Social Networks Analysis

- Social Networks Analysis and its Applications
- Network Descriptive and Network Connectivity Analytics
- Heavy-Tailed Degree Distributions
- Small World Phenomenon
- Large-Scale Network Structure
- Community Structures in Networks
- Community Detection Algorithms
- Community Detection in Temporal/Dynamic Networks

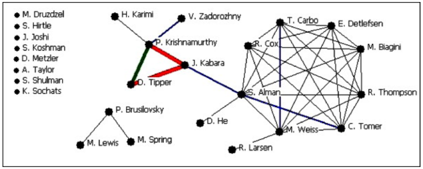
Social Networks

Social Network

A social network is a theoretical structure used to study relationships between individuals, groups, organizations, or entire societies

- Social Networks: people, friendship/acquaintance/coworker
- Online Social Networks (OSN): people, friendship or co-worker relation
- Coauthorship Networks: coauthorship betweeen authors

Social Network



Coauthorship Network

Online Social Nework

Online Social Networks

Social networks allow users to connect, communicate, share information

- Personal communication: Connecting with family and friends
- Professional networking: e.g., LinkedIn helps people connect with colleagues and explore career opportunities
- Information dissemination: Twitter and Facebook are key sources for real-time news and updates
- Multimedia Sharing: YouTube, Instagram, and TikTok focus on content creation and consumption
- Global Reach: Breaks down geographical barriers
- Crowdsourced Content: Users contribute to the creation and dissemination of content, from news to entertainment
- Political and Social Movements: Twitter and Facebook have been instrumental in organizing movements and influencing public opinion
- Virality: Social networks enable content to reach large audiences quickly, creating viral trends

The complex dynamics and massive scale of these networks make them a significant area of study in sociology, computer science, and data science

Early Development and Evolution of Facebook

- Started as “Thefacebook” exclusively for Harvard students
- Quickly expanded to other Ivy League institutions
- 2006: Opened to anyone aged 13+ with a valid email address, introduced the News Feed, revolutionized content distribution
- Unique selling point: A profile-based platform where users connect through friend requests, share status updates, and post on walls

- Shifted from a college-exclusive network to global social platform
- 2007: introduced Pages for businesses and public figures
- 2009: “Like” button- fundamentally changing user interaction

- 2012: Facebook went public, raising $16 billion in its IPO
- 2012: Acquisition of Instagram & WhatsApp
- 2016: Facebook Live allowing real-time video streaming
- 2021: Integrating social networks with the virtual world (Metaverse)
- 2024: Facebook has over 2.8 billion active users

History and Growth of Twitter

Twitter is popular for its microblogging concept (users post updates (tweets) restricted to 140 characters (later 280 characters)

- 2006: Launch as a microblogging service for short, real-time updates
- 2007: Hashtag (#) feature, for users to tag and follow topics
- 2008: Gained traction during the U.S. Presidential elections, becoming a critical platform for political discussions
- 2008-2012: Became a go-to platform for real-time news
- The Arab Spring (2010–2011) played role in real-time information dissemination and activism
- 2013: Goes public with a valuation of $14.2 billion
- Post-2016: Features like live streaming (Periscope) and integrated multimedia content
- 2020s: A significant platform for news, political discourse, and real-time event sharing

Social Networks as Platforms for Global Interaction

Social networks enable global interaction by providing platforms for:

- Cultural Exchange: Users share content across cultural boundaries
- Collaborative Projects: Facilitate collaborations across boundaries
- Education: Distance learning and global knowledge sharing through forums, webinars, and online courses
- Social Organization and Movements
    - Digital Activism: Social platforms are central for organizing protests and spreading awareness about social issues
    - Global movements like #FridaysForFuture, #MeToo, and #BlackLivesMatter gained momentum through social networks
    - Organizational and Coordination: Social networks help organizers coordinate events and actions in real-time
    - Amplification: Social media gives visibility to causes, allowing local events to gain global attention
    - Mobilization: Platforms enable rapid mobilization of resources and participants for movements

Social Networks Analysis and its Applications

Computational Questions in Social Network Analysis

Key computational challenges include:

- Influence Maximization: Identifying a subset of nodes to maximize the spread of influence.
- Community Detection: Algorithms to identify communities within networks.
- Rumor Spread and Blocking: Formulating strategies to control the flow of information in networks.

These problems are essential for understanding and optimizing the dynamics of social networks.

Applications of Social Networks

Social networks have applications across various fields including information science, biology, economics, sociology, and communication studies. It involves the study of social actors, or nodes, connected by social relations such as friendship, kinship, or professional collaboration

- Social Influence: Online social networks influence individuals' behaviors, opinions, and emotions
- Sentiment Analysis: Used to predict political outcomes by analyzing social media posts
- Community Detection: Helps in identifying groups of users with common interests or behaviors
- Rumor Blocking: Strategies to stop the spread of misinformation in networks

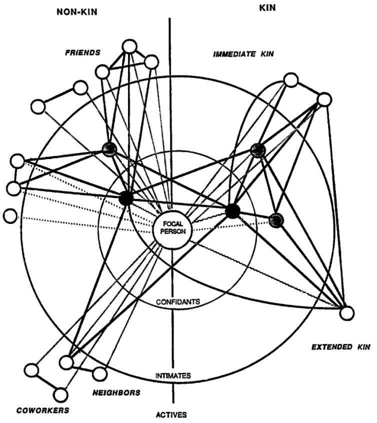
In practical scenarios, computational social network analysis aids in:

- Predicting election outcomes by analyzing social media data
- Designing efficient algorithms for rumor blocking and influence maximization
- Enhancing marketing strategies by targeting influential individuals in a network

Social Network Analysis

Network Perspective of Society

How the Individuals', communities' and society's behavior is influenced by their social connectivity

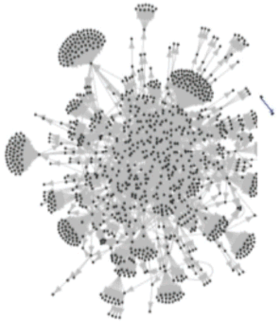


An early use of network analysis in sociology. This diagram of the 'ego-network' shows varying tie strengths in concentric circles-Wellman 1998

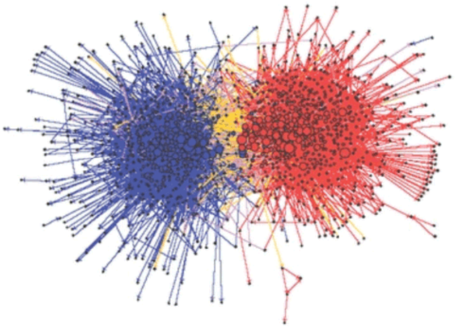
Social Network Analysis

Network Perspective of Political discourse

source: Ulicny, Kokar, Matheus, (2010)



(a) Malaysian Sopo blogosphere

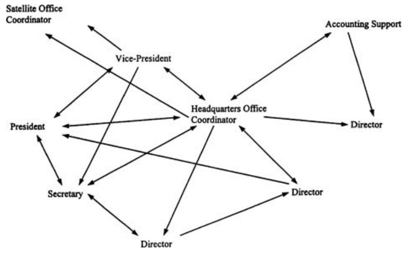


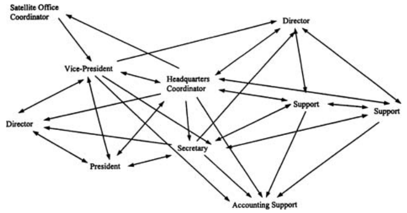
(b) US political blogosphere (Adamic & Glance, 2005[4])

A visualization of Malysia and US blogospheres (nodes are blogs and edges are links to blogs). Left reveals importance/credibility/popularity of blogs, while the right visual (two dense clusters with little interaction with the other cluster) shows that bloggers are more likely to link to bloggers with the same party affiliations

Social Network Analysis

Communication within an organization





Intra-organization communication before and after implementing a content management system (Garon et al 1997)

Early Network Analysis in Social Science

Linton Freeman (1996), Some Antecedents of Social Network Analysis

Network analysis is applied in Educational Psychology, Child Development, Sociology, Anthropology, Political Science, Information Science

> Society is not a mere sum of individuals. Rather, the system formed by their association represents a specific reality which has its own characteristics. . . The group thinks, feels, and acts quite differently from the way in which its members would were they isolated. If, then, we begin with the individual, we shall be able to understand nothing of what takes place in the group
>
> Émile Durkheim, The Rules of Sociological Method (1895)

Durkheim defined "social facts"-

a phenomenon that is created by interactions of individuals but it is independent of any individual

Network Descriptive Analytics

Network Descriptive Analytics

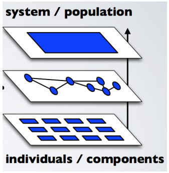
What is the structure of the Network? How does the network look like?

- What is the magnitude of the graph?
- How are the edges organized?
- How do vertices differ?
- Does Network location matter?
- Are there underlying patterns?
- What process shape these networks?
- How does the network structure shape the network function?
- What is the underlying reason for this structure?
- How can we exploit the structural features of the network?

Network Descriptive Analytics

Features of the network

- Local-Level Features
- Large scale-features
- Global Features



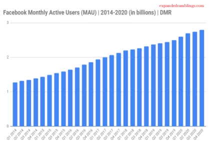
- Order, size, density
- Average degree, degree sequence, degree distribution
- Vertex positions and centrality
- Shortest loop density (triangles)
- Connectivity of the network
- Shortest path, radius, diameter
- Small world graphs

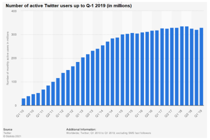
Network Descriptive Analytics: Magnitude

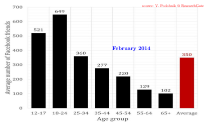
Network magnitudes have direct impact on the storage, computation, and communication, visualization complexity of the network

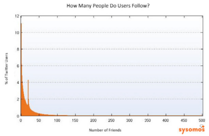
Order of Graph: Number of vertices

Size of Graph: Number of edges



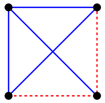






Density of the Network Ratio of number of edges present in the network to number of edges possible in (simple) network

Suppose $G = (V, E)$, $|V| = n$, $|E| = m$ $\quad d(G) = \dfrac{m}{\binom{n}{2}}$



$\mathbf{d(G)} = \frac{4}{6}$

- A single parameter to compare connectivity of two graphs
- Very useful in comparing subgraphs - Clusters are dense subgraphs
- A clique has density 1, an independent set has density 0

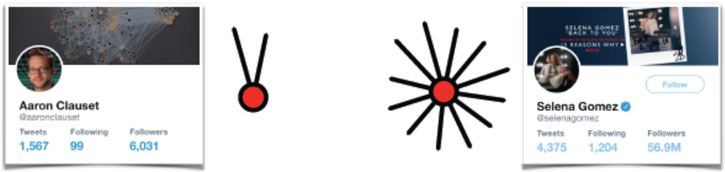
Degrees and Average Degree

Degree of a vertex v : is number of edges adjacent to v, $d(v) = |N(v)|$

in-degree and out-degree, in and out-neighborhood in digraphs

In a bipartite graph $G = (A, B, E)$ degree of $v \in A$ is usually normalized by $|B|$

source: Aaron Clauset @ UC Boulder



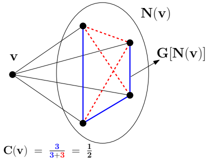
Average degree is the average degree over all vertices

$$d_{av}(G) = \frac{\sum_{v \in V} d(v)}{n} = \frac{2m}{n} = 2d(G)$$

Clustering Coefficient and Transitivity

Clustering Coefficient of $v, C(v) = \frac{|E(G[N(v)])|}{\binom{d(v)}{2}}$

$E(G[N(v)])$ is edges in graph induced by $N(v)$



clustering coefficient of graph $C(G)$: average clustering coefficient over all nodes

