time.
- **Example:** formation of alliances in World War 1.
  - Reference: Antal, Krapivsky, and Redner. Social balance on networks: the dynamics of friendship and enmity. Physica D, 224(130), 2006. https://arxiv.org/pdf/physics/0605183.pdf

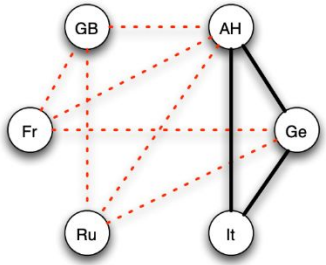
Formation of Alliances in World War I



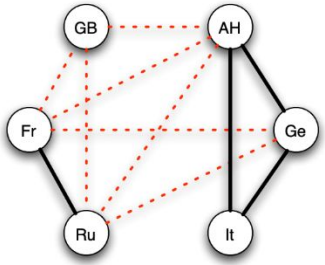
(a) *Three Emperor's League 1872–81*



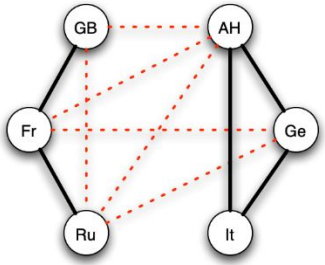
(b) *Triple Alliance 1882*



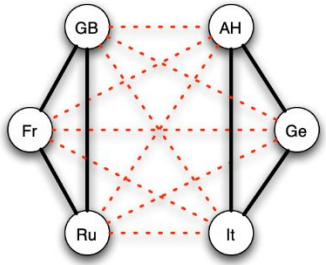
(c) *German-Russian League 1890*



(d) *French-Russian Alliance 1891–94*



(e) *Entente Cordiale 1904*



(f) *British Russian Alliance 1907*

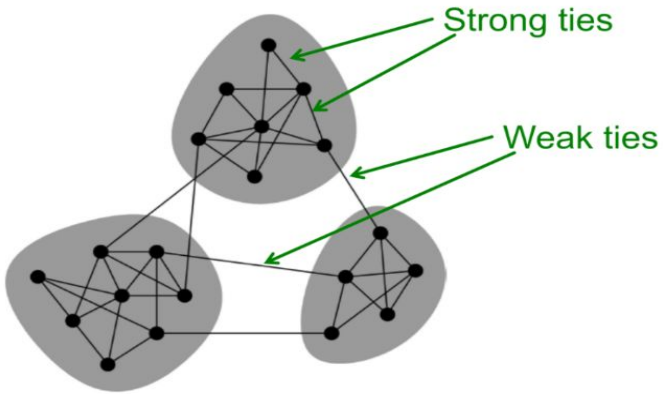
- Solid dark lines indicate alliance
- Red dotted lines indicate enmity
- Note the formation of two alliance groups over time.
  - Gradual reorganization of relations to a socially balanced state.
  - Social balance is a natural outcome!

https://courses.cit.cornell.edu/info204_2007sp/balance.pdf

Detecting Communities

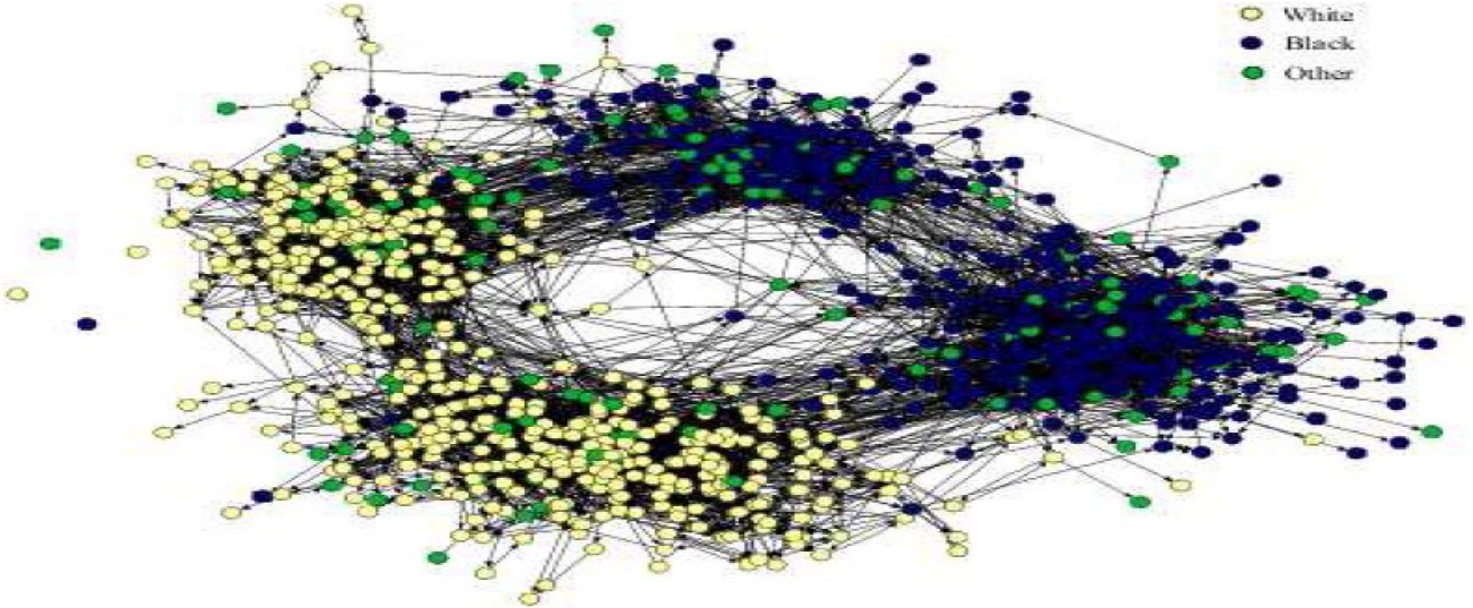
Community Structure in Networks

- Nodes are often clustered into tight knit groups
  - Communities, groups, clusters, modules
- High density ties within a community (*Intracluster density*)
- Low density ties between between communities (*Intercluster density*)
  - Strong ties within the community, weak ties bridging across communities



Community Structure (1): Density

- **High** density of edges ***within*** groups
- **Low** density of edges ***between*** groups
  - Age, gender, interest, job, race



Community Structure (2): Cluster and Block Analysis

- **Cluster Analysis:**
  - Extracting community structure from a network
- **Block Models**
  - Using *Structural Equivalence*
  - Divide into blocks/communities according to a criterion

Mixing Patterns

- There are types of vertices
  - 'Mixing matrix' btw types of vertices

- Which vertices pair up with which?

- **Social networks:**
  - Mixing by race
  - Selective linking called assortative mixing
    People tend to find people from their own background

- **Degree Correlations:**
  - Special case of assortative mixing
  - Mixing according to vertex degree

		women			
		black	hispanic	white	other
men	black	506	32	69	26
	hispanic	23	308	114	38
	white	26	46	599	68
	other	10	14	47	32

let $\mathbf{E}$ be the matrix with elements E_{ij}

$$P(j|i) = e_{ij} / \sum_j e_{ij} \qquad \sum_j P(j|i) = 1$$

$$Q = \frac{\sum_i P(i|i) - 1}{N - 1}$$

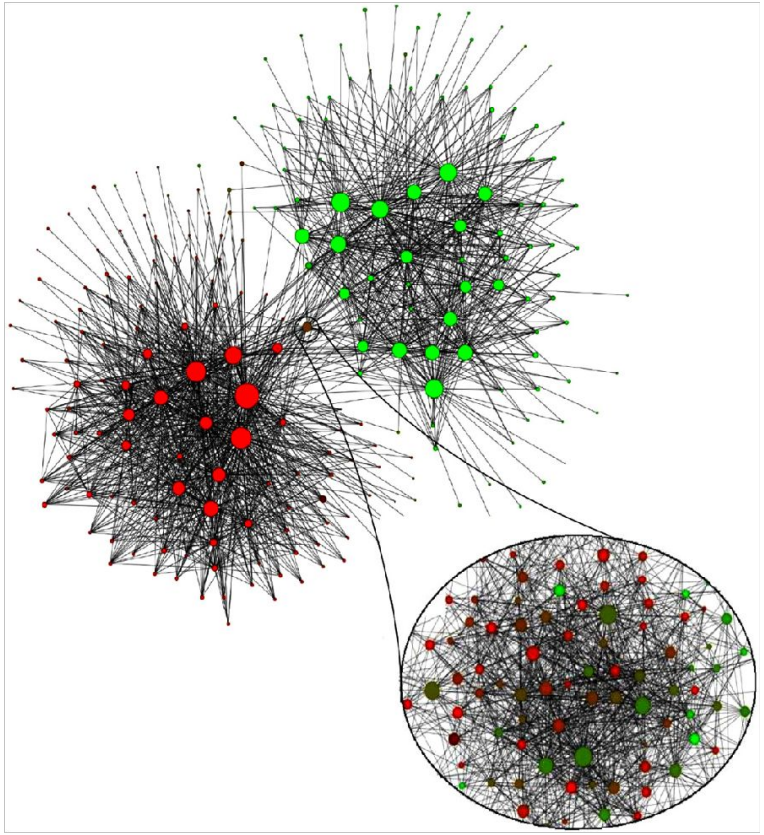
Assortive mixing coefficient

Belgium Communities

- Two ethnic cultures coexisting for centuries
- The country is bilingual:
  - 59% Flemish - Speak Dutch
  - 40% Walloons - Speak French

V.D. Blondel et al, *J. Stat. Mech*. P10008 (2008).
A.-L. Barabási, *Network Science: Communities*.
http://networksciencebook.com/

Communities via Belgium Mobile Phone Operators



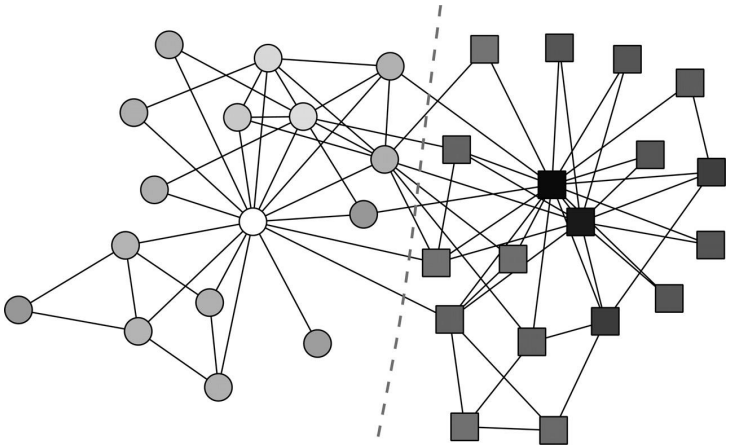
- Identify groups that talk to each other often
  - Family, neighbors, friends, colleagues
- Belgium is broken into 2 huge communities that never talk to each other
  - 3rd group in the middle is much smaller; mediating between the two parts of Belgium

V.D. Blondel et al, *J. Stat. Mech*. P10008 (2008).

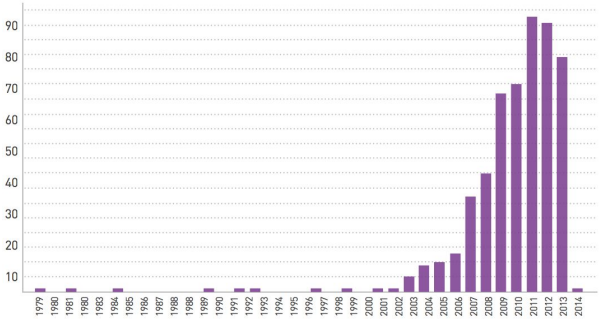
A.-L. Barabási, *Network Science: Communities*.

http://networksciencebook.com/

Zachary's Karate Club



- 34 karate club members that split into two factions after a disagreement between the coach and the president.
- Studied by sociologist Wayne Zachary in 1977
- Benchmark dataset used for community detection algorithms. Became famous after Mark Newman used it in his paper in 2002!

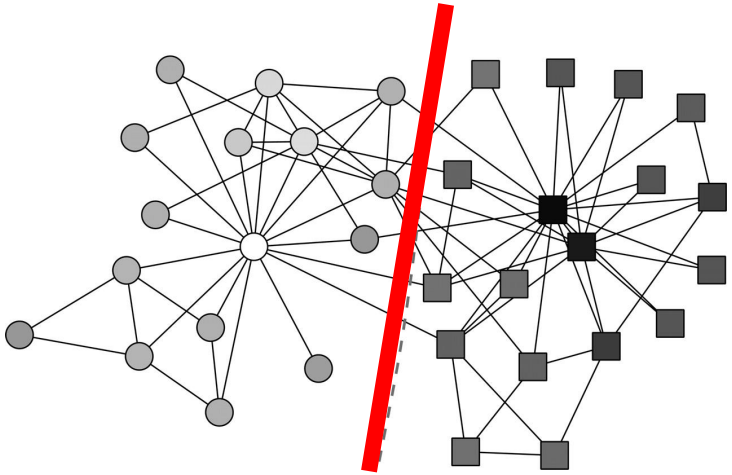


Citation history!

W.W. Zachary, *J. Anthropol. Res.* 33:452-473 (1977).

