Towards Graph Machine Learning

Ph.D. in Mathematics and Computer Science, University of Calabria

Francesco Cauteruccio[1], Aldo Marzullo[2]

[1]Polytechnic University of Marche, Italy
[2]University of Calabria, Italy

September 17, 2022

Course schedule

- THU 15/09 - 15:00-19:00 - Introduction to graph analytics
- FRI 16/09 - 15:00-19:00 - Introduction to machine learning and graph machine learning
- SAT 17/09 - 9:00-13:00 - Challenge

Who are we?

Francesco Cauteruccio, Ph.D.

Aldo Marzullo, Ph.D.

Who are we?

Francesco Cauteruccio received the Ph.D. in Mathematics and Computer Science from the University of Calabria in January 2018. Currently, he is a Junior Assistant Professor at the Department of Information Engineering at the Polytechnic University of Marche. His research interests include Social and Complex Network Analysis, IoT, Time Series Analysis and Biomedical Applications. He served as reviewer for more than 30 international journals. He also has been technical, scientific and program committee member of several international conferences. Among other roles, he is member of the editorial board of Frontiers in Internet of Things. He is co-owner of the unicorn of cubo 30B.

Contacts:

- `f.cauteruccio@univpm.it`
- `francescocauteruccio.info`

Who are we?

Aldo Marzullo, PhD (male, H-index = 8) was born in Paola (Cosenza, Italy) in November 1992. He graduated cum laude in Computer Science at University of Calabria (Cosenza, Italy) in September 2016, with a thesis entitled: "Retinal fundus image processing: a neural network based approach". In January 2020 he obtained his double Ph.D. at University of Calabria and University Claude Bernard Lyon 1 (Lyon, France), with a thesis entitled "Deep Learning and Graph Theory for Brain Connectivity Analysis in Multiple Sclerosis". He currently serves as Data Scientist (computer vision specialist) at Humanitas Hospital and owner of the unicorn of cubo 30B.

Contacts:

- `marzullo@mat.unical.it`

Introduction to Graph Theory and Applications

Francesco Cauteruccio, Ph.D.

Outline of today

Graphs

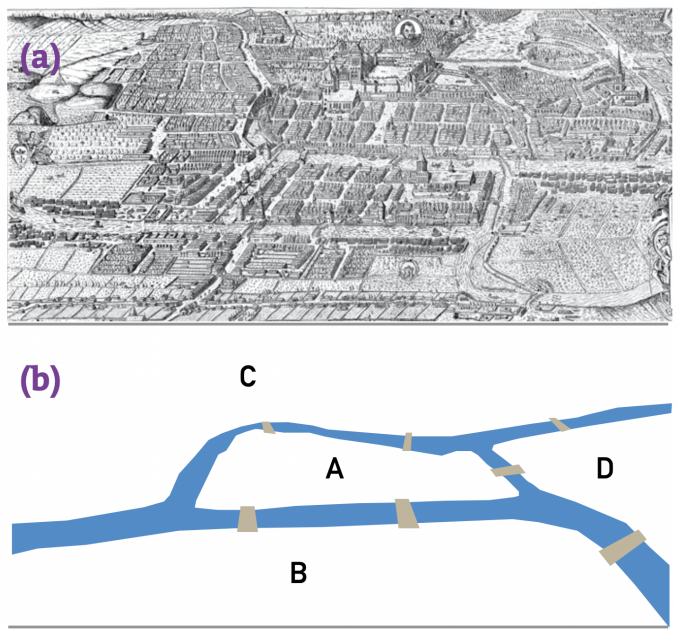
Mathematics of Graphs

Some tasks on graphs

Hands-on

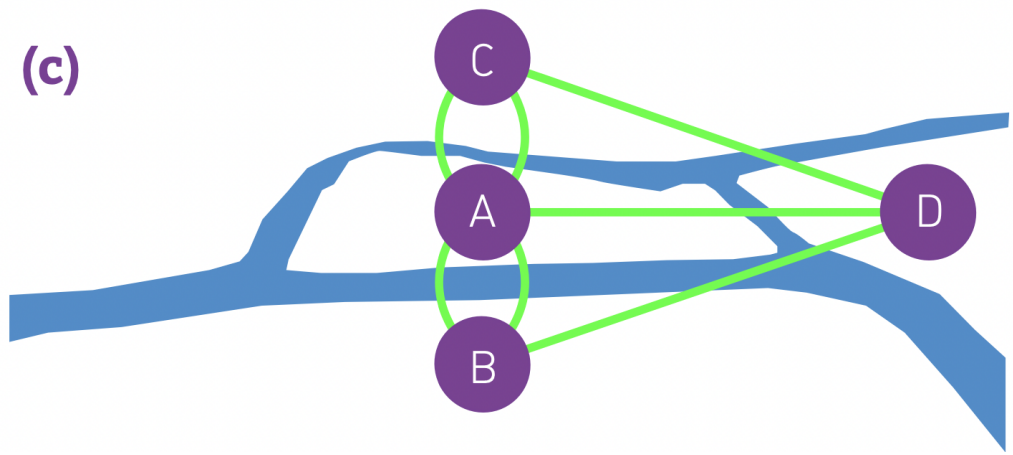
A bit of history I

- Graphs and graph theory has its roots back to 1753 in Königsberg, the capital of Eastern Prussia.
- The trade supported by its busy fleet of ships allowed city officials to build seven bridges across the river Pregel that surrounded the town. Five of these connected to the mainland the elegant island Kneiphof, caught between the two branches of the Pregel. The remaining two crossed the two branches of the river.