In some text $C(G)$ is defined as $C(G) = \frac{3 \times t(G)}{\sum_v \binom{d(v)}{2}}$

$t(G)$ is the number of triangles in G

Transitivity is the overall probability of nodes adjacent to a node being adjacent

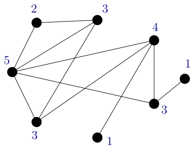
- Reveals existence of tightly connected communities (clusters/cliques)

Heavy-Tailed Degree Distributions

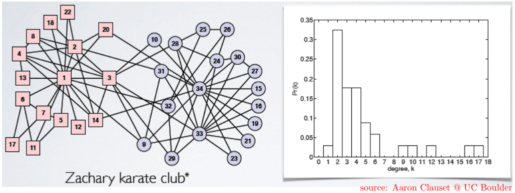
Exploring Degree Distribution

Degree Distribution

Pr(*k*): Probability that a randomly chosen vertex has degree *k*



k	$Pr(k)$
1	$2/8$
2	$1/8$
3	$3/8$
4	$1/8$
5	$1/8$
6	0
7	0

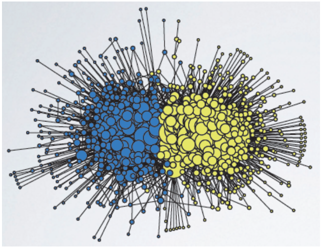


source: Aaron Clauset @ UC Boulder

- Degree distribution of a network describe its structure
- It is also used to determine the appropriate synthetic graph generation, to explain the observed structural patterns
- Typically real-wold graphs have heavy-tailed (power law) degree distributions
- Statistical signature of power and heavy tail: linear plot on a log-log scale

Exploring Degree Distribution

$Pr(k)$: Probability that a randomly chosen vertex has degree k

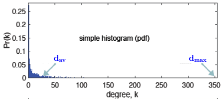


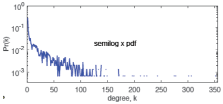
Simple pdf: $Pr(k)$ vs. k

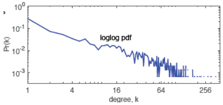
Semilog x pdf: $\log Pr(k)$ vs. k

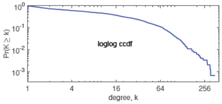
loglog x pdf: $\log Pr(k)$ vs $\log k$

complementary cdf: $\sum_{i=k}^{n} Pr(j)$ vs. k







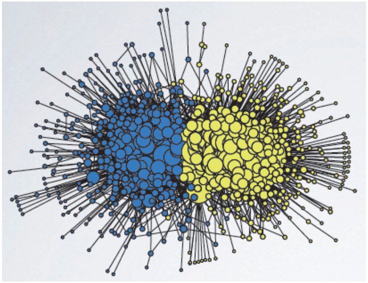


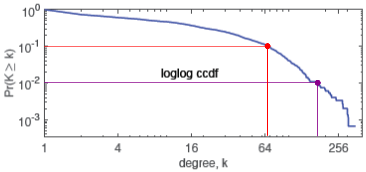
Exploring Degree Distribution

$Pr(k)$: Probability that a randomly chosen vertex has degree k

complementary cdf: $Pr(deg \geq k) = \sum_{i=k}^{n} Pr(deg = j)$ vs. k

Complementary cdf is monotonic, smoother than pdf and reveals more info





90% vertices have degree ≤ 67 (accounting for 53% all edges)

1% vertices have degree ≥ 169 (accounting for 90% all edges)

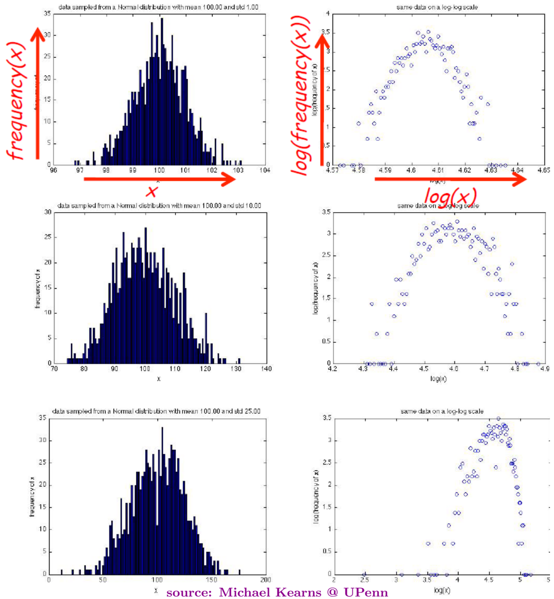
Heavy-Tailed Degree Distributions

Not Heavy-Tailed distribution (e.g., normal, Poisson distributions) are good for modeling many real-world quantities . . . but not degree distributions

If mean/average is μ then probability of value x, $Pr(x) \propto e^{-(x-\mu)^2}$

▷ exponentially fast decay as x moves away from μ

$\log(Pr(x)) \propto -(x-\mu)^2 \implies \log(x)$ vs $\log(Pr(x))$ plot has strong curvature!



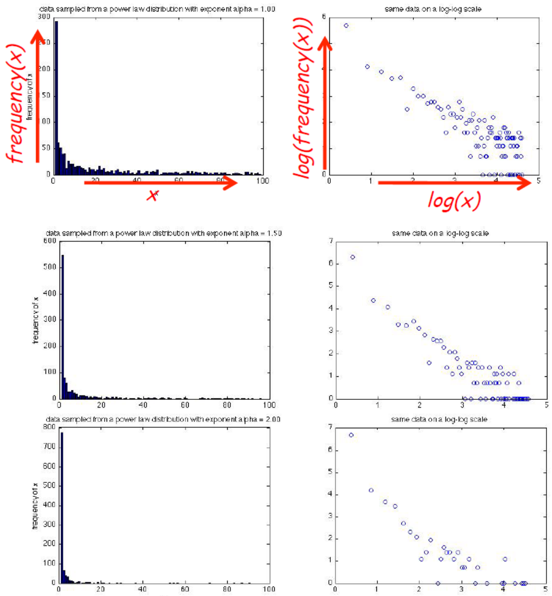
source: Michael Kearns @ UPenn

Heavy-Tailed Degree Distributions

One mathematical model of a typical “heavy-tailed” distribution: is the Power Law distribution with exponent β
$Pr(x) \propto \frac{1}{x^\beta}$

▷ exponentially fast decay as x increases

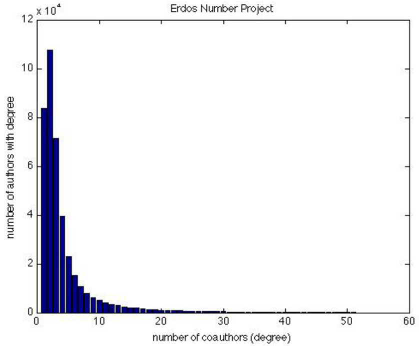
$\log(Pr(x)) \propto -\beta \log(x) \implies$
$\log(x)$ vs $\log(Pr(x))$ plot is a straight line!

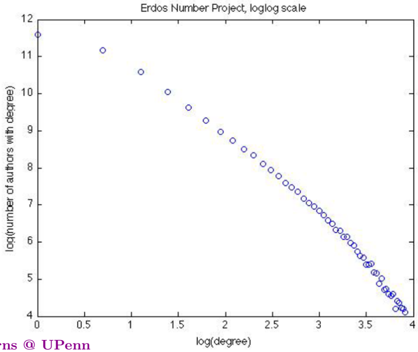


source: Michael Kearns @ UPenn

Exploring Degree Distribution

Nearly all real-world networks have a heavy-tailed degree distribution



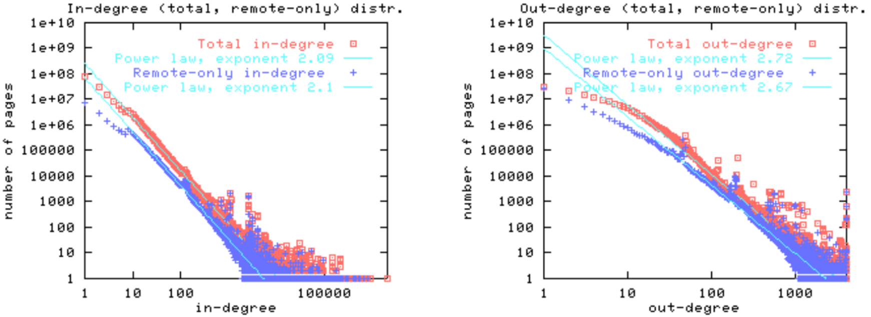


source: Michael Kearns @ UPenn

Exploring Degree Distribution

Nearly all real-world networks have a heavy-tailed degree distribution

source: Border et al. (2009): Graph structure in the web

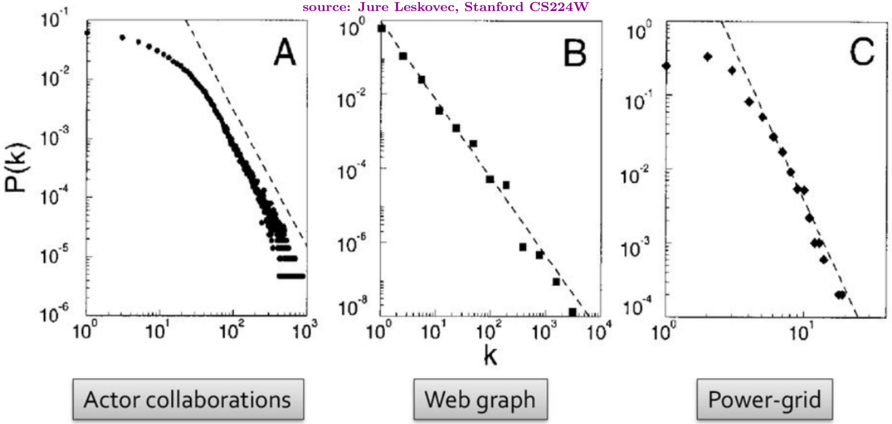


Figures 1 and 2: In-degree and out-degree distributions subscribe to the power law. The law also holds if only off-site (or "remote-only") edges are considered.

Exploring Degree Distribution

Nearly all real-world networks have a heavy-tailed degree distribution

source: Jure Leskovec, Stanford CS224W



The dashed lines have slopes (A) $\beta_A = 2.3$**, (B)** $\beta_B = 2.1$ **and (C)** $\beta_C = 4$

Exploring Degree Distribution

Nearly all real-world networks have a heavy-tailed degree distribution

source: Michael Kearns @ UPenn

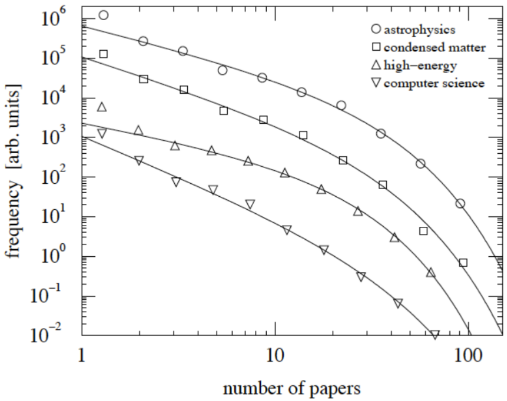


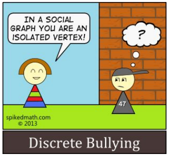
FIG. 2. Histograms of the number of papers written by scientists in four of the databases. As with Fig. 1, the solid lines are least-squares fits to Eq. (1).