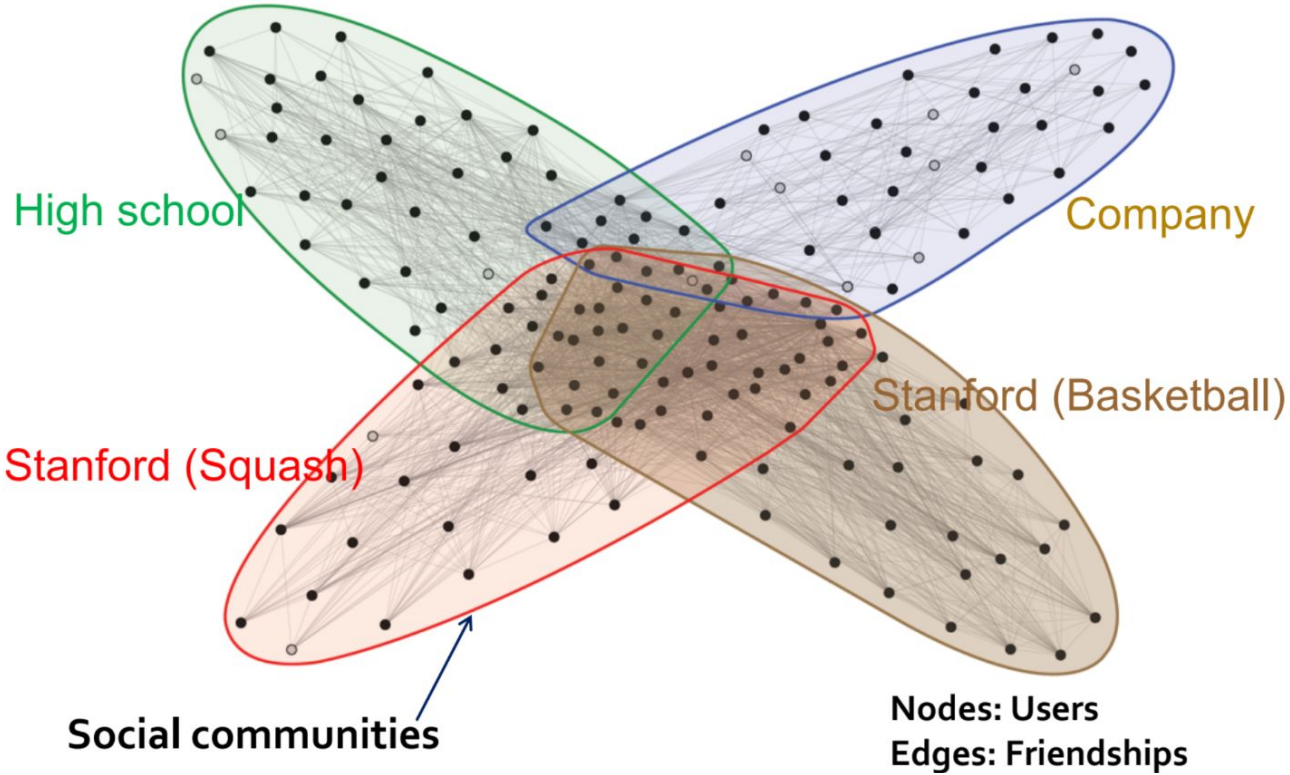
A.-L. Barabási, *Network Science: Communities*.
http://networksciencebook.com/

Zachary's Karate Club

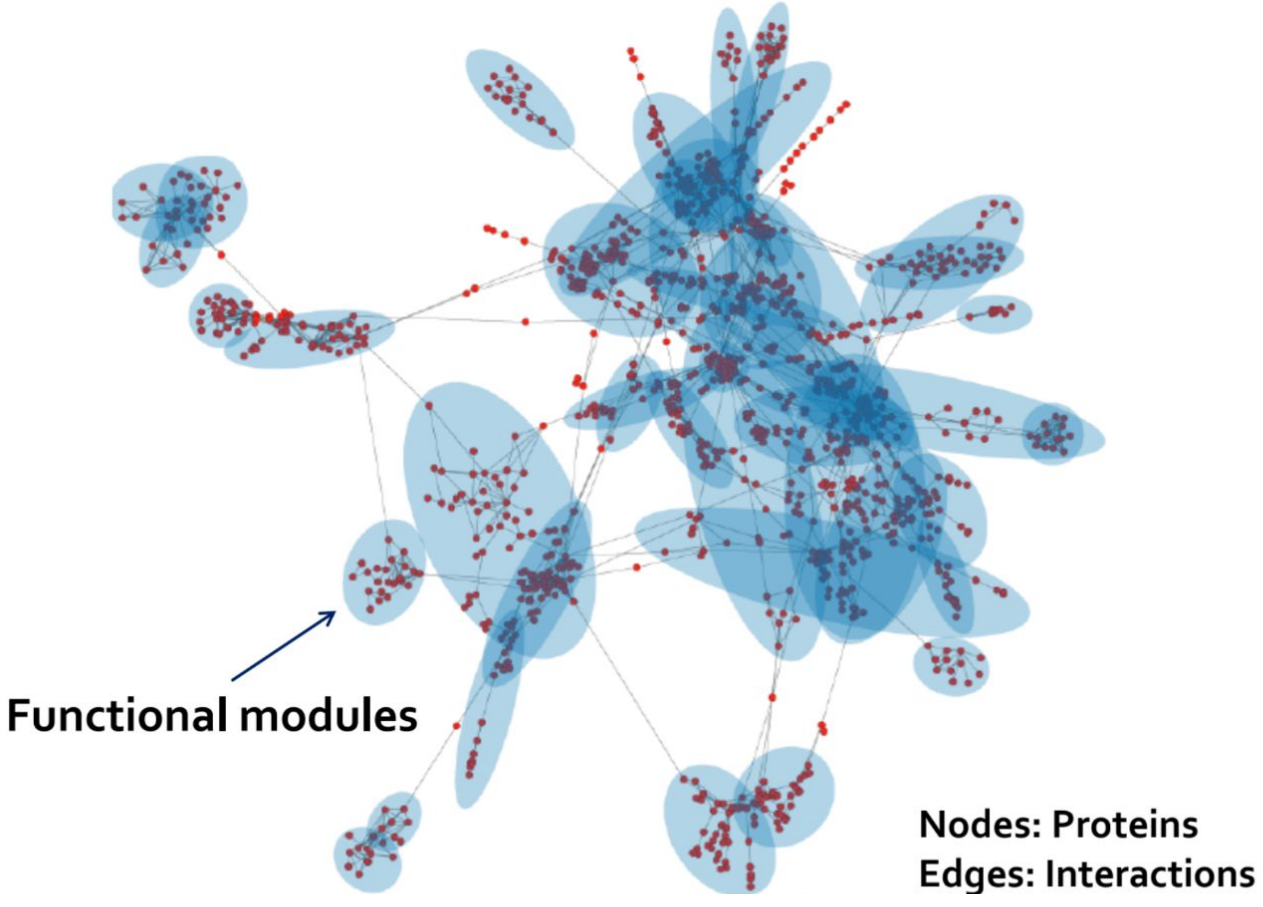


The split due to the conflict in the network can be explained by the minimum cut.

Facebook Ego Network



Protein - Protein Interactions



Community Detection in Different Networks

- Makes sense on sparse networks - *Most real life networks are sparse!*
- Form of relationships affect the community detection
- **Direction and weights of edges**
  - If the graph is directed, community detection is harder.
  - Reciprocity is usually low (e.g. In Web, it is less than 10%)
- **Overlapping communities**
  - In some graphs, a node may belong to more than one community
    - Alma Mater of people in an org. People with multiple degrees may belong to more than one community
- **Multimodal networks**
  - There might be different types of nodes.

Community Detection Methods

Taxonomy:

- **Divisive** (e.g. Girvan-Newman algorithm, Min-cut Max Flow)
- **Quality Optimization** (e.g. Modularity optimization)
- **Subgraph Discovery** (e.g Cliques, Quasi-Cliques, k-cores)
- **Vertex Clustering** (e.g. Spectral Clustering)
- **Model based** (e.g. Label propagation)

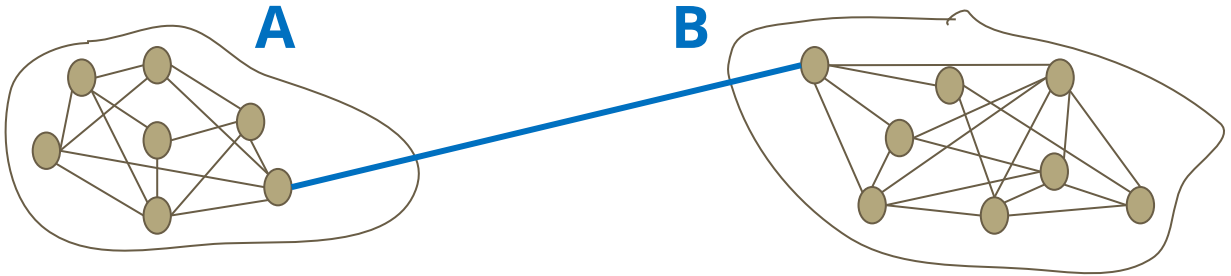
Potential Applications

- Classification
  - Identify customers with similar interests
  - Identify customers that are locally closeby to one another
- Graph compression (Size reduction and coarser representation)

Example for Divisive Community Detection Algorithms

Edge Betweenness & Strength of Weak Ties

- **Betweenness Centrality:** Defined on the node. The number of the shortest path passing through the node.
- **Edge-Betweenness:** Defined on the edge. The number of the shortest path passing through the edge.
- Assume 2 large clusters A and B, and a single link between A and B.
  - The bridging link will have be a part of every shortest path between A and B.



Girvan-Newman Algorithm

- "*Community structure in social and biological networks*" by M. Girvan and M. Newman (2002) https://www.pnas.org/content/99/12/7821

- Hierarchical Community Detection Algorithm
  - Identifies communities by progressively removing edges
  - Gives a hierarchical decomposition of communities

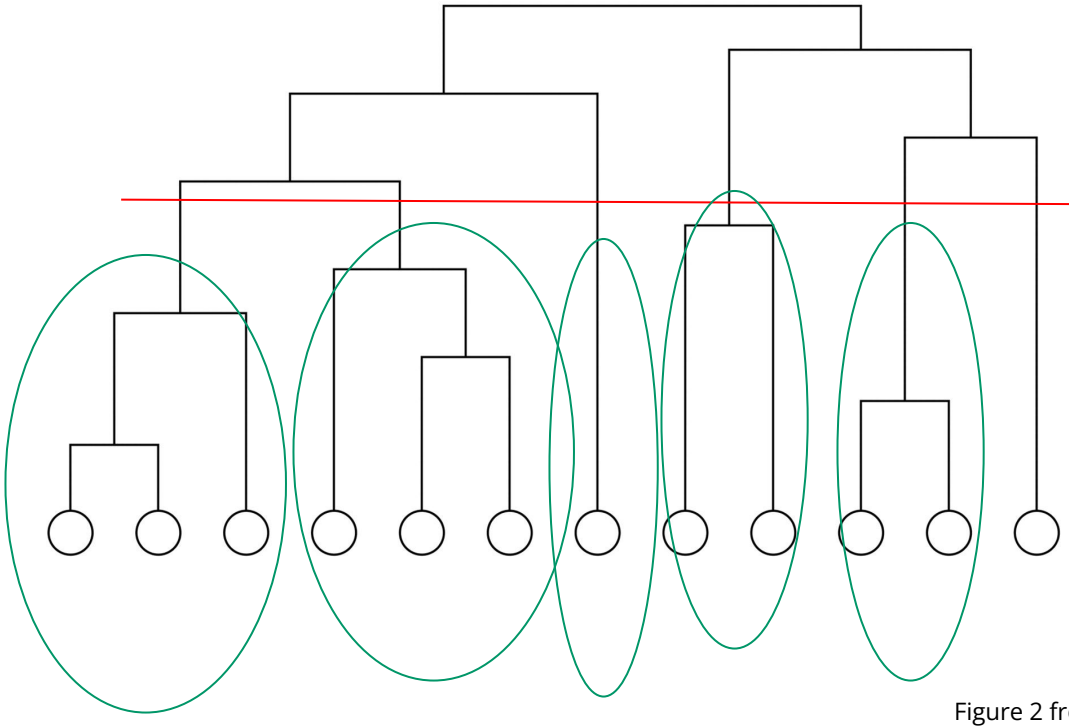
- **Which edge to remove?**
  - **Utilize Strength of Weak Ties!**
  - Keep removing the edge with the highest **edge-betweenness** until there are no more edges.

Girvan-Newman Algorithm

- Calculate edge betweenness for all edges
- Repeat until no edges remain
  - Remove the edge with the highest betweenness
  - Recalculate betweenness of all edges affected by the removal
- Cross cut the dendrogram of components

What is Dendrogram?