A bit of history II

- Problem: can one walk across all seven bridges and never cross the same one twice?
- In 1735, Leonard Euler offered a rigorous mathematical proof that such path does not exist.
- Euler represented each of the four land areas separated by the river with letters A, B, C, and D.
- He connected with lines each piece of land that had a bridge between them, and built a **graph** (and eventually proved that such path cannot exist!)



Images from [2]

Graphs are useful and ubiquitous

- Simply stated, a graph is a collection of entities and links between them.
- They represent a simple way to model systems in which the components have some sort of interactions between them.
- Graphs are studied in countless contexts
  - Mathematics (graph theory, etc)
  - Computer science (algorithms, data representation, network analysis, etc)
  - Sociology (social network analysis, etc)
  - Biology (pathways, protein interactions, etc)
  - ... and many more!

A quick note on why graph machine learning is cool

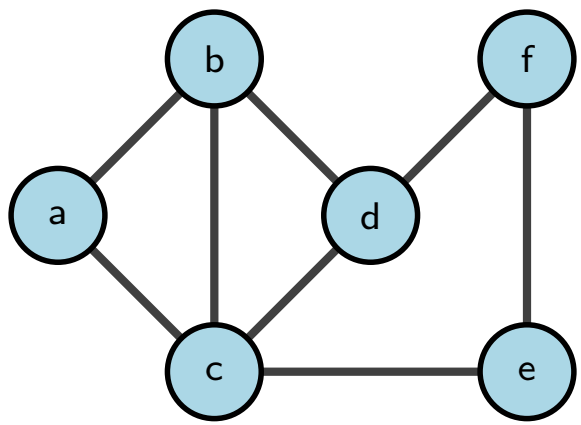
- Machine learning is a subset of artificial intelligence that aims to provide systems with the ability to learn and improve from data.
- In recent years, there has been an increasing interest in applying machine learning to graph-structured data.
- Several aims on the table
  - automatic learning of suitable representations to make predictions,
  - automatic discovering of patterns,
  - automatic understanding of complex dynamics encoded in such data.

Fundamentals

Definition (Graph)

A graph G is a triple consisting of a vertex set $V(G)$, and edge set $E(G)$, and a relation that associates with each edge two vertices (not necessarily distinct) called its endpoints.

- Vertices are also called *nodes*, while edges are called *connections* or *links*.
- We draw a graph on paper by placing each vertex at a point and representing each edge by a curve joining the locations of its endpoints.



Fundamentals

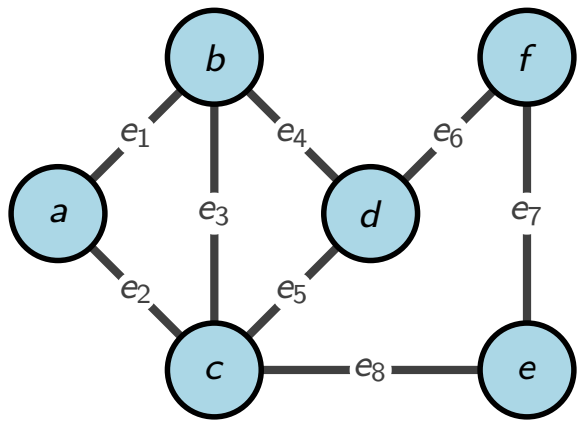
Definition (Adjacent)

When u and v are the endpoints of an edge, they are **adjacent** and are **neighbors**. It is commonly written as $u \leftrightarrow v$ for "u is adjacent to v". When it is clear from the context, we denote the edge having as endpoints u and v as $e_{u,v}$.

Definition (Order and size of a graph)

The order of a graph G is the number of its vertices, e.g., $|V(G)|$.
The size of a graph G is the number of its edges, e.g., $|E(G)|$.

$$V(G) = \{a, b, c, d, e, f\}, E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$$



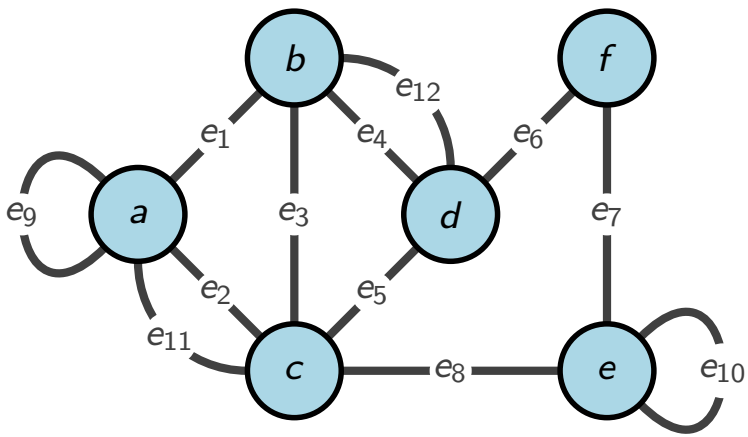
Fundamentals

Definition (Loop)

A loop (also called self-edge) is an edge whose endpoints are equal.