Network Connectivity Analytics

Connectivity Analytics

In social network analysis, understanding connectivity is crucial to identifying the resilience and robustness of a network. Key concepts include:

- Connectivity coefficient: Minimum number of nodes needed to remove to disconnect a graph (useful in network fragility and robustness analysis)
- Connectivity: Node X is reachable from node Y **OR** Y is reachable from X
- Strong connectivity: Node X is reachable from node Y **AND** node Y is reachable from node X
- Fully connected graph: Each node has edges to all other nodes in the graph
- Clique: A subset of vertices in which every two distinct vertices are adjacent
- Terminal node: A node with no outgoing edges
- Unreachable node: A node with no ingoing edges



Shortest Path, Geodesic Distance, Radius and Diameter of a Network

Geodesic distance $d(u, v)$: distance between u and v is the length of the shortest path b/w u and v

Underlying assumption that things being equal communication takes place using shortest paths

Some measures of graph connectivity (reachability) compare it to network density

Average distance: The average shortest path over all pairs of vertices

$$\ell_G = \frac{\sum_{u,v \in V} d(u,v)}{\binom{n}{2}}$$

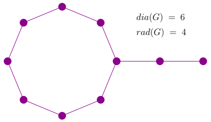
- A measure of network cohesion, efficiency of communication
- Indicates how far apart any two nodes are on average

Network diameter: The longest geodesic distance between a pair of vertices

$$dia(G) = \max_v \max_u d(v,u)$$

Network radius: The minimum maximum distance of a vertex

$$rad(G) = \min_v \max_u d(v,u)$$



Denser networks are likely to have small diameter and vice versa

Small World Phenomenon

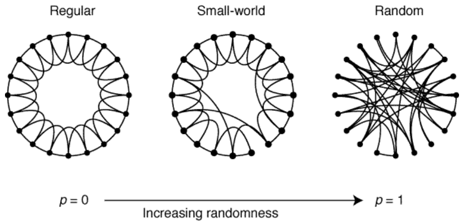
Small World Phenomenon

Real-world networks though are very sparse yet their diameters are typically small

A small world network is one in which most pairs of nodes are not adjacent (sparse) but they have larger clustering coefficients and pairs are reachable in a few hops (low diameter)

Small world graphs are in-between random graphs and regular graphs

- Regular graphs: All nodes have equal degrees
- Erdös-Renyi graph $G(n, p)$: n nodes and a pair is adjacent with prob. p
- $G(n, m)$ graphs: Randomly from all graphs on n nodes and m edges



Small World: Milgram's Experiment

This phenomenon was first popularized by Stanley Milgram's experiment in the 1960s, which suggested that people are connected by an average of six acquaintances—coining the phrase "six degrees of separation."

Random individuals in Omaha NB were asked to deliver a letter to a target person in Boston MA – Participants could only send the letter to someone they knew on a first-name basis

- Average lengths of successful chain was about 6
- Many did not reach and many reached via the same intermediaries

Illustration of Milgram's Small-World Experiments

START
TARGET

Real-World Examples of the Small World Phenomenon

Many real-world networks exhibit small-world properties

- Facebook Study (2016) found that the average degree of separation between any two users was 3.57 – Most users are separated by a small number of connections, despite the size of the network
- Professional networks (e.g., LinkedIN) show a small-world effect, where users are typically 2 to 3 hops away within industries
- Neural networks in the brain are highly clustered with short path lengths, enabling efficient signal transmission across regions
- The World Wide Web demonstrates small-world characteristics
- Email communication networks studies have found that users are often separated by around 5 to 6 steps

Watts-Strogatz Model of Small-Worlds

The Watts-Strogatz model generates small-world graphs with following features

- A small average path length (similar to random graphs)
- A high clustering coefficient (similar to regular lattices)
- Allows for the emergence of “small-world” properties from a regular graph

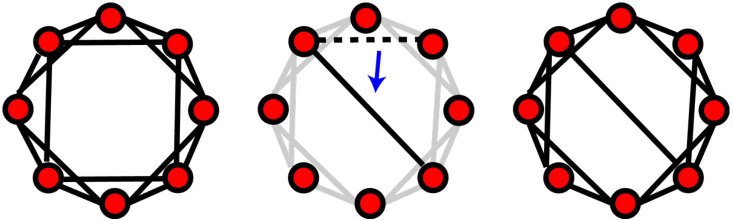
- Social Networks: Helps to model human social networks, where people are often connected to close friends and a few distant connections
- Neural Networks: Used in modeling brain networks, where neurons are highly clustered but also able to communicate efficiently over longer distances
- Infrastructure Networks: Applied in studying the robustness and communication efficiency in transportation or utility networks

These networks exhibit both local clustering and global reachability, essential for efficient network communication.

Watts-Strogatz Model of Small-Worlds

Watts-Strogatz Model: Generation Process

- Start with a regular ring lattice $R(n, k)$ on n nodes, where each node is connected to its k nearest neighbors ($^k/_2$ on either side)
- For each edge, with probability p, rewire the edge to a random node (from (u, v) to (u, w) for randomly chosen w) ▷ Randomly rewiring breaks regularity, creates shortcuts that significantly reduce the average path length



The network still retains a high clustering coefficient but has much shorter average path lengths, resembling small-world networks

Clustering Coefficient and Average Path Length

$E_0 = |E(R(n,k))| = {}^{nk}/_{2}$ Rewiring introduce $\sim p^{nk}/_{2}$ non-lattice edges

$L(p)$: the average length of shortest paths between all pairs of nodes

$C(p)$: fraction of a node's neighbors that are also neighbors of each other

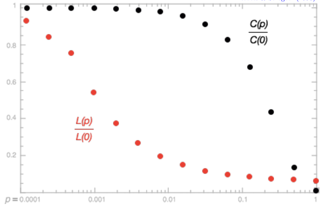
- For a regular lattice ($p = 0$), $L(0) \simeq {}^{n}/_{2k}$
- For a random graph ($p = 1$), $L(1) \approx {}^{\ln n}/_{\ln k}$

As p increases, the path length decreases, mimicking random graphs' behavior

- For a regular lattice ($p = 0$), $C(0) = {}^{3(k-2)}/_{4(k-1)}$
- For a random graph ($p = 1$), $C(1) \to {}^{k}/_{n-1}$

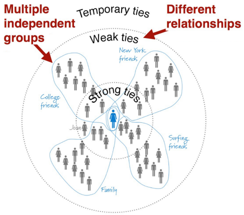
As p increases, the random rewiring reduces the local clustering

Watts & Strogatz (1998)

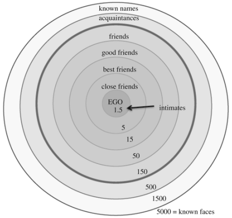


Strength of Weak Ties

In many networks all edges are not the same



source: towardsdatascience.com



https://royalsocietypublishing.org/doi/10.1098/rspa.2020.0446

- Structure of human egocentric social networks
- Number of people included in each circle increases, but the frequency of contact and emotional closeness declines, with each layer
- The outermost layer (5000) was identified by face recognition experiment (Num of faces that can be recognized as known by sight)
- Biological networks: tie strength based on biochemical interaction
- Computer networks: based on link bandwidth

Strength of Weak Ties

Granovetter,'s "The strength of weak ties": "Most job seekers (study subjects) found jobs through an acquaintance (weak tie), rather than a close friend (strong tie)"

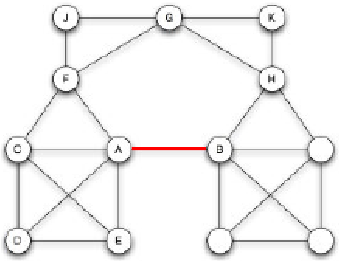
- Information at end-points of a strong tie is nearly identical
  ▷ frequent synchronization
- weak tie could help communication of novel information
  ▷ rare synchronization
- Acquaintance can more likely inform of "new" job opportunities

There are some vertices (and edges) that act as bridges between network segments, they are important for communication and explain the small-world phenomena in many network

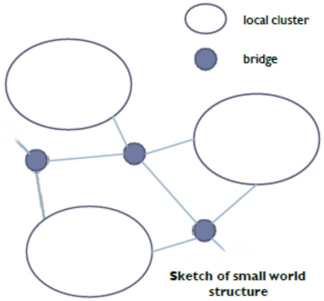
Bridges

There are some edges that act as bridges between network segments, they are important for communication and explain the small-world phenomena in many networks

An edge (i, j) is a local bridge if i and j have no friends in common



source: Frank Dignum @ Umea University



Preferential Attachment

- A mechanism in which a quantity (e.g. wealth, credit, degree) is distributed among objects according to how much they already have
- aka *rich gets richer, early bird advantage, cumulative advantage*

Popularity

We want to be associated with popular people, ideas, items, thus further increasing their popularity, irrespective of any objective, measurable characteristics

Also known as 'the rich get richer'

Quality

We evaluate people and everything else based on objective quality criteria, so higher quality nodes will naturally attract more attention, faster

Also known as 'the good get better'

Mixed model

Among nodes of similar attributes, those that reach critical mass first will become 'stars' with many friends and followers ('halo effect')

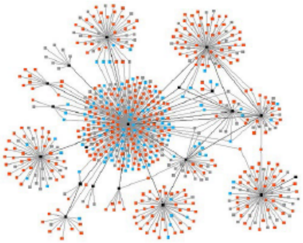
May be impossible to predict who will become a star, even if quality matters

source: Dr. Giorgos Cheliotis @ NUS

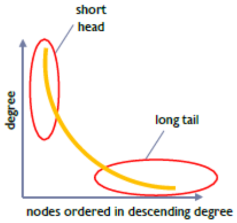
Preferential Attachment

In networks generated via preferential attachment process a great majority of new edges are incident to nodes with an already high degrees - degrees of these nodes increase disproportionately

- Results in network with few very high and majority low degrees nodes
- These networks have long-tailed degree distribution
- Tend to have small-world structure
- Transitivity and strong/weak tie characteristics are not necessary to explain small-world structure



source: Dr. Giorgos Cheliotis @ NUS



Barabási–Albert model

- Generate random networks using preferential attachment process
- WWW, citation networks, the Internet, and some OSN have long-tail degree distribution ▷ BA model tries to explain them

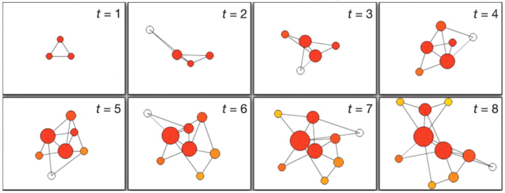
Initialize with a complete graph on m_0 nodes

Each new node has $m \leq m_0$ edges (dangling)

A dangling edge is adjacent to an existing node v_i with probability $p_i = \dfrac{d(v_i)}{\sum_j d(v_j)}$

High degree nodes quickly accumulate more edges ▷ rich get richer

Barabassi: (2009) Scale-Free Networks: A Decade and Beyond



degree distribution $P(k) \sim k^{-3}$ ▷ Power law with scale parameter 3

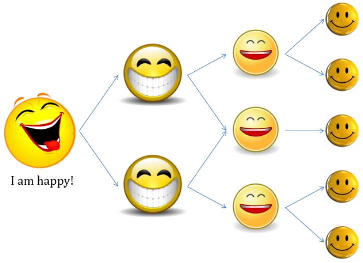
Three Degrees of Influence

Three Degrees of Influence in friendship network

Influence in social networks extends to three degrees (Christakis and Fowler):

- First degree: Direct influence on a person
- Second degree: Influence on a friend of a friend
- Third degree: Influence on a friend of a friend's friend

Beyond the third degree, influence becomes negligible. This phenomenon explains why certain behaviors, such as happiness or political opinions, can spread within a network, but influence dissipates with distance.