- Hierarchical cluster tree

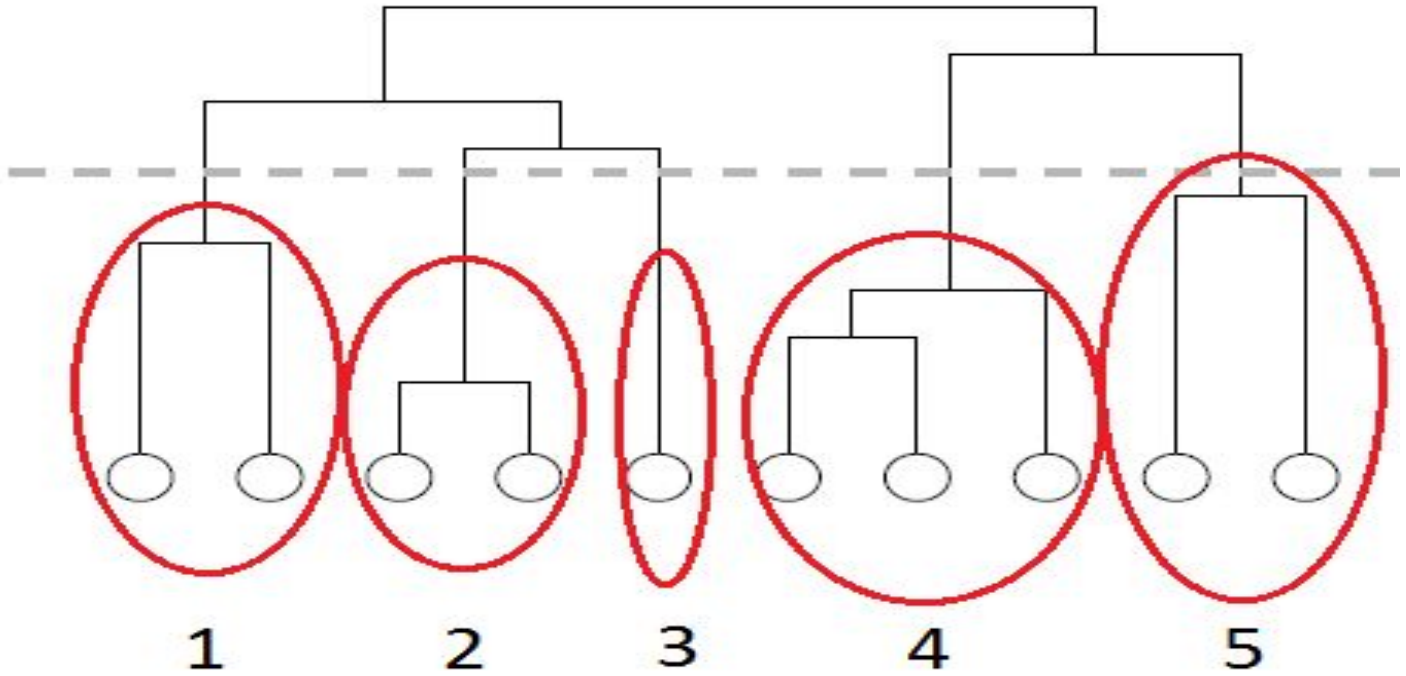


- The circles at the bottom represent the nodes in the network
- The tree shows the order the nodes join together to form communities for a given definition of the weight W_{ij} of connections between node pairs.

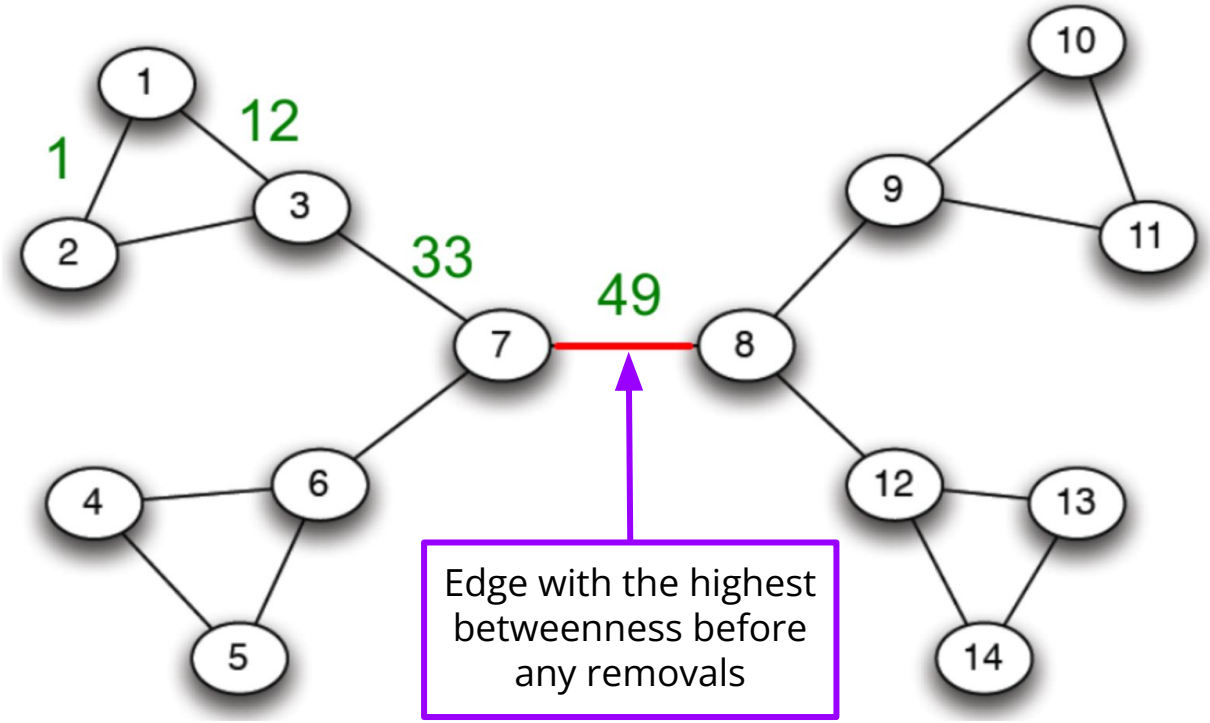
Figure 2 from https://www.pnas.org/content/99/12/7821/tab-figures-data

Community Structure via a Dendrogram

- Horizontal cut at any level = Number of clusters

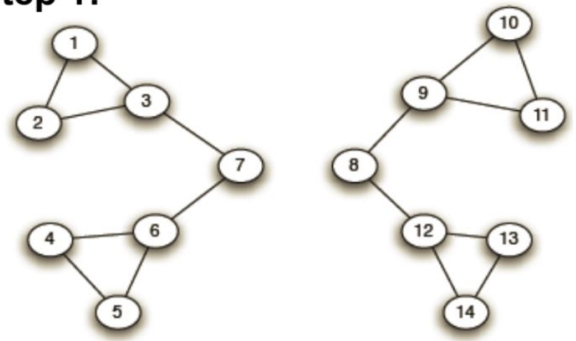


Girvan-Newman Algorithm Step by Step Example

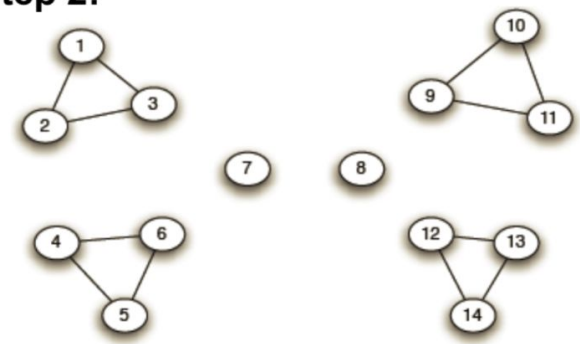


Girvan-Newman Algorithm Step by Step Example

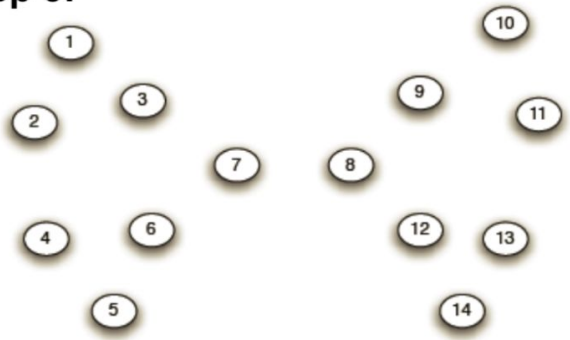
Step 1:



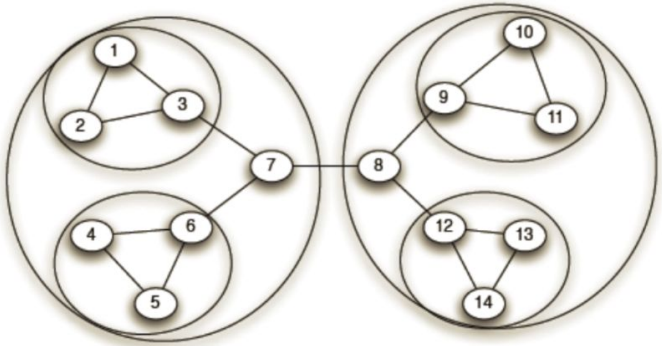
Step 2:



Step 3:



Hierarchical network decomposition:



Example for Quality Optimization based Community Detection Algorithms

Modularity *Q*

- **Communities:** Sets of tightly connected nodes.
  - A graph can be split into communities in multiple ways
- **Modularity:** Overall quality of partitioning of a graph
  - Higher if a particular partitioning of graph has more intra-community links than the same community structure being rewired under Erdos-Renyi graph model.

- Given a particular partitioning of a graph into communities $c \in C$

$$Q \propto \sum_{c \in C} [\#(links\ within\ c) - \#(expected\ links\ within\ c)]$$

This can return a negative value

⇒ *An **Erdos-Renyi** (ER) **graph** on the node set N is a **random graph** which connects each pair of nodes {i,j} with probability p, independent.*

Modularity *Q*

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

Observed number of intra community edges

Expected number of edges between *i* and *j* if they were created randomly

- A_{ij} = Adjacency Matrix
- k_i = Degree of node *i*
- m = number of edges on the graph
- c_i = Community of node i
- $D(c_i, c_j)$ = 1 if *i* and *j* are in the same community

Modularity *Q*

- In random graph (ER model), we expect **Q = 0**
- Typical values are between **0.3 and 0.7** in most real networks
  - **Higher side of 0.3 - 0.7 range:** Clear, separate communities
  - **Lower side of 0.3 - 0.7 range:** Fuzzy, overlapping communities

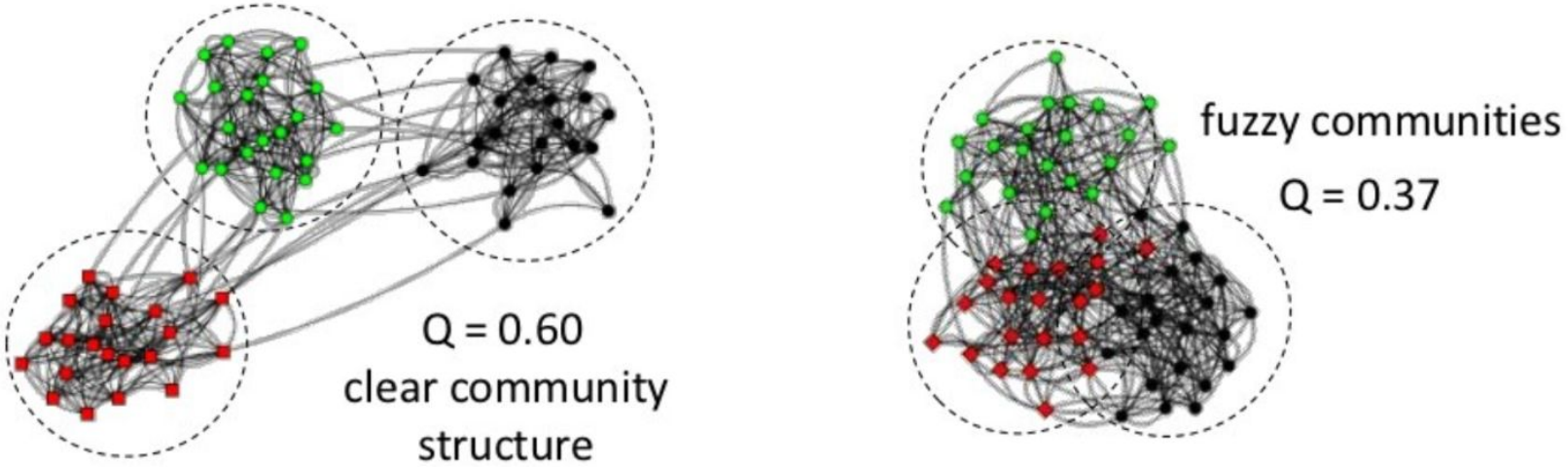


Image borrowed from Community Detection in Social Media by S. Papadopoulos

https://www.slideshare.net/sympapadopoulos/community-detection-in-social-media?next_slideshow=1

Modularity Optimization

- **Goal:** Seek a community structure that ***maximizes modularity***
- Search space is exponential with the number of nodes
  - Use heuristics or randomized algorithms for faster outcome

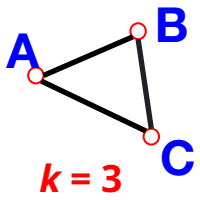
- **Greedy Algorithm (Newman 2004):**
  - **Initialization:** Each node belongs to a different community;
  - **Until one community remains:**
    - Put the nodes/communities that increase modularity the most when joined in the same community. (*Modularity is calculated for the full network*)
  - Choose the network partition with the highest modularity.

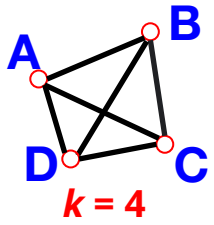
M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E,* vol. 69, no. 6, 2004

Example for Subgraph Discovery based Community Detection Algorithms

SubGraph Discovery

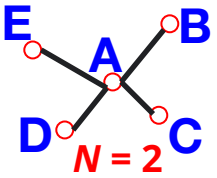
- ***k*-clique**





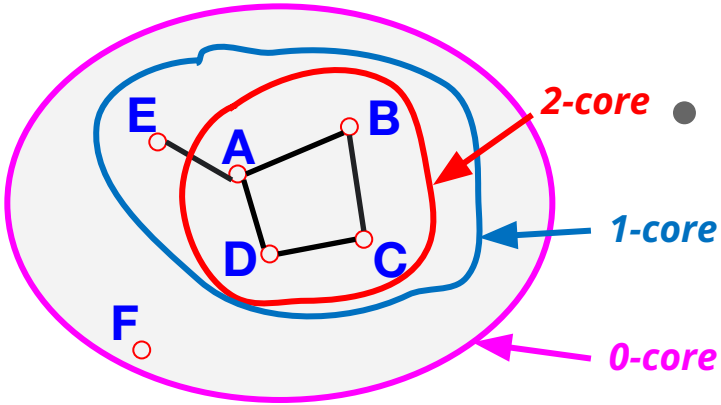
- All nodes are connected to one another

- ***N*-Clique**



- There is a path of less than *N* between all nodes

- ***k*-core**



- A node is connected to at least *k* other nodes

k-core Decomposition

- ***k-core:*** Maximal subgraph where all nodes have a degree of at least k

- ***k*-core decomposition** of a network is the maximal set of nodes that have at least *k* neighbours within the set.

- A ***k*-shell** is defined as the set of nodes belonging to the k^{th} core but not to the $(k+1)^{th}$ core.

- **Algorithm:** Recursively remove the nodes with less than *k* connections.

Malvestio, I., Cardillo, A. & Masuda, N. Interplay between k-core and community structure in complex networks. *Sci Rep* 10, 14702 (2020). https://doi.org/10.1038/s41598-020-71426-8

k-core Structures of Real-World Graphs

- **MIRROR PATTERN:** coreness of vertices (i.e., maximum k such that each vertex belongs to the k-core) is strongly correlated to their degree.
- **CORE-TRIANGLE PATTERN:** degeneracy of a graph (i.e., maximum k such that the k-core exists in the graph) obeys a 3-to-1 power law with respect to the count of triangles.
- **STRUCTURED CORE PATTERN:** degeneracy-cores are not cliques but have non-trivial structures such as core-periphery and communities.

K. Shin, T. Eliassi-Rad and C. Faloutsos, "CoreScope: Graph Mining Using k-Core Analysis — Patterns, Anomalies and Algorithms," *IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 469-478, doi: 10.1109/ICDM.2016.0058.

Examples for *k*-core Decomposition Usage

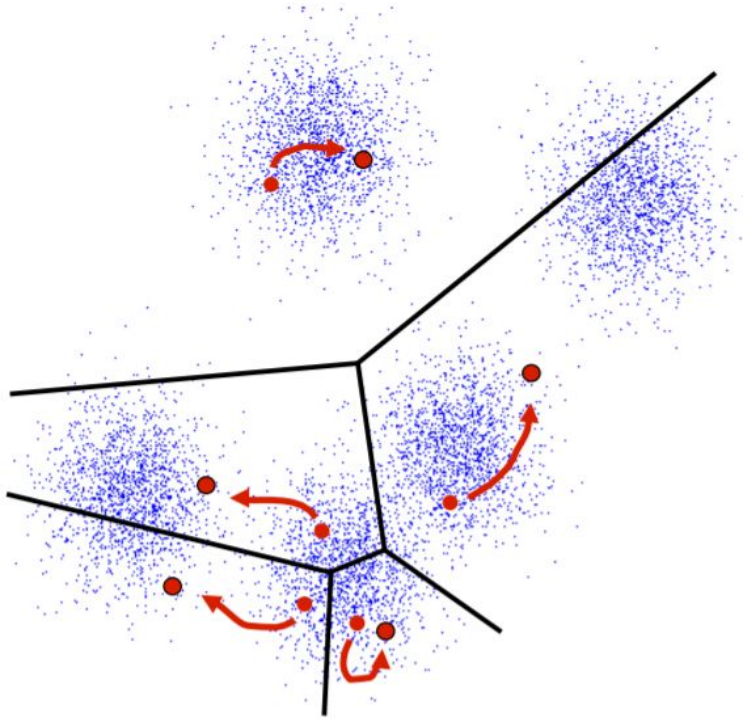
- Used in prior research for
  - Identifying and ranking the most influential spreaders
  - Identifying keywords used for classifying documents
  - Assessing the robustness of mutualistic ecosystem and protein networks.

Malvestio, I., Cardillo, A. & Masuda, N. Interplay between k-core and community structure in complex networks. *Sci Rep* 10, 14702 (2020). https://doi.org/10.1038/s41598-020-71426-8

Example for Vertex Clustering based Community Detection Algorithms

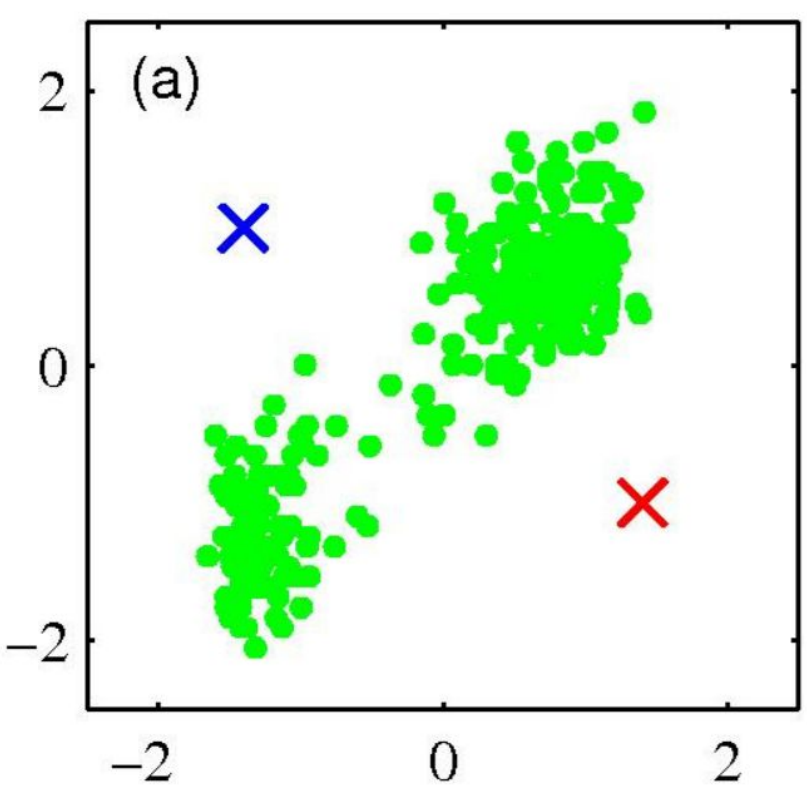
K-means Clustering

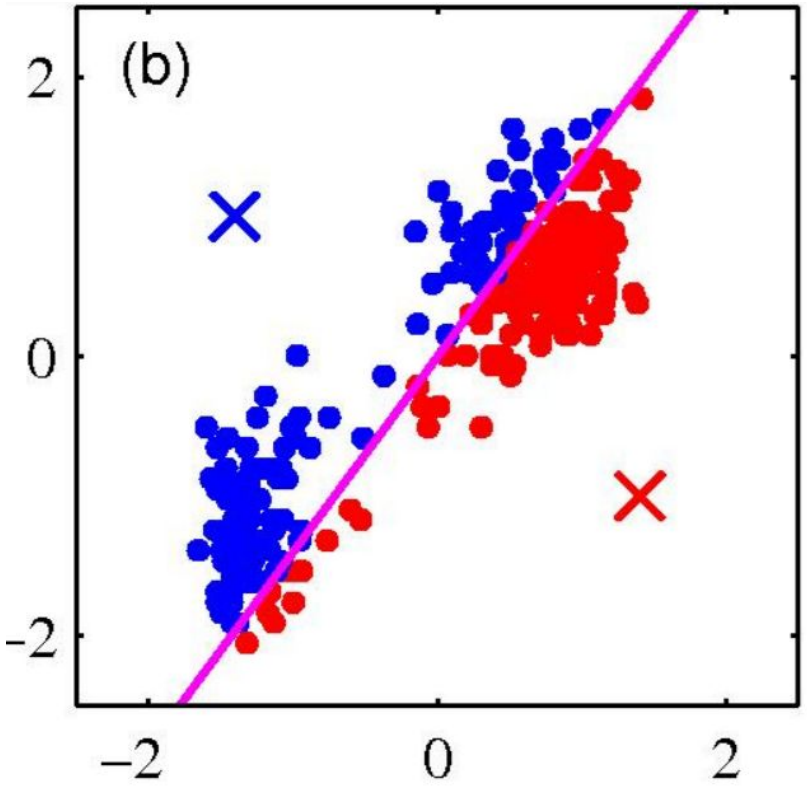
An iterative clustering algorithm
– Initialize: Pick K random points as cluster centers
– Alternate:
1. Assign data points to closest cluster center
2. Change the cluster center to the average of its assigned points
– Stop when no points' assignments change



Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

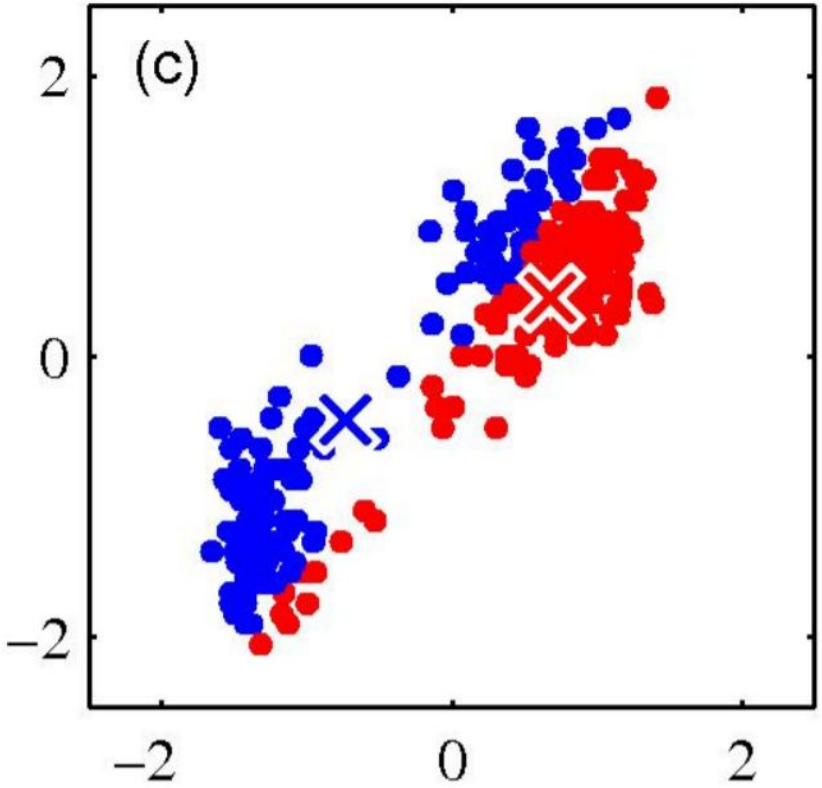
K-means Clustering Example

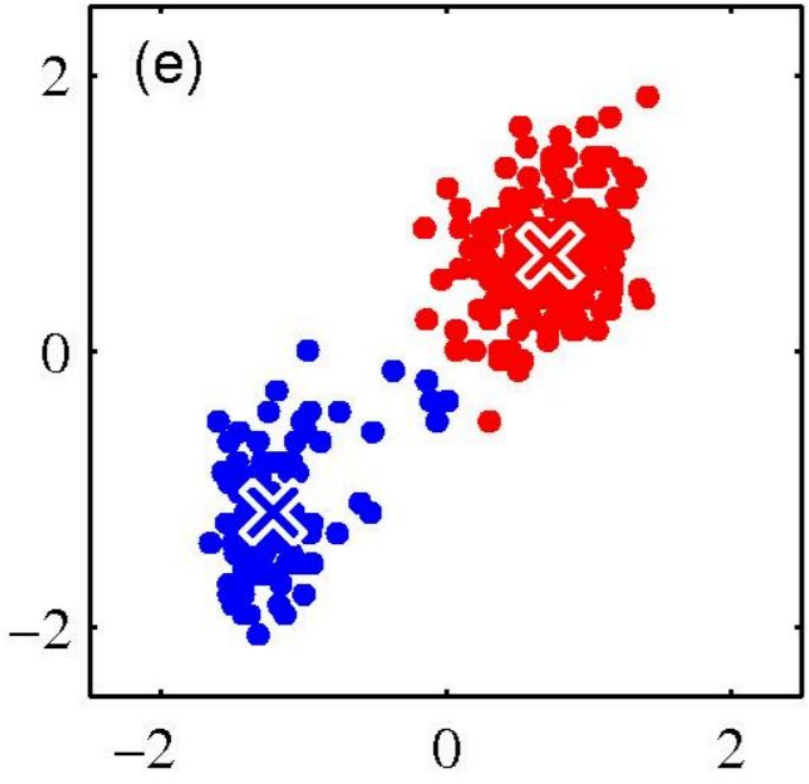




Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

K-means Clustering Example





Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

Spectral Clustering

- Transforms the **clustering** problem into a **graph-partitioning** problem.
- Define some notion of **similarity s_{ij}** between nodes *i* and *j*
  - Connect nodes *i* and *j* if **$s_{ij} > T$ (*T* is a threshold)** in the transformed graph
  - The edge between *i* and *j* is weighted s_{ij}
- **Partition** the graph such that similar nodes are in the same cluster
  - Very **low** weights **between** clusters
  - Very **high** weights **within** clusters
- Spectral Clustering uses information from the eigenvalues (spectrum) of special matrices (i.e. Affinity Matrix, Degree Matrix and Laplacian Matrix) derived from the graph or the data set.