Six Degrees of Separation and Three Degrees of Influence

While Six Degrees of Separation suggests that everyone is connected through a chain of acquaintances, the Three Degrees of Influence principle shows that influence in social networks is much more limited

The small world effect suggests a network of tightly clustered communities, but the influence effect shows that behaviors or information often fail to propagate beyond three steps

The interplay between these two concepts highlights:

- Global connectivity (Six Degrees) vs. local influence (Three Degrees)
- Information or influence spreads quickly within small communities, but only weakly between distant communities
- Influential nodes (e.g., key people in the community) are often more critical for spreading influence than the total number of connections

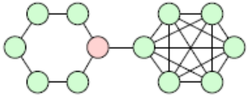
Thus, although people are globally connected, real influence is typically confined to a much smaller, local network.

Large-Scale Network Structure

Large-Scale Network Structures

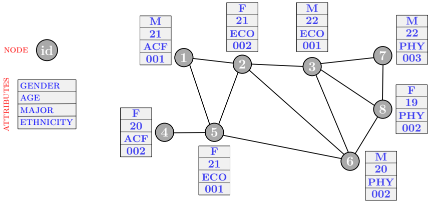
- Vertex-level structural measures (degree, centrality) are local and don't reveal the global structure of the network
- Network-level measures (average degree, clustering coefficient, radius, diameter) reveal network shape but can be unstable (sensitive to non-uniformity)

- Vertex level measures (distributions thereof) provide some insights into structural heterogeneity, e.g. degree/centralities distribution
- Don't reveal organization patterns, do high degree nodes form cliques?
- Graph measures over aggregate, vertex measures over disaggregate


Nearly all value of clustering coefficient comes from the clique

Heterogeneity in Networks

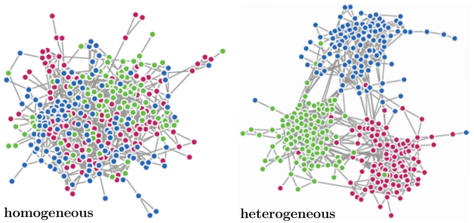
Nodes in a social networks can have attributes like gender, age, political affiliation, interests, ...



Network heterogeneity: Understanding how nodes differ in terms of attributes

- What is network's organizational pattern of structural heterogeneity?
- Quantify the tendency of vertices with similar characteristics to be found close to each other in network (or the lack of this property)

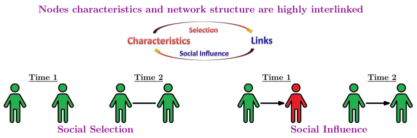
Homogeneous and Heterogeneous Mixing: Analyzes how vertices with different attributes interact



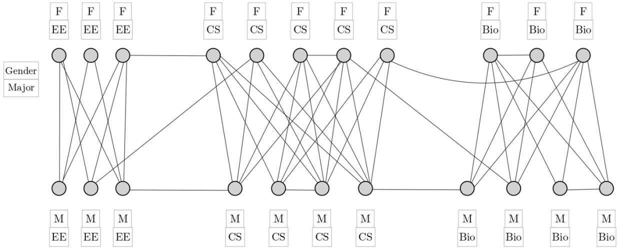
Social Influence and Social Selection – Homophily and Hetrophily

Two important phenomena in Sociology

- Social Selection: Individual's attributes drive the interaction with others
- Social Influence: Interactions among people shape people's attributes



- Homophily: Connections among nodes having same attribute values
  ▷ assortative mixing
- Heterophily: Connections among nodes having different attribute
  ▷ disassortative mixing



'MAJOR' attribute is homophilic

'GENDER' attribute heterophilic

Community Structures in Networks

Graph Structures in Social Media: Cliques and Circles

In social networks, friendship networks can exhibit specific graph structures

- Cliques: Represent tightly-knit groups of friends, family, or colleague
  ▷ A small group of Facebook friends who frequently interact with each other's posts forms a clique
- Circles: Looser connections where a user is connected to multiple groups but not all connections are mutual. Represents broader social connections
  ▷ A person may belong to multiple circles such as work colleagues, school friends, and interest-based groups, with varying degrees of overlap

Analyzing graph structures like cliques and circles helps understand community dynamics and the diffusion of information in social networks

- Communities are critical in understanding the flow of influence because influence tends to spread faster within highly connected groups
- Community structure can impact the rate of adoption of behaviors or ideas, especially when a node in one community influences a node in another
- Networks with strong clustering tend to have robust diffusion of information within communities, which can then spread to other groups

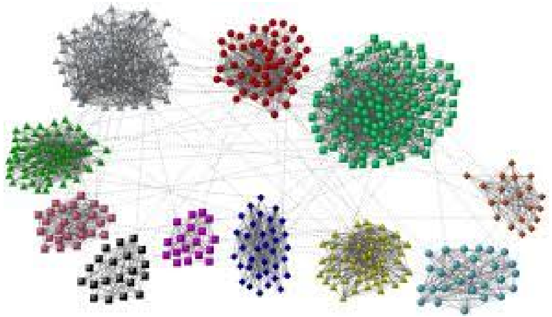
Clusters in Graphs and Community Structure

Modular or community structure in networks: homogeneous building blocks of the larger heterogeneous structure

Communities form through a combination of factors such as homophily, geographical proximity, or shared interests

Cluster: A dense subgraph within a graph

- Cohesion: Nodes are more similar/adjacent to other nodes in the same cluster
- Separation: Nodes in a cluster are dissimilar to nodes in other clusters



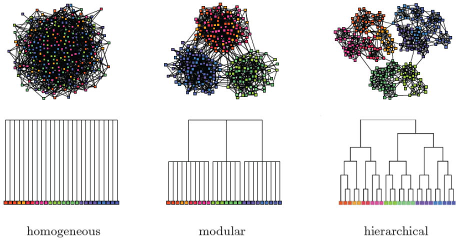
Communities in Graphs

Modular or community structure in networks: homogeneous building blocks of the larger heterogeneous structure

- People in the same community share common interests in clothes, music, beliefs, movies, food, etc.
- They influence each other strongly

Community structure alone is a single level of organization and reveals little information about structure within and between communities

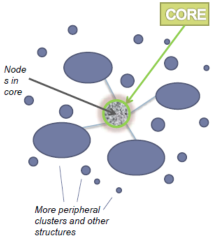
Hierarchical community structure e.g. CS department inside SSE inside LUMS inside Lahore ...



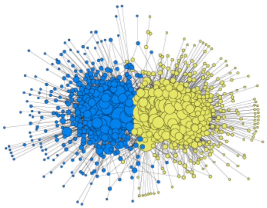
Core-periphery Structure in Graphs

Central dense communities surrounded by weaker connections

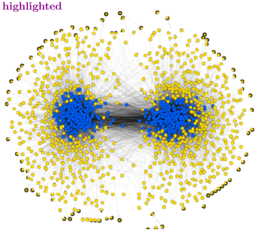
- The network could still be modular, where each module has a local core-periphery structure
- Recall the bow-tie structure of web graph is actually this pattern
- Cores can be identified visually ▷ Small networks
- It can be examined by whether highly centrality nodes are connected to other central nodes, i.e. centralities used as proxy for core nodes



A modular division of the classic political blogs network on the left, and the same network on the right but with core and peripheral nodes within each module highlighted



modular



core-periphery within modules

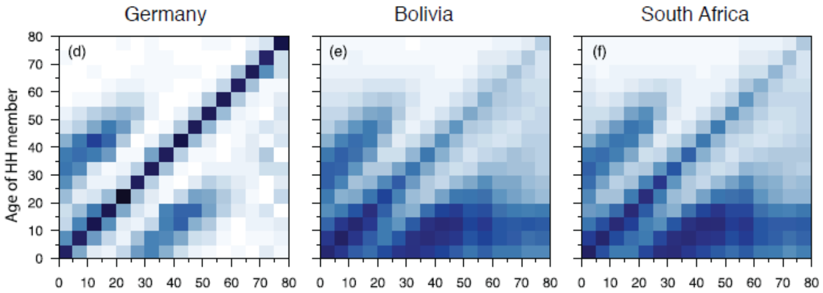
Linear Hierarchy Patterns

When nodes are ordered into a linear hierarchy (each node has a level)

Nodes at level k tend to connect only to those at levels $k \pm \Delta$

An ordered pattern can be see in such graphs

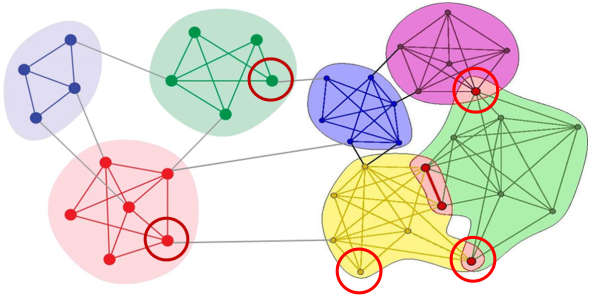
source: Prem, Cook, & Jit (2017) Profecting social contact matrices in 152 countries using surveys and demographic data



Social interactions among people are ordered with respect to people's ages

Community Structure in Networks

- Generally, clusters are non-overlapping
- But social communities may overlap
- People can belong to multiple communities

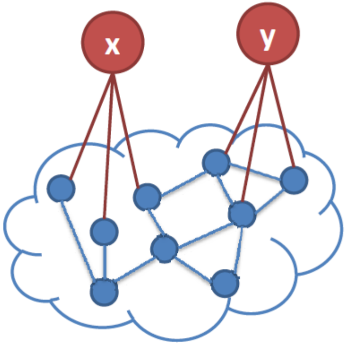


- In the same community, two nodes can reach each other in three steps
- For different communities, two nodes may have distance more than three
- For two overlapping communities, two nodes can reach each other by at most six steps

Community Structure in Networks

Dependence on number of friends and connectivity of friends

- x and y both have 3 friends in a group
- x's friends are independent
- y's friends form a clique
- Question: Who is more likely to join the group?



- Information argument [Granovetter 1973]: Unconnected friends give independent support.
- Social capital argument [Coleman 1988]: Safety/trust advantage in having friends who know each other.
- In LiveJournal, community joining probability increases with more connections among friends in the group.

Applications of Community Detection

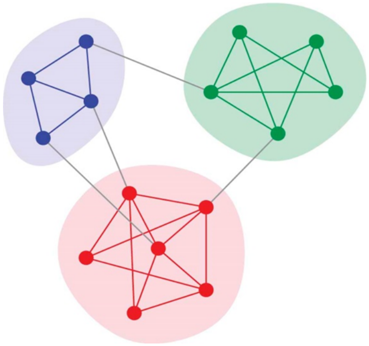
Community detection in online social networks has numerous applications

- Targeted Advertising to specific interest groups or communities
- Communities are used to recommend products or services that are popular within a user's network (e.g., Netflix's recommendation engine uses community structure to suggest shows).
- Identify nfluential communities that drive political opinion or organize collective actions
- Communities can also help detect anomalous behavior in networks (e.g., fraud rings or spam networks), as fraudulent entities often cluster in distinct communities

Quantified Community Descriptions

Connection Based Community Description

- Cohesion: More connections inside each community
- Separation: Less connections between different communities



This is a generic requirement, there are many specific interpretations of it

Connection Based Community Description

Let $A = (a)_{ij}$ be the adjacency matrix of $G = (V, E)$

Let $C_1, C_2, \dots, C_k$ be a partition of V. For $U, W \subset V$, Define

$E(U, W) = \sum_{i \in U, j \in W} a_{ij}$ (number of edges with one endpoint each in U and W)