Eigengap Heuristic for Spectral Clustering

- Compute the optimal number of clusters based on the largest distance between consecutive Eigenvalues of the input's Laplacian

Laplacian Matrix (L) = Degree Matrix (D) - Adjacency Matrix (A)

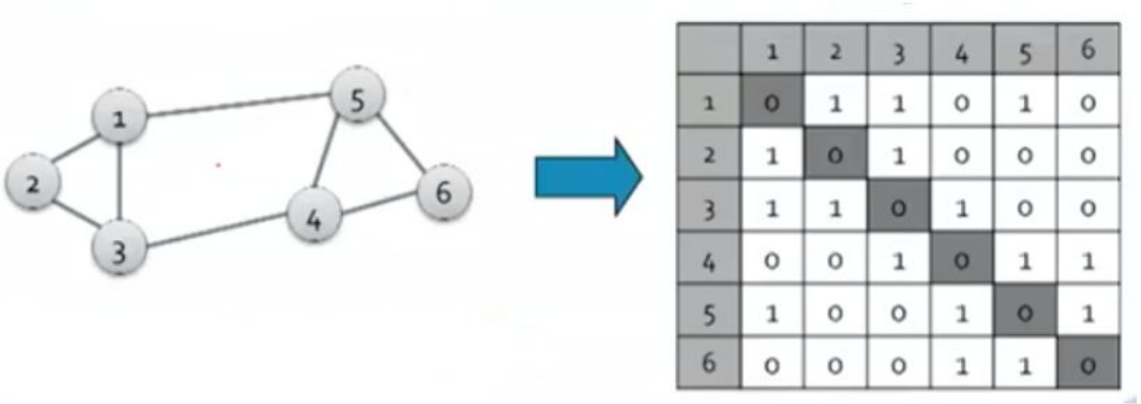
- Computationally expensive since Eigenvalues need to be computed
  - Good since it makes no assumptions about the structure/data
  - Popular unsupervised ML algorithm
  - Can be computed with simple algebraic operations
  - Quick blog if you want to read more on ML side

https://towardsdatascience.com/spectral-graph-clustering-and-optimal-number-of-clusters-estimation-32704189afbe

Eigengap Heuristic for Spectral Clustering

Adjacency Matrix (A)

A[i][j] = 1 if there is an edge between node i and node j, 0 otherwise



	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

Because A is symmetric, its eigen vectors are real and orthogonal (the dot product is 0).

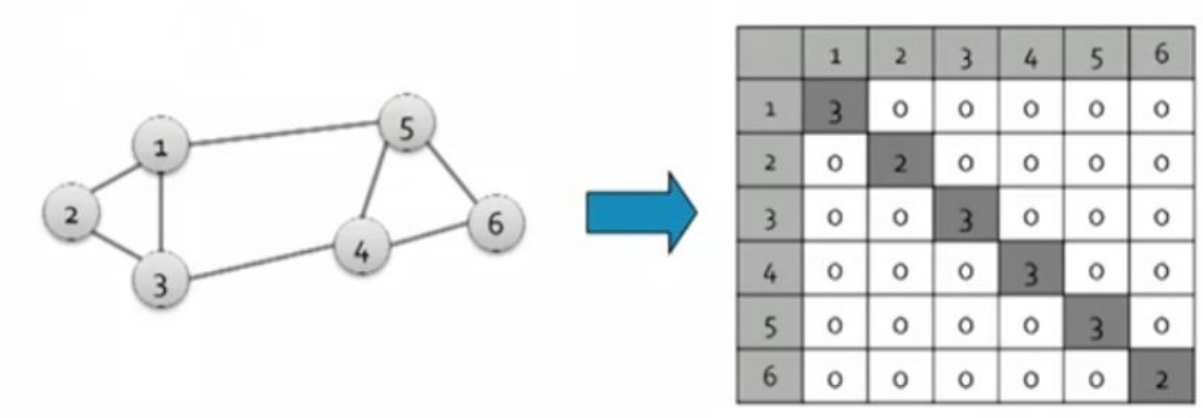
https://towardsdatascience.com/spectral-graph-clustering-and-optimal-number-of-clusters-estimation-32704189afbe

Eigengap Heuristic for Spectral Clustering

Degree Matrix (D)

d[i][i] = the number of adjacent edges in node i or the **degree of node i**

d[i][j] = 0



	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

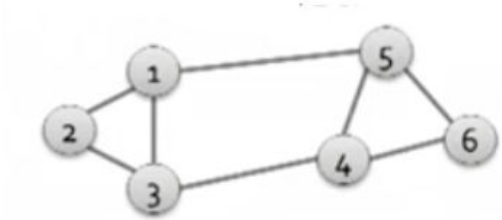
https://towardsdatascience.com/spectral-graph-clustering-and-optimal-number-of-clusters-estimation-32704189afbe

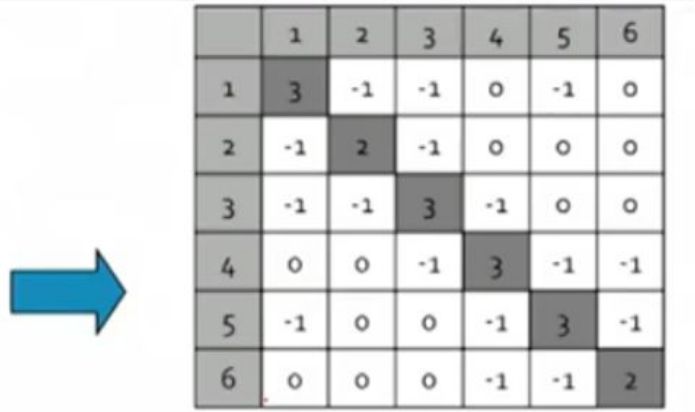
Eigengap Heuristic for Spectral Clustering

Laplacian Matrix (L)

The laplacian matrix is a n*n matrix defined as: **L = D -A**

Its **eigen values are positive real numbers** and the **eigen vectors are real and orthogonal** (the dot product of the 2 vectors is 0)





	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

https://towardsdatascience.com/spectral-graph-clustering-and-optimal-number-of-clusters-estimation-32704189afbe

Spectral Clustering

- Input: Similarity matrix $S \in n \times n$, number k of clusters to construct.
- Construct a similarity graph by one of the ways described. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- Compute the first k eigenvectors $u_1,\ldots,u_k$ of L.
- Let $U \in n \times k$ be the matrix containing the vectors $u_1,\ldots,u_k$ as columns.
- For $i = 1,\ldots,n$, let $y_i \in k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in k with the k-means algorithm into clusters $C_1,\ldots,C_k$.
- Output: Clusters $A_1,\ldots,A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

Example for Model based Community Detection Algorithms

Label Propagation Algorithm (LPA)

- Detects communities by propagating labels.
- Uses network structure as a guide
- **The Intuition Behind:**
  - A label quickly becomes dominant in a densely connected communities.
  - Labels get trapped in densely connected communities, and cannot easily propagate across communities.
  - Nodes that end up with the same label belong to the same community.
- Semi-supervised ML algorithm
  - It is possible to hand-label some nodes/communities to guide the solutions generated.

Label Propagation Algorithm (LPA)

- **Initialization:** Assign each node a unique community label.
- **Iteration** (Label Propagation through the network):
  - Each node updates its label to the one that the maximum numbers of its neighbors belongs to.
  - Ties are broken arbitrarily but deterministically.
- **Exit Condition:** Convergence or the user-defined maximum number of iterations.
  - **Convergence:** When each node has the majority label of its neighbors.

Example Use Cases for Label Propagation Algorithm (LPA)

- Twitter polarity classification with label propagation over lexical links and the follower graph
- Label Propagation Prediction of Drug-Drug Interactions Based on Clinical Side Effects
- "Feature Inference Based on Label Propagation on Wikidata Graph for DST"

Overlapping Community Detection

Overlapping Communities

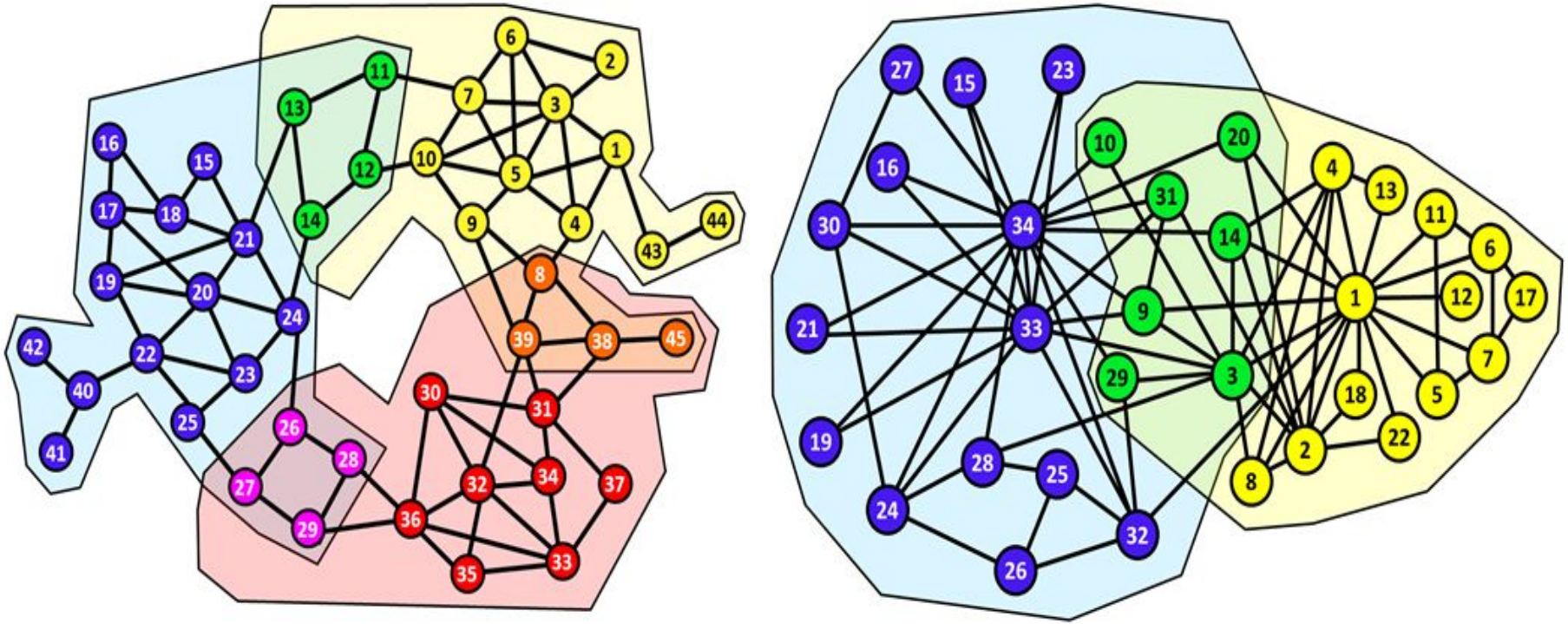


Image taken from https://bigdata.oden.utexas.edu/project/graph-clustering/

Clique Percolation Method (CPM)

- One of the most popular method to detect overlapping communities

- Builds up communities from ***k*-cliques**.
  - ***Reminder:*** *k-cliques are fully connected subgraphs of k nodes (3-clique is a triangle).*

- *k* is given as a parameter to the algorithm.

Clique Percolation Method (CPM) - Algorithm

Key idea

- Construct a clique graph. Mark each connected component of the clique graph as a community

Clique Graph

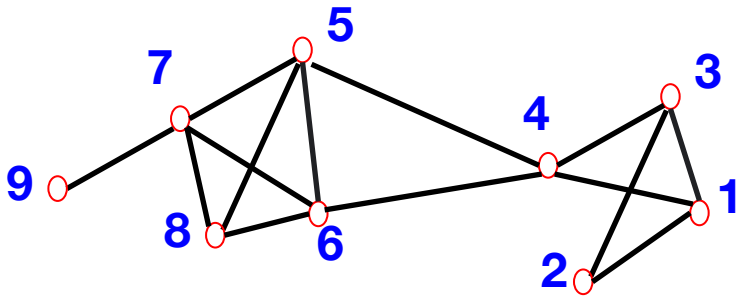
- **Nodes** ⇒ Cliques of size *k* (*k* is a parameter)
- **Edges** ⇒ There is an edge between two cliques if they share *k*-1 nodes.

Algorithm

For a given parameter *k*;

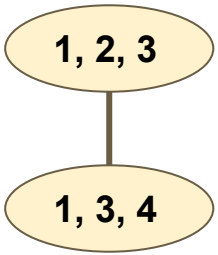
1. Find all cliques of size *k*
2. Construct clique graph. Add an edge between two cliques if they share *k*-1 nodes.
3. Mark each connected component of the clique graph as a community

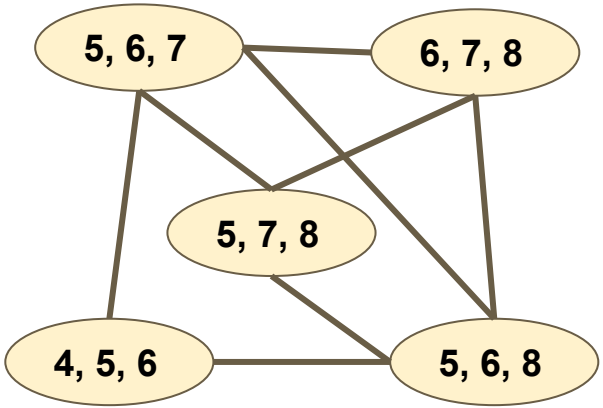
Clique Percolation Method (CPM)

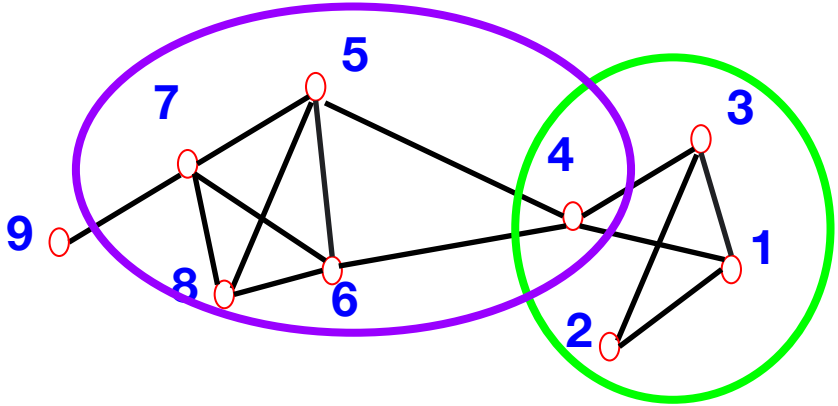


Cliques of size-3

{{1, 2, 3}, {1, 3, 4}, {4, 5, 6}, {5, 6, 7}, {5, 6, 8}, {6, 7, 8}, {5, 7, 8}}







Adapted from https://www.slideshare.net/NicolaBarbieri/community-detection-in-networks

Clustering Algorithms in ORA

Locate Groups: Measures & Algorithms

Analysis > Generate Reports > Characterize Groups and Networks > Locate Groups

Measures

- Betweenness Centrality
- Clique Count
- Clustering Coefficient
- Density
- Total Degree Centrality

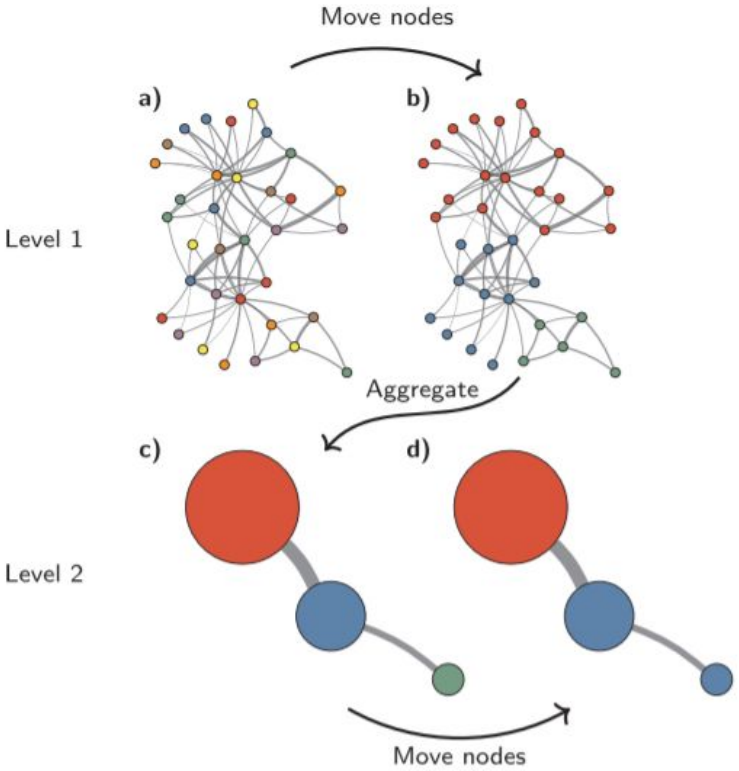
Algorithms

- Newman
- Girvan-Newman
- Louvain Method
- CONCOR
- Johnson
- K-Means
- Spectral Analysis
- K-Fog
- Alpha-Fog
- Leiden algorithm
- Latent Semantic Analysis
- Latent Dirichlet Allocation
- Local Patterns
- K-Cores

Locate Groups: Algorithms (1)

- **Louvain Method:** Detects groups by greedy optimization based on modularity scores.
    - Recursively merges communities into a single node and executes the modularity clustering on the condensed graphs. Handles large networks efficiently
- **Leiden Method:** Extends Louvain method. Faster. 3 phases: (1) local moving of nodes, (2) refinement of partition, (3) aggregation of the network

Louvain Algorithm



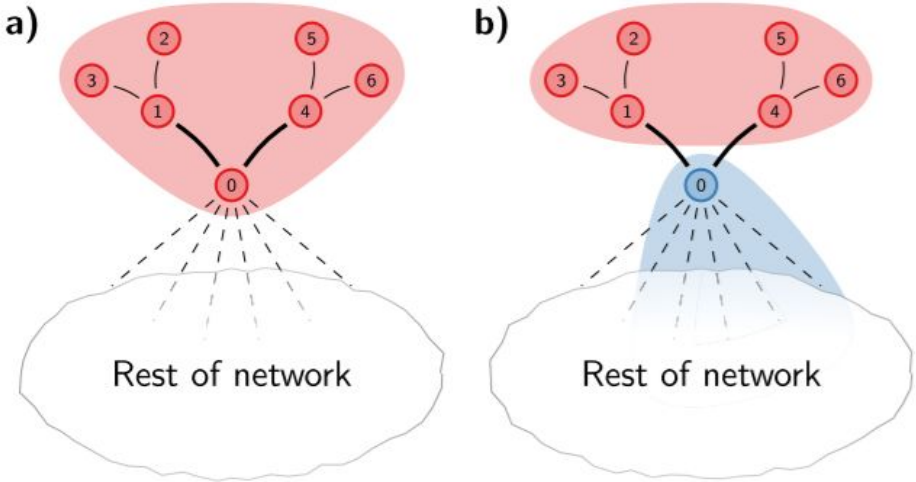


Figure 1. Louvain algorithm. The Louvain algorithm starts from a singleton partition in which each node is in its own community (**a**). The algorithm moves individual nodes from one community to another to find a partition (**b**). Based on this partition, an aggregate network is created (**c**). The algorithm then moves individual nodes in the aggregate network (**d**). These steps are repeated until the quality cannot be increased further.

https://www.nature.com/articles/s41598-019-41695-z.pdf

Leiden Algorithm

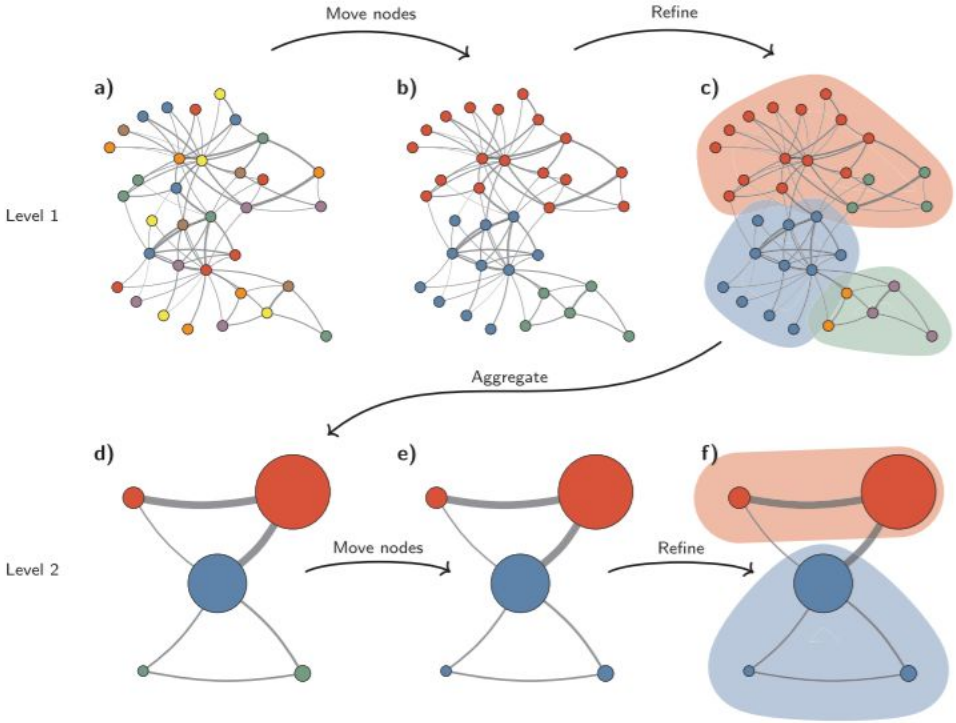


Figure 3. Leiden algorithm. The Leiden algorithm starts from a singleton partition (**a**). The algorithm moves individual nodes from one community to another to find a partition (**b**), which is then refined (**c**). An aggregate network (**d**) is created based on the refined partition, using the non-refined partition to create an initial partition for the aggregate network. For example, the red community in (**b**) is refined into two subcommunities in (**c**), which after aggregation become two separate nodes in (**d**), both belonging to the same community. The algorithm then moves individual nodes in the aggregate network (**e**). In this case, refinement does not change the partition (**f**). These steps are repeated until no further improvements can be made.

https://www.nature.com/articles/s41598-019-41695-z.pdf

Locate Groups: Algorithms (2)

- **Newman:** Starts with each node as its own group, and then combines groups in a hierarchical manner until only one group remains (e.g. it is agglomerative hierarchical clustering). At each step, which two groups to merge is based on what will maximize the modularity value - or group community structure value.
    - Finds unusually dense clusters, even in large networks.
- **Girvan-Newman:** A hierarchical clustering method where all nodes start in the same group and are split off into new groups by removing links with high betweenness scores.
- **CONCOR:** Starting with all nodes in a single group, CONCOR recursively splits (or partitions) each existing group into two groups based on the maximum correlation of outgoing connections (however groups with fewer than two nodes are not further partitioned).
    - CONCOR helps find groups with similar roles in networks, even if dispersed. Suitable for smaller networks with a few thousand nodes.
    - CONvergence of iterated CORrelations

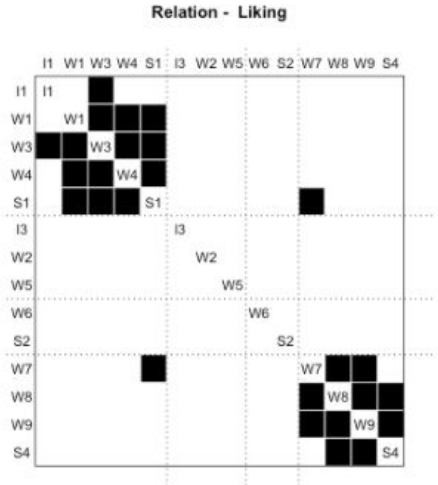
CONCOR

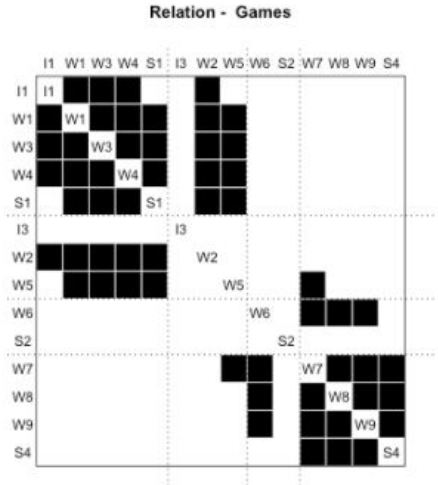
- CONCOR begins by correlating each pair of actors.
- Each row of the actor-by-actor correlation matrix is then extracted, and correlated with each other row.
    - Asking "how similar is the vector of similarities of actor X to the vector of similarities of actor Y?"
- This process is repeated over and over. Eventually the elements in this "iterated correlation matrix" converge on a value of either +1 or -1
- CONCOR then divides the data into two sets on the basis of these correlations.
- Then, within each set (if it has more than two actors) the process is repeated. The process continues until all actors are separated (or until we lose interest).
- The result is a binary branching tree that gives rise to a final partition.

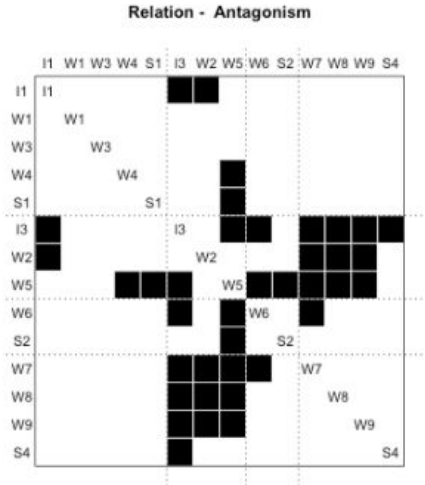
https://faculty.ucr.edu/~hanneman/nettext/C13_%20Structural_Equivalence.html#concor

CONCOR

- It is a block model idea based on structural equivalence
- Example: Consider a country map, and cities/towns are the nodes.
  - Two towns of Ankara are similar enough to one another they will have similar connections and distances to the towns of other cities. They won't be exactly the same, but they will be close enough when you consider the whole country.