- Condition 1 (Radicchi et al. 2004): C_i is a community in the weak sense if the number of internal connections for C_i is greater than the number of external connections to other communities $E(C_i, C_i) > E(C_i, \overline{C_i})$
- Condition 2 (Hu et al. 2008): C_i is a community in the most weak sense if the number of internal edges within C_i is greater than the number of edges connecting to any other community $E(C_i, C_i) > \max_{j \neq i} E(C_i, C_j)$
- Condition 3 : C_i is a community if each node in C_i has more connections inside than connections to outside $\forall v \in C_i, \quad E(v, C_i) > E(v, \overline{C_i})$
- Condition 4: C_i is a community if each node in C_i has more connections inside than connections to any other community
  $\forall v \in C_i, \quad E(v, C_i) > \max_{j \neq i} E(v, C_j)$

Relationships Among Connection-Based Conditions

The four conditions can be viewed as hierarchical:

- Condition 1: Internal connections $>$ External connections
- Condition 2: Internal connections $>$ External connections to any other community
- Condition 3: Each node's internal connections $>$ External connections
- Condition 4: Each node's internal connections $>$ Connections to any other community

$$\begin{tikzcd}
(3) \arrow[r] \arrow[d] & (4) \arrow[d] \\
(1) \arrow[r] & (2) \\
\text{Weak sense} & \text{Most weak sense}
\end{tikzcd}$$

Metrics for Subset Connectedness

Vertex Subset or Group connectedness refers to the extent to which members of a group are interconnected. Various metrics can quantify connectedness in networks:

- Density: The ratio of actual connections within the group to the maximum possible number of connections

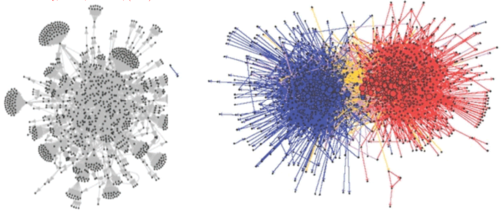
  Density $= \frac{2E_i}{|V_i|(|V_i|-1)}$ E_i is the number of edges in the subset V_i
- Average Degree: Average connections per node within group
- Clustering Coefficient: Measures the degree to which nodes tend to cluster together, forming triangles.

Quantifying Cluster Quality

Broadly, we want dense (more intra-cluster edges) and well-separated (few inter-cluster edges) clusters

- Let $G = (V, E)$, $|V| = n$, and C a subset of nodes, $|C| = n_c$
- $G[C]$: the subgraph induced by C
- $G[C, \overline{C}]$: bipartite graph between C and $\overline{C}$
- $m_c = |E(G[C])|$ number of edges inside C
- $f_c = |\{(u, v) \in E : u \in C, v \notin C\}|$ number of frontier edges

source: Ulicny, Kokar, Matheus, (2010)


(a) Malaysian Sopo blogosphere
(b) US political blogosphere (Adamic & Glance, 2005[4])

A visualization of Malysia and US blogospheres (nodes are blogs and edges are links to blogs). Left reveals importance/credibility/popularity of blogs, while the right visual (two dense clusters with little interaction with the other cluster) shows that bloggers are more likely to link to bloggers with the same party affiliations

Quantifying Cluster Quality

Goodness of a cluster

- Intra-cluster density of C aka internal density $\delta_{int}(C) = {}^{m_c}/\binom{n_c}{2}$
- Inter-cluster density of C aka cut ratio $\delta_{ext}(C) = {}^{f_c}/_{n_c(n-n_c)}$
- Conductance of C, fraction of total edge volume pointing outside C $conductance(C) = {}^{f_c}/_{2m_c+f_c}$

Quantifying Clustering Quality

All above were goodness of a community, for the whole network, compute their weighted average

$metric(G) = \sum_{C \in communities(G)} \frac{n_C}{n} \times metric(C)$

Modularity of Communities

Modularity, a quality measure of a community structure is based on the strength of a given vertex partition – It compares the density of edges within communities to the expected density in a random graph

Let A be adjacency matrix of $G = (V, E)$ and $C_1, C_2, \ldots, C_k$ be a partition of V

$$Q = \frac{1}{2|E|} \sum_{i,j \in V} \left(A_{ij} - \frac{k_i k_j}{2|E|} \right) \delta(c_i, c_j)$$

- k_i and k_j are the degrees of nodes i and j
- $\delta(c_i, c_j)$ is 1 if nodes i and j are in the same community

If an edge were distributed at random, it has endpoint i with probability $\frac{k_i}{2|E|}$ and endpoint j with probability $\frac{k_j}{2|E|}$

Hence, it lies between (i, j) with probability $\frac{k_i}{2|E|} \cdot \frac{k_j}{2|E|}$

Modularity Function

Let A be adjacency matrix of $G = (V, E)$ and $C_1, C_2, \ldots, C_k$ be a partition of V

$$Q = \frac{1}{2|E|} \sum_{i,j \in V} \left[A_{ij} - \frac{k_i k_j}{2|E|} \right] \delta_{C_i, C_j} \; = \; \frac{1}{2|E|} \sum_{c_i = c_j} \left[A_{ij} - \frac{k_i k_j}{2|E|} \right]$$

$$= \frac{1}{2|E|} \sum_{c_i} \left[2|E_{in}^{c_i}| - \frac{\left(2|E_{in}^{c_i}| + |E_{out}^{c_i}|\right)^2}{2|E|} \right]$$

$$= \sum_{c_i} \left[\frac{|E_{in}^{c_i}|}{|E|} - \left(\frac{2|E_{in}^{c_i}| + |E_{out}^{c_i}|}{2|E|} \right)^2 \right]$$

Equivalently, $$Q = \sum_{s=1}^{k} \left[\frac{E(V_s, V_s)}{E(V, V)} - \left(\frac{E(V_s, V_s) + E(V_s, \bar{V_s})}{E(V, V)} \right)^2 \right]$$

where $E(U, W) = \sum_{i \in U, j \in W} A_{ij}$

Conductance of Graphs

Conductance of Graphs is a measure used to evaluate the quality of graph partitions. It helps in understanding how easily information can flow between different parts of the network.
Key characteristics:

- Conductance is defined as the ratio of the number of edges that go between two communities (partitions) to the number of edges that are entirely within the communities.
- High conductance means that the communities are well-connected to the rest of the network, making it easier for influence or information to spread across communities.
- Low conductance indicates that there is limited interaction between communities, which can restrict the spread of influence.

Conductance is an important concept in community detection and influence maximization, especially when determining how influence can cross boundaries between communities.

Community Detection Algorithms

Community Detection Algorithms

We want to identify groups (or communities) of nodes in a network such that nodes within the same group are more densely connected to each other than to nodes outside the group.

- Networks (graphs) are ubiquitous: social, biological, information, infrastructure systems.
- Community detection helps reveal latent structure, functional modules, or clusters in such networks.
- Algorithms vary in:
  - Optimization objective: modularity, likelihood, label agreement.
  - blueApproach: greedy, hierarchical, spectral, probabilistic.
  - Scalability and resolution: efficiency on large graphs, ability to detect fine-grained or coarse communities.

Understanding community structure aids in compression, recommendation, epidemic modeling, fraud detection, and more.

Graph Partitioning and Community Detection

Community detection aims to divide a graph into subgraphs (communities) with dense intra-group connections and sparse inter-group connections.

Goal

Find partitions $C_1, C_2, \ldots, C_k$ of the node set V such that edges are denser within communities than between them.

Two Main Approaches:

- Optimization-based: Maximize an objective like modularity.
- Clustering-based: Use similarity metrics to group nodes.

Examples of Optimization-based Methods:

- LP Relaxation: Relax integer partitioning into a continuous linear program.
- Modularity Optimization: Find a partition maximizing the modularity function Q:

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

where A is the adjacency matrix, k_i is the degree of node i, and c_i is its community.

Clustering-Based Community Detection

Clustering methods group nodes using similarity derived from adjacency or embedding.

- *k*-Means Clustering:
  - Assigns nodes to k groups by minimizing intra-cluster variance.
  - Often applied on graph embeddings.
- Agglomerative Clustering:
  - Each node starts in its own cluster; merge clusters based on similarity.
- Hierarchical Clustering:
  - Builds a tree (dendrogram) of nested communities.
  - Can be agglomerative (bottom-up) or divisive (top-down).
- Spectral Clustering:
  - Uses eigenvectors of the graph Laplacian to embed nodes.
  - Clustering is then applied in the spectral space.
  - Low eigenvalues of the Laplacian correspond to smooth variations across well-connected nodes—ideal for detecting communities.

LP Relaxation for Community Detection: Intuition

Community detection is often formulated as a discrete optimization problem (e.g., minimize cut, maximize modularity). These are NP-hard in general.

Idea

Relax the discrete constraints (like binary variables indicating cluster membership) into continuous variables and solve using Linear Programming.

- Encode community assignment using indicator variables $x_{i,c} \in \{0, 1\}$, where $x_{i,c} = 1$ if node i is in community c.
- Constraints such as: each node belongs to exactly one community.
- Objective encodes edge-based cost: e.g., minimize inter-cluster edge weights.
- Relax $x_{i,c} \in [0, 1]$ to make problem solvable via LP.

Goal: Efficient approximation of optimal community structures.

LP Relaxation for Community Detection: Formulation

Consider the Min-k-Cut problem: partition graph $G = (V, E)$ into k communities minimizing total edge weight across communities.

Let $x_{ij} = 1$ if nodes i and j are in the same community, else 0.

Integer Program (IP):

$$\min \sum_{(i,j)\in E} w_{ij}(1 - x_{ij}) \quad \text{(penalize inter-community edges)}$$

$$\text{subject to:} \quad x_{ij} \in \{0,1\} \quad \forall i,j$$

$$x_{ij} + x_{jk} - x_{ik} \leq 1 \quad \forall i,j,k \quad \text{(transitivity constraint)}$$

LP Relaxation: Replace $x_{ij} \in \{0,1\}$ with $x_{ij} \in [0,1]$

- LP now solvable in polynomial time
- Use rounding or spectral post-processing to extract communities

LP Relaxation: Interpretation and Limitations

Interpretation of LP Solution

- The relaxed solution assigns “soft” similarity scores between nodes.
- x_{ij} close to 1 $\Rightarrow$ nodes i and j likely in same community.
- Can view x_{ij} as a similarity matrix, usable for spectral clustering or rounding.

Pros

- Convex, globally optimal solutions
- Can incorporate complex constraints
- Useful for approximating NP-hard problems

Limitations

- May yield fractional results (need post-processing)
- Doesn't scale well for very large graphs
- Objective must be linear (limits expressiveness)

Newman's Modularity Optimization

Modularity Q measures the strength of division of a network into communities. High modularity means dense connections within communities and sparse connections between them.

Modularity Function

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

- A_{ij}: adjacency matrix
- k_i: degree of node i
- m: total number of edges
- $\delta(c_i, c_j) = 1$ if i and j are in the same community

Newman's Modularity Optimization

Greedy Community Merging

At each step, merge the pair of communities C_i, C_j that yields the maximum increase in modularity ΔQ:

$$\Delta Q = \left[\frac{e_{ij}}{m} - 2\frac{a_i a_j}{(2m)^2}\right]$$

where

- e_{ij}: number of edges between communities i and j
- $a_i = \sum_{k \in C_i} k_k$: total degree of community i

- Intuition: Communities with many edges between them relative to a random model merge first.
- Repeat until no merge increases modularity.

Newman's Modularity Optimization

Complexity and Limitations

- Naive implementation is $O(n^3)$ due to repeated modularity calculations.
- Works well for small to medium networks.
- Can get stuck in local maxima (greedy).

The greedy hierarchical approach is conceptually simple and provides interpretable communities, but computational improvements are needed for large networks.

Clauset-Newman-Moore Algorithm

The Clauset-Newman-Moore Algorithm is a fast greedy hierarchical modularity optimization that improves on Newman's basic approach by using efficient data structures.

- Key idea: Maintain a max-heap or priority queue of modularity gains for community pairs.
- Update modularity gain after each merge efficiently.