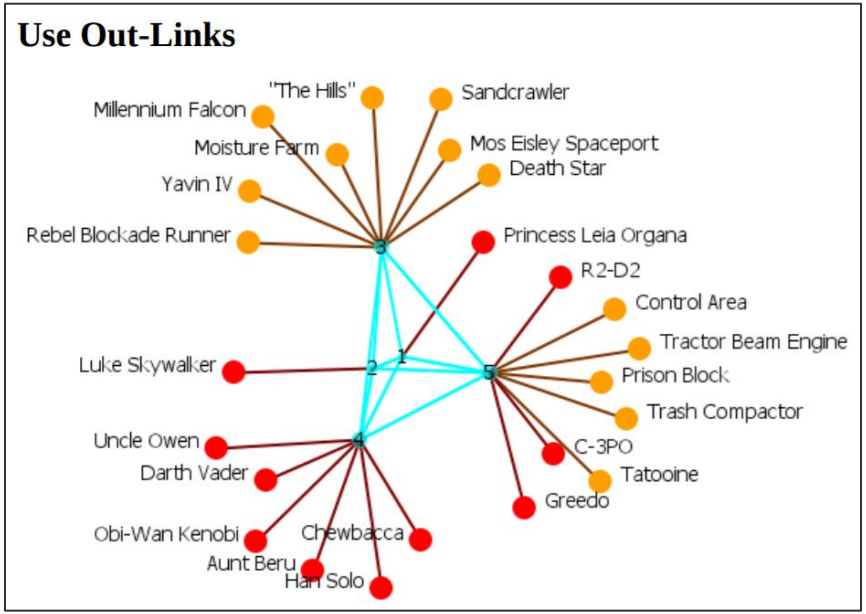


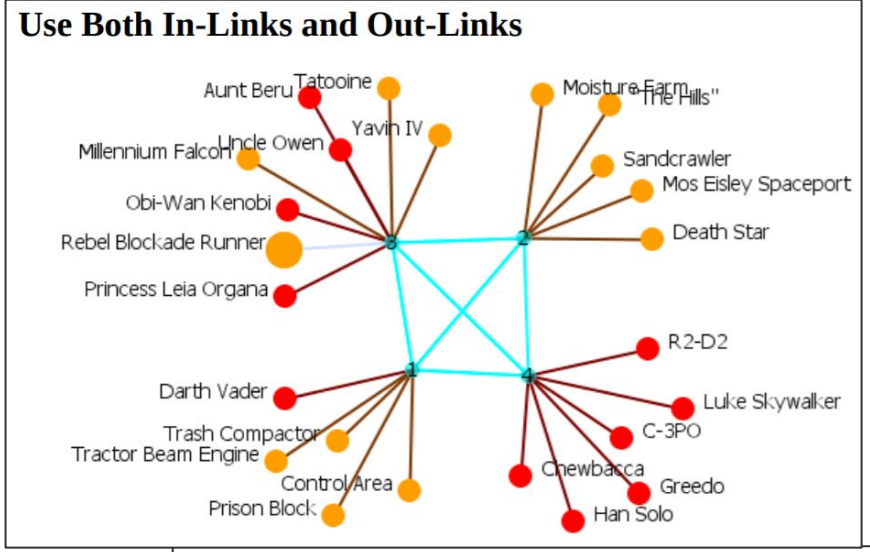


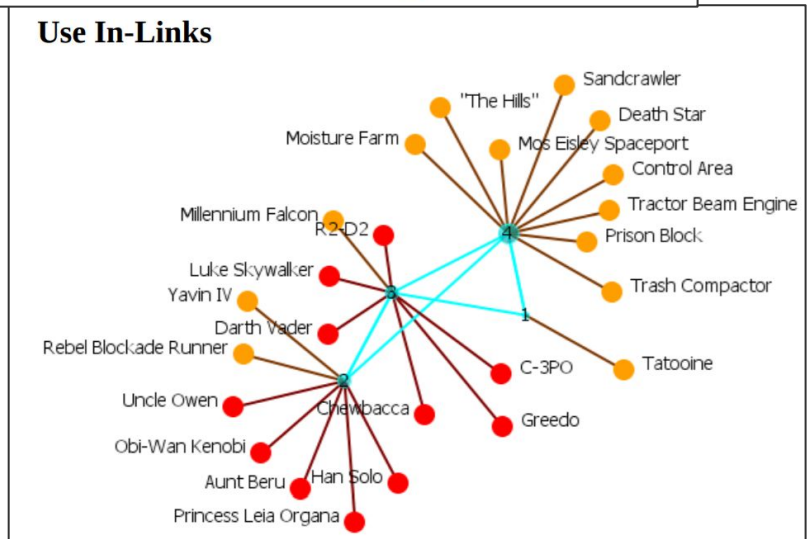
Images from: https://badhessian.org/2015/05/concor-in-r/

CONCOR

You can use outgoing-links (rows), incoming-links (columns), or both.







Images from: http://reports-archive.adm.cs.cmu.edu/anon/isr2019/CMU-ISR-19-100.pdf

Locate Groups: Algorithms (3)

- **Johnson:** (1967) Performs hierarchical clustering to create groups based on similarity scores
- **K-Means:** Partitions nodes into groups so as to minimize the within-cluster sum of squares
- **K-Cores:** Defines groups by recursively removing all nodes with fewer than k links (thus 0-core includes all nodes in a network while 2-core includes only nodes with two or more links).
- **Spectral Analysis**: Creates groups using eigenvectors derived from the input network(s).
- **K-Fog:** Identifies **F**uzzy **O**verlapping **G**roup membership using expectation maximization (EM) clustering rather than hierarchical (# of groups specified by the user).
    - Suitable for smaller networks (< 1000) (pdf).
- **Alpha-Fog:** Identifies fuzzy overlapping group membership using (α) parameter to encode the desired level of group cohesion (as opposed to specifying a number of groups).
    - Suitable for smaller network (< 1000).

Locate Groups: NLP Algorithms

- **Latent Semantic Analysis (LSA):** Derives topics (groups) for the row nodes by performing Singular Value Decomposition on the matrix to determine the correlations between nodes.
- **Latent Dirichlet Allocation (LDA):** Groups row nodes into topic group
  - Most common, basic topic analysis
  - Unsupervised classification of documents (or text content)
  - Each document is made up of various words, and each topic also has various words belonging to it. The aim of LDA is to find topics a document belongs to, based on the words in it.

Locate Groups: Measures & Algorithms

- Most grouping (clustering, community detection) work on one-mode networks.
  - The grouping algorithms may support analysis of both one and two mode networks.
- Local Patterns
  - Detects a variety specific patterns in the network (cliques, stars, chains...)
  - Nodes can be assigned to multiple groups

Which of these algorithms work with 1-mode vs. 2-mode?

One Mode

- Newman
- Girvan-Newman
- Louvain
- Johnson
- K-core

One and Two Mode

- CONCOR
- K-means
- Spectral Analysis
- K-Fog
- Alpha-Fog

Which algorithms generate single vs. multiple memberships?

At most one membership

- Newman
- Girvan-Newman
- Louvain
- CONCOR
- Johnson
- K-means
- Spectral Analysis

Multiple memberships possible

- k-Fog
- Alpha-Fog

What else can you do with these algorithms?

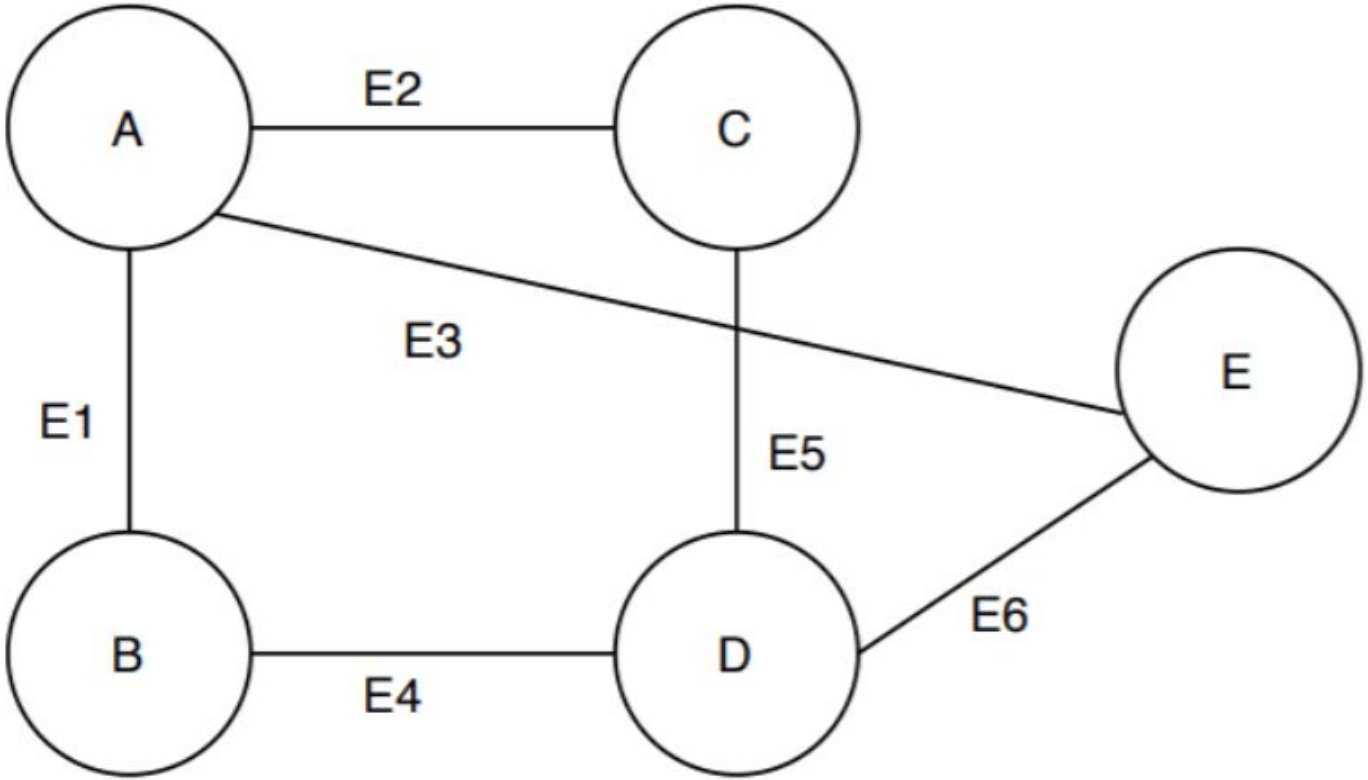
- Can be used to generate dendrograms
  - Only applies to hierarchical algorithms
  - Clique based techniques cannot give out hierarchical representations like dendrograms
- Can be used to color nodes in visualizations

How can you tell if a graph is Bipartite?

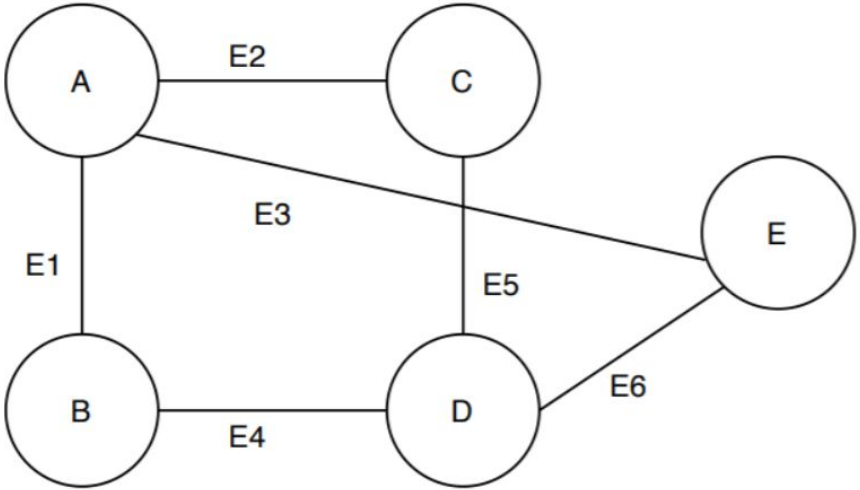
- Two conditions:
  - The node set of can be partitioned into two disjoint and independent sets N_1 and N_2.
  - All the edges have one endpoint node from N_1 and the other endpoint node from N_2.

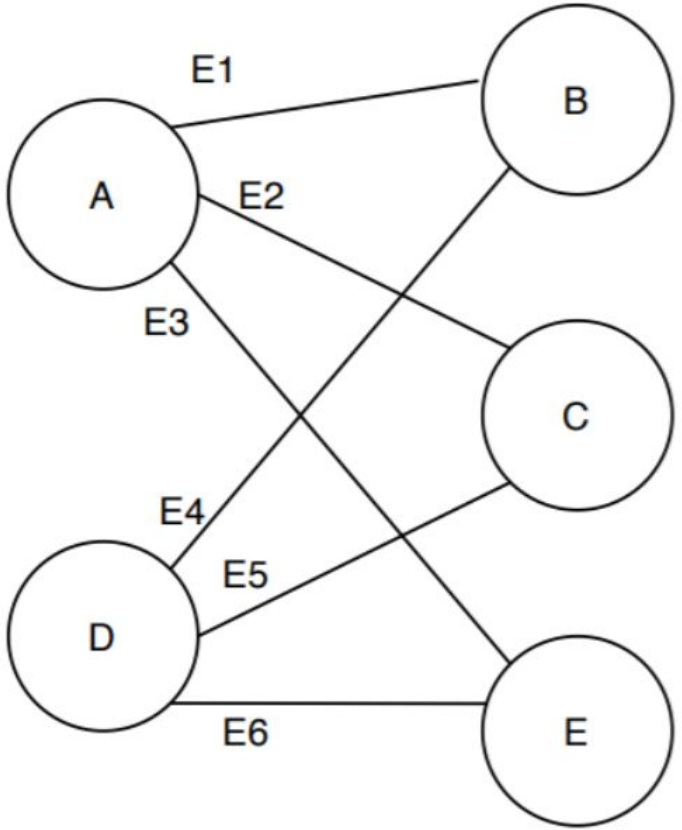
Bipartite graphs are superset of graphs with even-length cycles(subset)

Is this graph Bipartite?



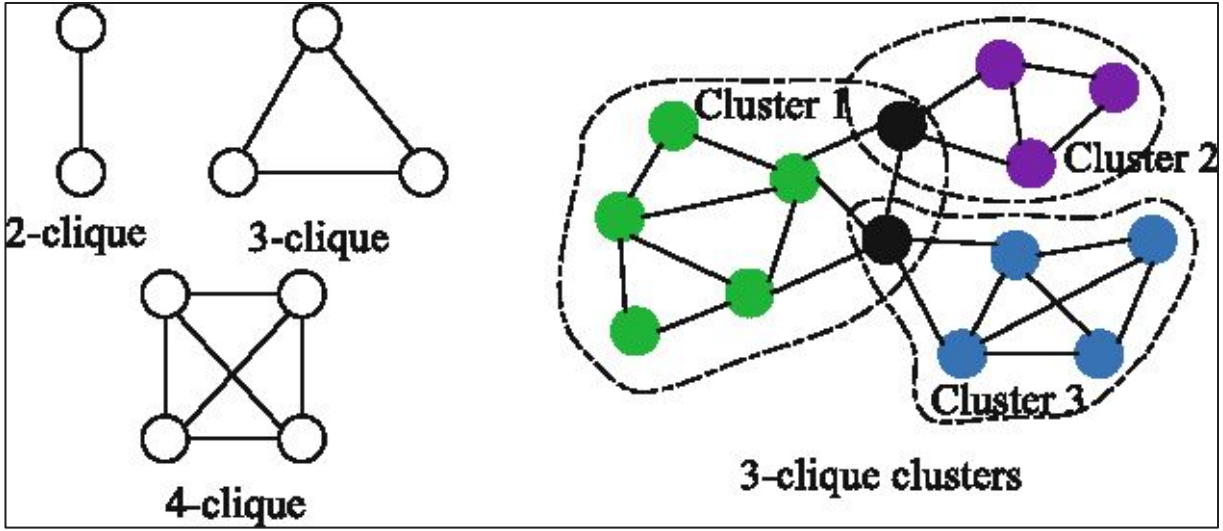
Is this graph Bipartite? - Yes!