Modularity Gain (Same as Newman)

$$\Delta Q_{ij} = \frac{e_{ij}}{m} - 2\frac{a_i a_j}{(2m)^2}$$

Clauset-Newman-Moore Algorithm

Data Structures for Efficiency

- Sparse adjacency representation: Store only non-zero ΔQ_{ij}
- Max-heaps to quickly find pair (i, j) with max ΔQ
- After merging C_i and C_j, update ΔQ values incrementally

$$O(m \log^2 n)$$

for a sparse network with n nodes and m edges.

- Much faster than naive greedy approach.
- Can handle large sparse networks efficiently.
- Still greedy and may get stuck in local maxima.
- The CNM algorithm is widely used when a scalable, fast community detection method is needed without sacrificing too much accuracy.

Louvain Algorithm: Step 1 — Initialization

Initialization:

Assign each node $i \in V$ to its own community $C_i = \{i\}$.

- Initially, there are N communities for N nodes.
- Modularity Q for a partition is defined as:

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

where

- A_{ij} = edge weight between nodes i and j
- $k_i = \sum_j A_{ij}$ = degree of node i
- $m = \frac{1}{2} \sum_{i,j} A_{ij}$ = total edge weight in the network
- c_i = community assignment of node i
- $\delta(c_i, c_j) = 1$ if $c_i = c_j$, else 0

Starting with fine-grained communities (one node per community) allows the algorithm to explore local improvements efficiently.

Louvain Algorithm: Step 2 — Local Moving Phase

- For each node i, consider moving i from its current community C to the community C' of a neighbor.
- Compute modularity gain ΔQ when moving i to C':

$$\Delta Q = \left[\frac{\Sigma_{in} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

  where
  - Σ_{in} = sum of weights of edges inside community C'
  - Σ_{tot} = sum of weights of edges incident to nodes in C'
  - k_i = degree of node i
  - $k_{i,in}$ = sum of weights of edges from i to nodes in C'
- Move i to the community C' that yields the largest positive ΔQ.
- Repeat for all nodes until no further improvement.

Intuition

This step greedily maximizes modularity locally by relocating nodes, balancing internal density and total degree.

Louvain Algorithm: Step 3 — Community Aggregation and Iteration

- Once local moving stabilizes, build a new network:

  $$G' = (V', E')$$

  where each node in V' represents a community from previous step.

- Edge weights between nodes in G' are the sum of edge weights between corresponding communities:

  $$w_{C_i, C_j} = \sum_{u \in C_i, v \in C_j} A_{uv}$$

- Repeat Step 2 (local moving) on this aggregated network.
- Continue iterating until modularity no longer improves.

.

By aggregating communities, the Louvain algorithm uncovers hierarchical structure, detecting communities at multiple scales efficiently.

Louvain vs. CNM: Community Detection Algorithms

- Both Louvain and CNM are greedy algorithms for optimizing modularity — a common quality function for community detection.
- Clauset–Newman–Moore (CNM):
  - Hierarchical agglomerative method.
  - Starts with singleton communities and repeatedly merges pairs with the highest modularity gain.
  - Suffers from getting trapped in local optima.
- Louvain Algorithm:
  - Uses a two-phase iterative process:
    - *Local Modularity Optimization:* Move nodes to neighboring communities greedily.
    - *Community Aggregation:* Collapse communities into super-nodes and repeat.
  - Can escape poor local optima by re-optimizing after collapsing.
- Key Implementation Advantage: Louvain's repeated re-optimization over progressively coarsened graphs improves both speed and modularity quality.

Louvain vs. CNM: Performance and Scalability

Comparison between the two algorithms

Feature	Louvain	CNM
Strategy	Local greedy + Hierarchical	Agglomerative Hierarchical
Modularity Quality	Typically Higher	Often Lower
Complexity (sparse)	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log^2 n)$
Speed	Faster	Slower
Scalability	High (Millions of nodes)	Moderate (Up to 100K nodes)
Heuristics	Multi-level Coarsening	Pairwise Merging

Louvain improves upon CNM on both scalability and quality due to its multi-phase re-optimization approach.

k-Partition Problem

- Given a set of n points, $\mathcal{P} \subset \mathbb{R}^m$ and $k \in \mathbb{Z}$, number of clusters
- Assume Euclidean distance measure over $\mathcal{P}$ $\triangleright$ ℓ_p, cosine can be used
- For a subset $C_i \subseteq \mathcal{P}$, denote by $\mathbf{c}_i$ the centroid of C_i $\mathbf{c}_i := \frac{1}{|C_i|} \sum\limits_{x \in C_i} x$
- Centroid is the arithmetic mean of m-dim vectors (coordinate-wise mean)
- Goodness of a k-partition $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ is measured by

sum of squared error, $SSE(\mathcal{C}) = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mathbf{c}_i\|^2$ also called Within SSE

- Problem: Find a k-partition $\mathcal{C}^*$ of $\mathcal{P}$ with minimum SSE
- Brute force approach (try all $\binom{n}{k}$ partitions) is not feasible

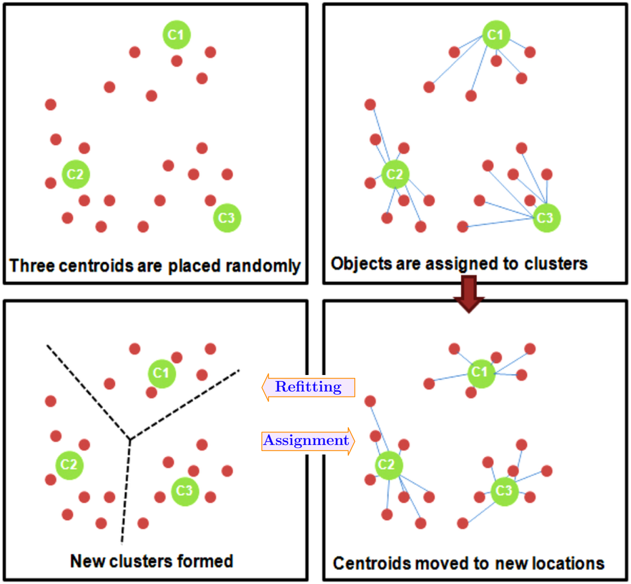
k-means Algorithm

- A basic greedy algorithm for the k-Partition problem

Algorithm : k-means algorithm $(\mathcal{P}, k)$

```
Select $k$ random points as initial centroids
                                        ▷ Alternatives of centroids can be used
while Stopping criterion is not met do                          ▷ Many choices
    Assign each point $x \in \mathcal{P}$ to the centroid closest to $x$
                                        ▷ closeness w.r.t the similarity measure
                                                             ▷ Assignment Step
    Compute the centroids of (modified) clusters
                                                               ▷ Refitting Step
```

k-means Algorithm: Illustration


C1
C2
C3
Three centroids are placed randomly
Objects are assigned to clusters
New clusters formed
Refitting
Assignment
Centroids moved to new locations

k-means Algorithm: Runtime

```
Algorithm : k-means algorithm ($(\mathcal{P}, k)$)
Select $k$ random points as initial centroids        ▷ Alternatives of centroids can be used
while Stopping criterion is not met do                ▷ Many choices
    Assign each point $x \in \mathcal{P}$ to the centroid closest to $x$
                                                      ▷ closeness w.r.t the similarity measure
    Compute the centroids of (modified) clusters
```

- Each iteration: $O(nk)$ distance computations
  - For each $x \in \mathcal{P}$ compute distances to centroids and find closest
- Recompute centroids: in total takes $O(n)$ time
- number of iterations is $t \implies$ total runtime is $O(tkn)$
- t depends on the stopping rule, generally, $t, k \ll n$
- Total time: $O(n)$ distance computations ▷ very efficient

k-Medians algorithm

- Median is less sensitive to outliers than mean
- We use *'median of clusters'* instead of centroids as clusteroids
- Let med_i be the *'median'* of a cluster C_i.
- Goodness of a k-partition $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ is measured by

$$S_{med}(\mathcal{C}) \;:=\; \sum_{i=1}^{k} \sum_{x \in C_i} \|x_i - med_i\|^2$$

Problem: Find a k-partition $\mathcal{C}^*$ of $\mathcal{P}$ with minimum $S_{med}(\cdot)$

- Various definitions of median for points in higher dimensions
- Oja Median, Simplicial Median, 1-Median, Coordinate-wise median

k-Medians algorithm

- We give a generic version of k-medians algorithms

Algorithm : k-medians algorithm $(\mathcal{P}, k)$

```
Select $k$ points as initial medians                    ▷ randomly or arbitrarily
while Stopping criterion is not met do                  ▷ many choices
    Assign each point $x \in \mathcal{P}$ to the median closest to $x$    ▷ closeness w.r.t the similarity measure
    Compute the medians of (modified) clusters          ▷ using the adopted definition of median
```

Partition Around Medoids

A pseudocode of Partition Around Medoids (PAM) is as follows:

```
Algorithm : Partition Around Medoids ($\mathcal{P}, k$)
  Select $k$ points as initial clustroids (medoids) arbitrarily
  while Stopping criterion is not met do                    ▷ many choices
      Choose a non-medoid point $p$
      Compute change in SSE with replacing a medoid $m$ with $p$
  If the change in SSE is negative, then swap $m$ with $p$
```

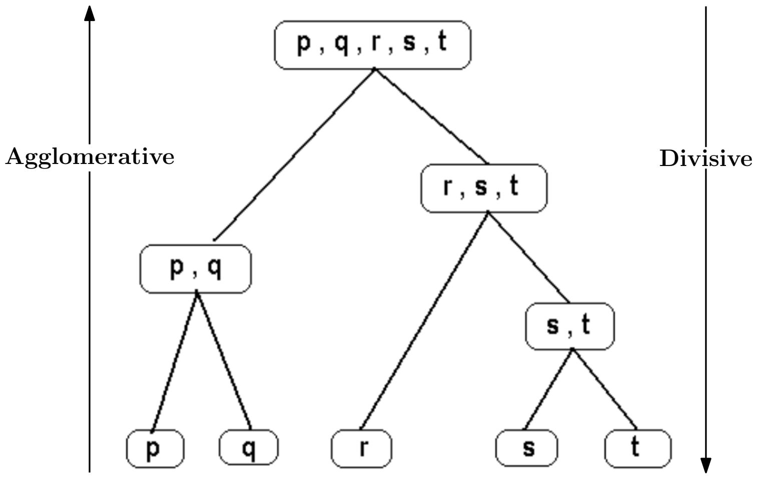
Runtime is $O(k(n-k)^2)$ in each iteration

Hierarchical Clustering

Hierarchical Clustering

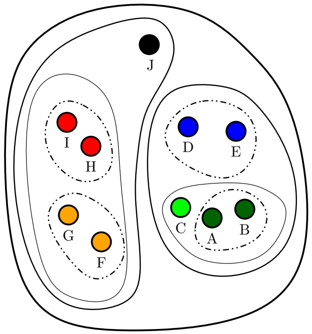
- Creates a hierarchy of clusters (multi-level partitions)
  - returns a set of nested clusters
- Generally no requirement of a fixed number of k clusters
- Hierarchical method can be
- Divisive Approach (Top-Down)
  - Initially all points are in one huge cluster
  - In every step one current cluster is split into two
  - Generates a top-down hierarchy of clusters
- Agglomerative Approach (Bottom-Up)
  - Initially each point is a cluster itself
  - In every step two clusters are merged into one
  - Generates a bottom-up hierarchy of clusters

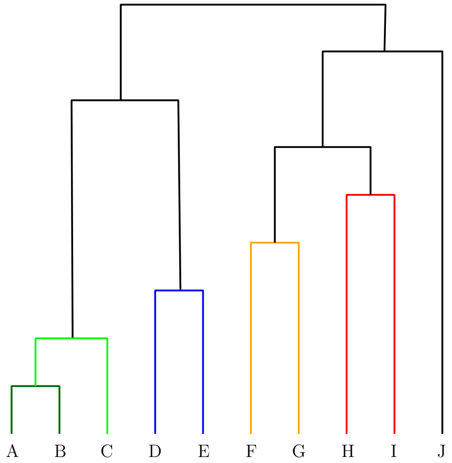
Hierarchical Clustering: Agglomerative and Divisive Approach



Hierarchical Clustering

Output of hierarchical clustering is represented by a dendrogram (a tree recording the sequence of merges or splits)





Hierarchical Clustering: Agglomerative Approach

Agglomerative Clustering

- Initially each point is a cluster itself
- In every step two *'close by'* clusters are merged into one
- Generates a bottom-up hierarchy of clusters

Key considerations:

- Representation of clusters
- Distance between clusters
- The choice of pairs of clusters to be merged
- A stopping criterion

Spectral Clustering: Intuition

- Spectral clustering uses the eigenstructure of a graph to reveal hidden clusters.
- Nodes are mapped to a low-dimensional space using eigenvectors of a matrix derived from the graph (e.g., Laplacian).
- Idea: nodes in the same community have similar positions in the spectral space.
- Analogy: Like PCA, but for graphs — preserve connectivity structure rather than variance.
- Once embedded, traditional clustering methods like k-means are applied.

Spectral embedding reveals bottlenecks and well-connected regions in the graph structure.

Spectral Clustering: Graph Laplacian

Let $G = (V, E)$ be an undirected graph with:

- Adjacency matrix $A \in \mathbb{R}^{n \times n}$
- Degree matrix D where $D_{ii} = \sum_j A_{ij}$

Unnormalized Laplacian:

$$L = D - A$$

Normalized Laplacians:

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}$$

$$L_{\text{rw}} = D^{-1} L = I - D^{-1} A$$

- These Laplacians capture smoothness and flow in the graph.
- Eigenvectors of L encode important structural info.

Spectral Clustering: Algorithm

Input: Graph G (or similarity matrix), number of clusters k

Steps:

1. Construct the (normalized) Laplacian L
2. Compute the first k eigenvectors $u_1, \dots, u_k$ of L
3. Form matrix $U \in \mathbb{R}^{n \times k}$ with rows u_i
4. Normalize rows of U (optional: especially for L_{sym})
5. Cluster rows of U using k-means $\longrightarrow$ get cluster assignments

Output: Partition of nodes into k communities based on spectral features

Why it works: Low eigenvectors encode smooth cuts with minimal edge crossing — ideal for identifying well-separated communities.

Finding Communities: Similarity between nodes

Suppose nodes of $G = (V, E)$ are labeled $v_1, \dots, v_n$. v_i is referred to as i

Let A be the adjacency matrix, i.e. $A_ij = A(i,j) = 1 \leftrightarrow (v_i, v_j) \in E$

- Jaccard Index $\triangleright$ treating A's rows as sets

$$s_J(i,j) \;=\; \frac{N(i) \cap N(j)}{N(i) \cup N(j)} \;=\; \frac{\sum_k A_{ik}A_{kj}}{\sum_k (A_{ik} + A_{kj})}$$

- Consine Similarity $\triangleright$ treating A's rows as vectors

$$s_{cos}(i,j) \;=\; \frac{\sum_k A_{ik}A_{kj}}{\sqrt{\sum_k A_{ik}^2}\sqrt{\sum_k A_{jk}^2}} \;=\; \frac{d(i,j)}{\sqrt{d_i d_j}}$$

  - $d(i,j) \;=\; |N(i) \cap N(j)|$ called co-degree of v_i and v_j

Finding Communities: Similarity between nodes

Suppose nodes of $G = (V, E)$ are labeled $v_1, \ldots, v_n$. v_i is referred to as i

Let A be the adjacency matrix, i.e. $A_ij = A(i,j) = 1 \leftrightarrow (v_i, v_j) \in E$

- Euclidean distance ▷ treating A's rows as bit-strings

  $d(i,j) \ = \ \sum_k (A_{ik} - A_{jk})^2$

- Normalized Euclidean distance

  $$d(i,j) \ = \ \frac{\sum_k (A_{ik} - A_{jk})^2}{d(i) + d(j)} \ = \ 1 - 2\frac{d(i,j)}{d(i) + d(j)}$$

  - Adjust denominator appropriately when $d(i)$'s, or $d(i,j)$'s can be 0

Finding Communities: Similarity between sets of nodes

Suppose nodes of $G = (V, E)$ are labeled $v_1, \dots, v_n$. v_i is referred to as i

Let A be the adjacency matrix, i.e. $A_{ij} = A(i, j) = 1 \leftrightarrow (v_i, v_j) \in E$

Let $X, Y \subseteq V$ and $s(i, j)$ be a similarity measure between nodes

- Single link $\triangleright$ tend to make small clusters $S_{XY} = \min_{u \in X, v \in Y} s(u, v)$
- Complete link $\triangleright$ tend to make large clusters $S_{XY} = \max_{u \in X, v \in Y} s(u, v)$
- Average link $\triangleright$ regular $S_{XY} = \dfrac{\sum_{u \in X, v \in Y} s(u, v)}{|X| \times |Y|}$

Clustering Methods Share a Common Goal

Across different techniques—k-means, agglomerative, hierarchical, spectral—the core goal is:

Partition the nodes into groups that are "similar"

- In networks, "similarity" often reflects:
  - Structural proximity (connected nodes)
  - Shared neighborhoods or patterns
  - Similar roles in the graph
- Thus, clustering becomes community detection when:
  - Similarity is derived from the graph structure.
  - The resulting clusters correspond to densely connected subgraphs.

Unifying View: Clusters Are Communities

- Community detection is a special case of clustering—just over graph data.
- Different clustering methods use different "lenses":
  - *k*-means: partitions based on node embeddings
  - Hierarchical: builds nested community structures
  - Spectral: uses smooth partitions via Laplacian
  - Agglomerative: merges local dense regions
- But they all aim to: Group together nodes that are tightly connected or structurally similar

When applied carefully, all of these methods become powerful tools for uncovering hidden communities in graphs.

Clustering as a Lens on Community Structure

- Core idea: Use clustering to discover structure in data.
- In networks, this structure is often community-based:
  - Groups with high internal connectivity
  - Fewer connections to outside nodes
- All clustering methods discussed:
  - Are unsupervised (no labels)
  - Operate on representations of nodes (adjacency, embeddings, spectral)
  - Seek to uncover latent groupings — which in graphs, are communities
- Conclusion: Clustering $\approx$ Community Detection when the similarity signal reflects structural connectivity.

Challenges in Community Detection

Community detection, while valuable, faces several challenges in real-world networks:

- Scalability: Social networks are massive, with millions or billions of nodes and edges. Algorithms must scale efficiently to handle such large data sets without excessive computational costs
- Overlapping Communities: In real life, individuals often belong to multiple communities (e.g., a user may belong to both a work-related and a family-related community). Many algorithms struggle to handle these overlapping structures effectively.
- Dynamic Networks: Social networks evolve over time. Algorithms need to adapt to changes in community structure as new connections form and existing ones disappear.

  incremental algorithms detect communities by updating the structure locally rather than recomputing the entire solution from scratch

Overlapping vs Disjoint Community Detection

Clustering based algorithms assume each node belongs to one community. In real-world networks, nodes may participate in multiple communities e.g., people belong to multiple groups, such as family, friends, work colleagues, and interest-based communities.

Classical community detection (e.g., CNM, Louvain):

$$\text{Partition } V = \bigcup_{i=1}^{k} C_i, \quad C_i \cap C_j = \emptyset, \quad \forall i \neq j$$

Each node belongs to exactly one community.

Overlapping community detection:

$$\exists\, v \in V \text{ such that } v \in C_i \text{ and } v \in C_j, \quad i \neq j$$

Nodes may belong to multiple communities simultaneously.

- Realistic modeling: Social actors participate in multiple groups.
- Classical modularity optimization unsuitable due to hard partitions.
- Requires new representations, algorithms, and evaluation metrics.

Formal Problem Statement

Given an undirected graph $G = (V, E)$, detect a family of communities:

$$\mathcal{C} = \{C_1, C_2, \ldots, C_m\}, \quad C_i \subseteq V$$

with the property that

$$\exists\, v \in V, \quad v \in C_i, v \in C_j, i \neq j$$

Goal: Find $\mathcal{C}$ such that communities are:

- *Cohesive*: nodes in each C_i densely connected.
- *Overlapping*: nodes can appear in multiple C_i.

Evaluation metrics: Overlapping modularity, Omega Index, F1 score for overlaps.

Clique Percolation Method: Intuition

- Cliques: Complete subgraphs K_k of size k.
- Communities are formed by *percolating* or linking adjacent cliques.
- Two cliques are adjacent if they share $k-1$ nodes.
- Chains of adjacent cliques form overlapping communities.

Why?

- Cliques represent tightly connected groups.
- Overlaps naturally arise as cliques share nodes.

Clique Percolation Method: Formal Definition

Define the set of all k-cliques:

$$\mathcal{K}_k = \{S \subseteq V : |S| = k, \text{ and } G[S] \text{ is complete}\}$$

Define adjacency of cliques:

$$S, T \in \mathcal{K}_k \text{ are adjacent} \iff |S \cap T| = k - 1$$

Construct clique graph $G_k = (\mathcal{K}_k, E_k)$, where edges correspond to adjacency.

Communities $\longrightarrow$ connected components in G_k:

$$\mathcal{C} = \{\text{connected components of } G_k\}$$

Each community corresponds to union of nodes in cliques in a component.

Clique Percolation Algorithm

1. Enumerate all cliques of size k in G (using e.g., Bron–Kerbosch algorithm).
2. Build clique adjacency graph G_k where two k-cliques are connected if they share $k-1$ nodes.
3. Find connected components in G_k.
4. Output communities as unions of nodes in each connected component.

Complexity:

$$O(f(k, |V|, |E|)), \text{ clique enumeration is exponential in worst case.}$$

Practical for moderate k and sparse graphs.

Clique Percolation Example

- $k = 3$ (triangles)
- Two triangles are adjacent if they share an edge.
- Overlapping community formed by chaining triangles:

$$\text{Community} = \bigcup_{\text{adjacent cliques}} \text{nodes}$$

- Nodes in multiple triangles belong to overlapping communities.

[Include diagram here showing overlapping triangles merging]

Fuzzy Clustering for Overlapping Communities

Idea: Nodes belong to multiple communities with membership degrees.

Define membership matrix:

$$U = [u_{vi}] \quad \text{where } u_{vi} \in [0,1], \quad \sum_{i=1}^{m} u_{vi} = 1 \quad \forall v \in V$$

Here, u_{vi} represents membership strength of node v in community C_i.

Goal: Minimize objective function (Fuzzy c-means):

$$J = \sum_{v=1}^{n} \sum_{i=1}^{m} u_{vi}^{q} \|x_v - c_i\|^2$$

where

- c_i = center of community i,
- $q > 1$ = fuzziness parameter,
- x_v = feature vector for node v.

Fuzzy Clustering Algorithm

1. Initialize cluster centers $\{c_i\}$ randomly.
2. Update memberships:

$$u_{vi} = \frac{1}{\sum_{j=1}^{m} \left(\frac{\|x_v - c_i\|}{\|x_v - c_j\|} \right)^{\frac{2}{q-1}}}$$

3. Update centers:

$$c_i = \frac{\sum_{v=1}^{n} u_{vi}^{q} x_v}{\sum_{v=1}^{n} u_{vi}^{q}}$$

4. Repeat until convergence.

Output: Soft assignment of nodes to multiple communities.