Cliques

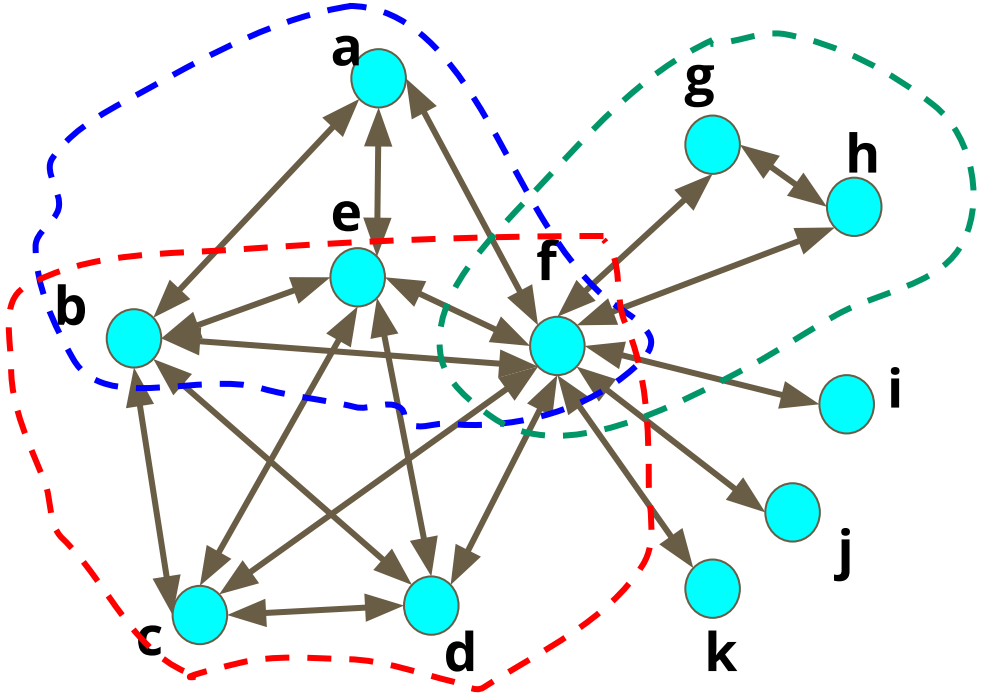
Cliques can make people feel left out!



- **Clique:** Max subset where all nodes are connected
- **K-Clique:** Clique with K members

- Cliques can overlap

How Many Cliques [Pen & Paper]



6 Cliques!

5-Clique: {b, e, f, c, d}

4-Clique: {a, b, e, f }

3-Clique: {g, h, f}

2-Clique: {f, i}

2-Clique: {f, j}

2-Clique: {f, k}

a-e-b would be a 3-clique, but it is counted within a-b-e-f 4-clique. Here, it is counted within the maximal set you can count it in.

Clustering Algorithms in Gephi

Clustering Algorithms in Gephi

- They are available as plugins of “Clustering” Category
- From Statistics > Network Overview, you can run
  - Modularity (it applies the Louvain method)
  - Leiden method
  - Girvan-Newman Clustering
  - DBSCAN

DBSCAN History

- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise.
- It is able to find arbitrary shaped clusters and clusters with noise (i.e. outliers).
- Was proposed in 1996
- In 2014, the algorithm was awarded the test of time award in ACM SIGKDD Algorithm (Award note)
- Covered in many textbooks, also used in Machine Learning. Many variants/implementations available

DBSCAN Parameters

All you need is a function to calculate distance between values and some guidance for what amount of distance is considered "close"!

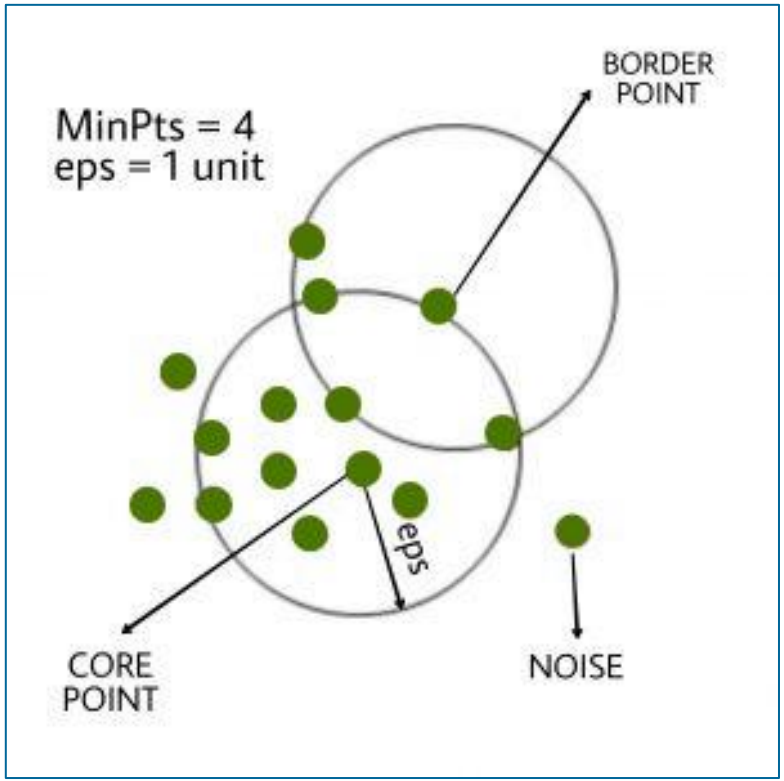
DBSCAN algorithm requires two parameters:

- **eps** : It defines the neighborhood around a data point. if the distance between two points is <= ‘eps’ then they are considered as neighbors.
  - If eps is too small then large part of the data will be considered as outliers.
  - If eps is too large then the clusters will merge and majority of the data points will be in the same clusters. You can define a function to find a proper eps value.
- **MinPts**: Minimum number of neighbors (data points) within eps radius.
  - Larger the dataset, the larger value of MinPts must be chosen.
  - MinPts must be >= 3.

DBSCAN Algorithm

3 types of data points.

- ***Core Point**: A point is a core point if it has more than MinPts points within eps.*
- ***Border Point**: A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.*
- ***Noise or outlier**: A point which is not a core point or border point.*



https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/

DBSCAN Algorithm

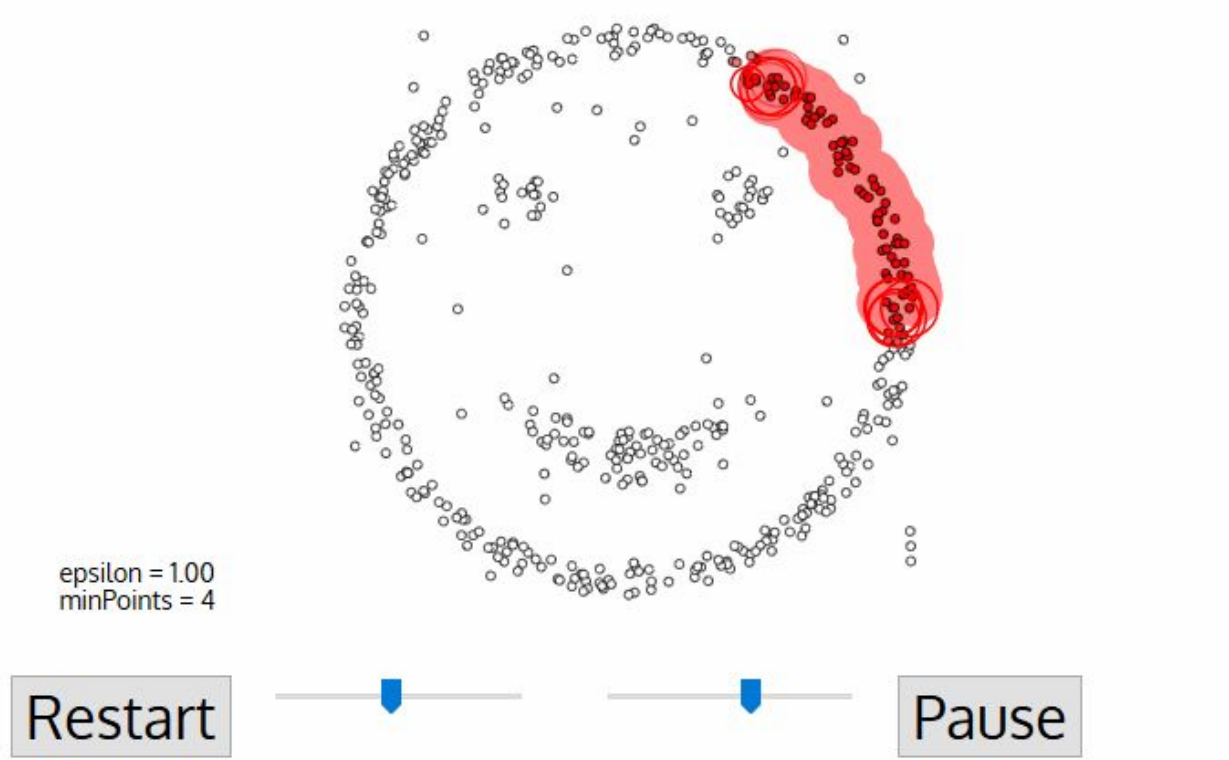
1. Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.
2. For each core point if it is not already assigned to a cluster, create a new cluster.
3. Find recursively all its density connected points and assign them to the same cluster as the core point.
   A point *a* and *b* are said to be density connected if there exist a point *c* which has a sufficient number of points in its neighbors and both the points *a* and *b* are within the *eps distance*. **This is a chaining process**. So, if *b* is neighbor of *c*, *c* is neighbor of *d*, *d* is neighbor of *e*, which in turn is neighbor of *a* implies that *b* is neighbor of *a*.
4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

```
DBSCAN(dataset, eps, MinPts){
# cluster index
C = 1
for each unvisited point p in dataset {
          mark p as visited
          # find neighbors
          Neighbors N = find the neighboring points of p

          if |N|>=MinPts:
              N = N U N'
              if p' is not a member of any cluster:
                  add p' to cluster C
}
```

https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/

DBSCAN



Img src: https://www.digitalvidya.com/blog/the-top-5-clustering-algorithms-data-scientists-should-know/

DBSCAN vs. k-means

- K-Means (distance between points)
- DBSCAN (distance between nearest points)
- In *k*-means, you need to specify the number of clusters ("*k*") to use it.
    - Most of the time, you won't know what a reasonable *k* value.
- DBSCAN does not require a pre-set number of clusters.

DBSCAN vs. k-means

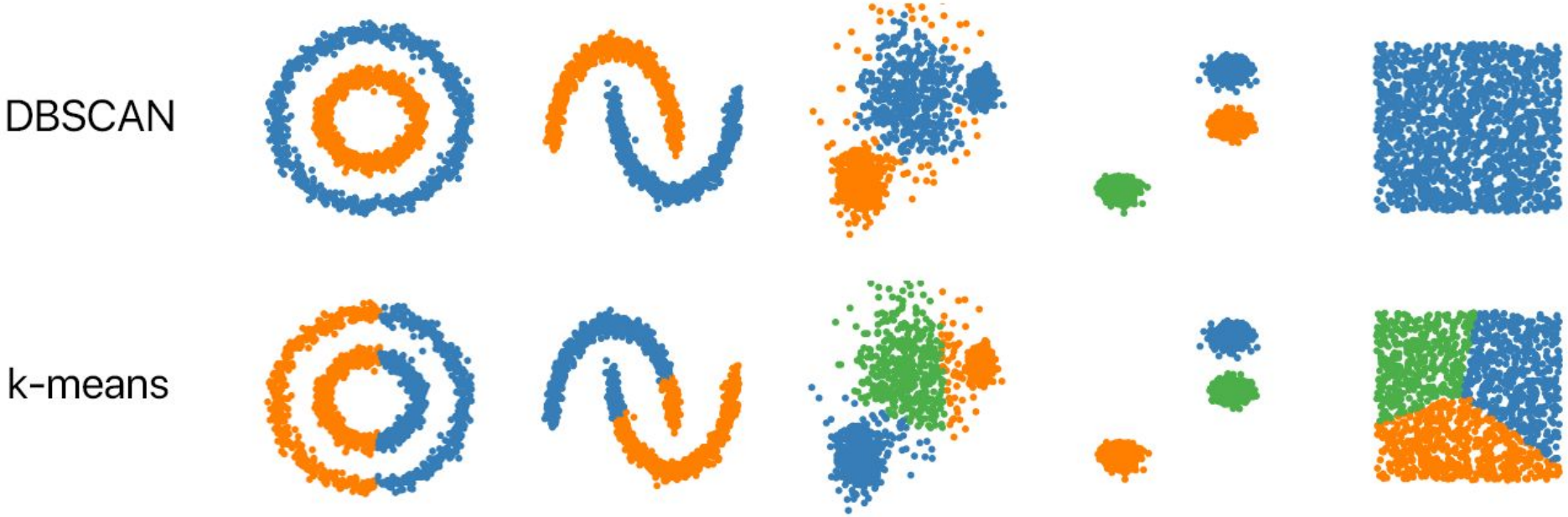


Image src: https://github.com/NSHipster/DBSCAN

Next Lecture:

Network Evolution