Interpreting Fuzzy Memberships

- Nodes with high membership in multiple communities belong to overlaps.
- Allows modeling of gradations of belonging.
- Fuzzy clustering can incorporate node features or topology-based embeddings.
- Suitable when soft boundaries exist between communities.

Limitations:

- Sensitive to initialization and parameter q.
- May produce memberships that are hard to interpret without thresholding.

Other Overlapping Community Detection Methods

- Label Propagation Variants: Allow multiple labels per node, update memberships iteratively.
- Link Clustering: Cluster edges instead of nodes, naturally overlapping nodes.
- Mixed Membership Stochastic Block Models (MMSB):

$$P(\text{edge } (u, v)) = \sum_{i,j} \pi_u(i)\pi_v(j)\theta_{ij}$$

where π_u is the membership distribution of node u.

Summary of overlapping community detection

- Overlapping community detection relaxes the disjoint membership constraint.
- CPM leverages clique adjacency for natural overlap detection but is computationally demanding.
- Fuzzy clustering provides soft memberships, accommodating uncertainty and partial participation.
- Choice depends on data nature, network size, and interpretability needs.

Community Detection in Temporal/Dynamic Networks

Static vs Temporal Community Detection

Static detection: Given a single graph snapshot

$$G = (V, E)$$

find communities $\mathcal{C} = \{C_1, \ldots, C_k\}$ with

$$\bigcup_{i=1}^{k} C_i = V, \quad C_i \cap C_j = \emptyset, \quad i \neq j.$$

Temporal detection: Given a sequence of graph snapshots

$$\mathcal{G} = \{G_1, G_2, \ldots, G_T\}, \quad G_t = (V_t, E_t),$$

detect a sequence of community partitions

$$\mathcal{C}_t = \{C_1^t, \ldots, C_{k_t}^t\},$$

that capture evolving community structure over time.

Key distinction:

$$\text{Temporal smoothness} \implies \mathcal{C}_t \approx \mathcal{C}_{t-1}$$

communities should evolve smoothly rather than jump arbitrarily.

Formal Problem Statement

Input:

$$\mathcal{G} = \{G_1, \ldots, G_T\}, \quad G_t = (V_t, E_t).$$

Optionally, $V_t = V$ fixed or varying.

Output: Sequence of community assignments

$$\mathcal{C}_t = \{C_1^t, \ldots, C_{k_t}^t\}, \quad \bigcup_i C_i^t = V_t,$$

with partitions that respect temporal consistency:

$$\text{maximize} \sum_{t=1}^{T} \left(Q(\mathcal{C}_t, G_t) - \lambda D(\mathcal{C}_t, \mathcal{C}_{t-1}) \right)$$

where

- $Q(\cdot)$: quality function (e.g. modularity),
- $D(\cdot)$: distance between partitions,
- λ: trade-off parameter for smoothness.

Goal: Detect meaningful, temporally consistent communities.

Snapshot-Based Methods

Approach:

- Detect communities independently on each snapshot G_t using static algorithms (e.g., Louvain).
- Post-process communities to match clusters across time steps (e.g., Hungarian algorithm).

Mathematically:

$$\mathcal{C}_t = \arg\max_{\mathcal{C}} Q(\mathcal{C}, G_t)$$

$$\text{Find matching } M_{t-1 \to t} : \mathcal{C}_{t-1} \to \mathcal{C}_t$$

to track evolution (splits, merges, births, deaths).

Limitations:

- Ignores temporal smoothness during detection.
- Matching post-processing can be unstable.

Evolutionary Clustering Framework

We want to incorporate temporal smoothness explicitly in clustering.

Optimization problem:

$$\mathcal{C}_t = \arg\max_{\mathcal{C}} \left(Q(\mathcal{C}, G_t) - \lambda D(\mathcal{C}, \mathcal{C}_{t-1}) \right)$$

where

$$Q(\cdot) : \text{quality function (modularity, conductance)},$$

$$D(\cdot) : \text{distance between partitions, e.g., Variation of Information}$$

Variation of Information (VI):

$$VI(\mathcal{C}, \mathcal{C}') = H(\mathcal{C}) + H(\mathcal{C}') - 2I(\mathcal{C}, \mathcal{C}')$$

with

$$H(\mathcal{C}) = -\sum_i p_i \log p_i, \quad p_i = \frac{|C_i|}{|V|}$$

and mutual information

$$I(\mathcal{C}, \mathcal{C}') = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{p_i p_j'}$$

where $p_{ij} = \frac{|C_i \cap C_j'|}{|V|}$.

Algorithmic Approaches for Evolutionary Clustering

Example: Evolutionary Spectral Clustering

$$\min_{U_t} \underbrace{\|A_t - U_t U_t^\top\|_F^2}_{\text{fit to } G_t} + \lambda \underbrace{\|U_t - U_{t-1}\|_F^2}_{\text{temporal smoothness}}$$

where

$$A_t : \text{adjacency matrix at time } t, \quad U_t \in \mathbb{R}^{n\times k} \text{ spectral embedding}$$

Iteratively solve:

$$U_t^{(r+1)} = \arg\min_{U_t} \|A_t - U_t U_t^\top\|_F^2 + \lambda\|U_t - U_{t-1}\|_F^2$$

Extract communities by clustering rows of U_t.

Dynamic Stochastic Block Models (DSBM)

Probabilistic model: At each t, nodes belong to latent communities $z_i^{(t)}$. Edges generated via

$$P(A_{ij}^{(t)} = 1 | z_i^{(t)} = a, z_j^{(t)} = b) = \theta_{ab}^{(t)}$$

Temporal dynamics:

$$P(z_i^{(t)} | z_i^{(t-1)}) = \text{Markov chain transition probabilities}$$

Inference goal:

$$\max_{\{z_i^{(t)}\}, \{\theta^{(t)}\}} P(\{A^{(t)}\}, \{z^{(t)}\}, \{\theta^{(t)}\})$$

via Expectation-Maximization or Variational Bayes.

Allows modeling birth, death, evolution of communities naturally.

Incremental and Online Algorithms

Motivation: Large-scale temporal graphs require efficient updates.

Incremental methods:

$$\mathcal{C}_t = \mathcal{C}_{t-1} + \Delta_t,$$

where Δ_t updates communities based on changes $E_t \setminus E_{t-1}$.

Examples:

- Label propagation updates with changed edges.
- Incremental Louvain by locally refining communities.

Complexity: $O(|\Delta_t|)$, significantly faster than re-clustering.

Tensor Factorization Methods for Temporal Communities

Tensor representation of temporal graphs:

A temporal network with T snapshots is encoded as a 3D tensor:

$$\mathcal{A} \in \mathbb{R}^{n \times n \times T}, \quad \mathcal{A}_{ijt} = A_{ij}^{(t)},$$

where $A_{ij}^{(t)} = 1$ if node i is linked to node j at time t, otherwise 0.

Goal: Extract evolving communities via low-rank decomposition of $\mathcal{A}$.

Canonical Polyadic (CP) Decomposition

Idea: Decompose the tensor into a sum of k rank-1 components:

$$\mathcal{A} \approx \sum_{r=1}^{k} \lambda_r \, u_r \otimes u_r \otimes v_r,$$

Component meanings:

- $u_r \in \mathbb{R}^n$: community membership vector for mode-1 (source nodes)
- $u_r \in \mathbb{R}^n$: same vector used for mode-2 (target nodes, assuming undirected)
- $v_r \in \mathbb{R}^T$: temporal activity pattern of community r
- λ_r: importance of component r

Kronecker interpretation: $\otimes$ denotes outer product, forming rank-1 tensors:

$$(u_r \otimes u_r \otimes v_r)_{ijt} = u_r(i) \cdot u_r(j) \cdot v_r(t)$$

Optimization Objective

We minimize the squared Frobenius norm of the approximation error:

$$\min_{\{u_r, v_r, \lambda_r\}_{r=1}^{k}} \left\| \mathcal{A} - \sum_{r=1}^{k} \lambda_r u_r \otimes u_r \otimes v_r \right\|_F^2$$

Solution strategy: Alternating Least Squares (ALS)

- Fix two factor matrices, optimize the third
- Repeat until convergence

Result: For each r, we get:

- $u_r(i)$ large $\Rightarrow$ node i is in community r
- $v_r(t)$ large $\Rightarrow$ community r active at time t

Interpretation and Use Cases

Key insights from CP factorization:

- Communities are *soft clusters* determined by u_r
- Each cluster evolves over time via v_r
- Overlapping is allowed — nodes can appear in multiple u_r's

Use cases:

- Detecting evolving functional groups in social or biological networks
- Modeling temporal user behavior in recommendation systems
- Identifying temporally coherent topics in co-authorship or citation networks

Evaluation Metrics for Temporal Communities

Quality metrics:

$$Q_t = Q(\mathcal{C}_t, G_t)$$

static quality at each snapshot.

Temporal smoothness:

$$S = \frac{1}{T-1} \sum_{t=2}^{T} \left(1 - \frac{D(\mathcal{C}_t, \mathcal{C}_{t-1})}{\max D}\right)$$

Combined objective:

$$J = \alpha \frac{1}{T} \sum_{t=1}^{T} Q_t + (1-\alpha) S, \quad \alpha \in [0,1]$$

Tracking events:

- *Birth, death, merge, split* of communities
- Use metrics such as normalized mutual information (NMI) over time

Summary: Temporal Community Detection

- Temporal community detection generalizes static methods by integrating time and smoothness.
- Mathematical formulation balances snapshot quality and temporal smoothness.
- Algorithms include snapshot methods, evolutionary clustering, probabilistic models, incremental updates, and tensor factorization.
- Evaluation requires metrics capturing both quality and temporal consistency.

Personalized PageRank (PPR) for Community Detection

We can also used random walk based methods like pagerank, the goal is to identify local communities around seed nodes by biased random walks

Standard PageRank: Stationary distribution π of a random walk with teleportation:

$$\pi = \alpha \frac{1}{n}\mathbf{1} + (1-\alpha)P^\top \pi,$$

where P is the transition matrix, α teleport prob.

Personalized PageRank: Restart biased towards seed node s:

$$\pi_s = \alpha e_s + (1-\alpha)P^\top \pi_s,$$

with e_s the indicator vector for seed s.

Interpretation: $\pi_s(i)$ is the probability of being at node i in a random walk restarting at s.

Computing Personalized PageRank

Iterative method (power iteration):

$$\pi_s^{(t+1)} = \alpha e_s + (1-\alpha) P^\top \pi_s^{(t)}$$

with initialization $\pi_s^{(0)} = e_s$.

Convergence: Guaranteed for $\alpha \in (0,1)$ since P is stochastic.

Computational complexity:

$$O\left(\frac{m}{\alpha}\right),$$

where m is number of edges.

Approximate methods: Use push algorithms for scalable local computations.

Using PPR for Local Community Detection

Idea: Nodes with high $\pi_s(i)$ values are strongly connected to seed s.

Thresholding: Define community

$$C_s = \{i \mid \pi_s(i) \geq \tau\},$$

where τ controls community size.

Conductance-based refinement: Select τ minimizing conductance $\phi(C_s)$

$$\phi(S) = \frac{|\partial S|}{\min(\mathrm{vol}(S), \mathrm{vol}(\bar{S}))},$$

with $|\partial S|$ edges crossing S and $\mathrm{vol}(S) = \sum_{i \in S} d_i$.

Result: Extracts well-connected local clusters overlapping with global structure.

Advantages and Extensions of PPR

- Locality: Efficient for large graphs, focuses on neighborhood of s.
- Overlapping communities: Nodes can belong to multiple PPR communities from different seeds.
- Personalization vector: Can generalize restart to multiple seeds or distributions:

  $$\pi_v = \alpha v + (1 - \alpha) P^\top \pi_v,$$

  where v is a probability vector over nodes.
- Dynamic graphs: Incremental PPR methods update π_s after graph changes without full recomputation.