Definition (Multiple edges)

Multiple edges are edges having the same pair of endpoints.

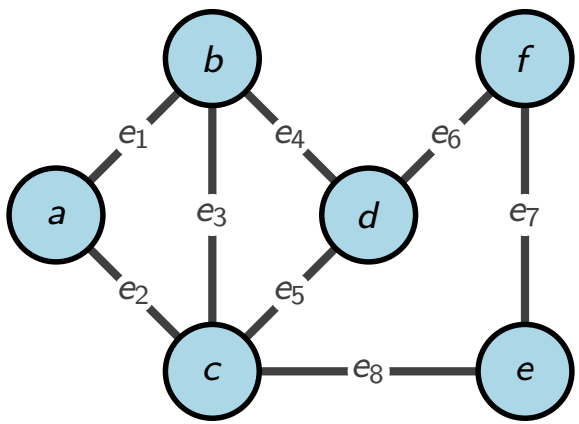


Fundamentals

Definition (Simple graph)

A simple graph (also called *undirected graph*) G is a graph having no loops or multiple edges. A single graph is specified by its vertex set and edge set, treating the edge set as a set of unordered pairs of vertices and writing $e = uv$ (or $e = \{u, v\}$) for an edge e with end points u and v.

- In set notation, $E(G) = \{\{u, v\} : u \in V(G), v \in V(G), u \neq v\}$
- A commonly used alternative notation for a graph is $G = (V, E)$ where $V = \{u_1, \ldots, u_n\}$ is the set of vertices and $E = \{e_1, \ldots, e_m\}$ is the set of edges.



Variations on directions

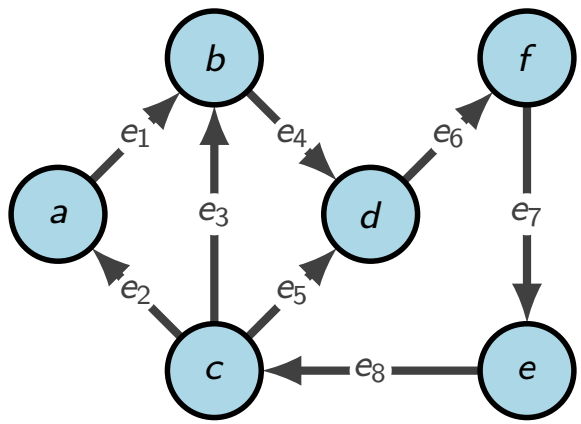
- Edges in a graph can be *undirected* or *directed*.
- Edges in a simple graph are *undirected*, i.e., $e = uv = vu$.
- In many important applications, we need to give edges a direction, i.e., $uv \neq vu$.

Variations on directions

Definition (Directed graph)

A directed simple graph G is a simple graph whose edge set is a set of ordered pairs of vertices. We write $e = (u, v)$ for an edge e starting from u and ending to v.

- Commonly, we also say that (u, v) is an edge from u to v.
- $E(G) \subseteq \{(u, v) : (u, v) \in V(G)^2, u \neq v\}$.



Variations on weights

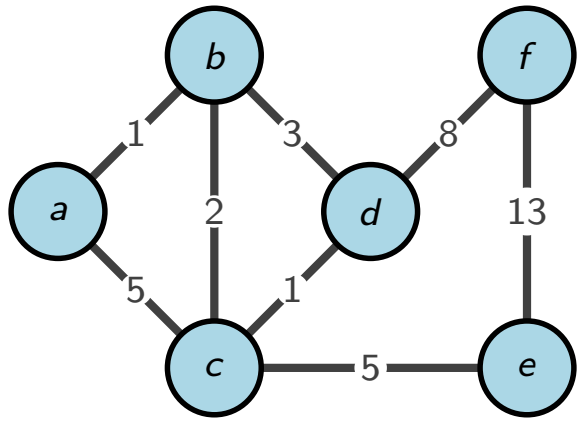
- Edges in a graph can have *weights*.
- A weight is a value associated to the edge
  - It is used to represent these situations in which edges are not simply on/off connections between vertices.
  - Imagine a map including point of interests (POIs): vertices represent POIs, edges represent possible ways of moving between two POIs, and the weight of an edge indicates the distance.

Variations on weights

Definition (Weighted graph)

A weighted graph G is a graph in which a value, called *weight*, is assigned to each edge.

- Real values are commonly used as weights, therefore we represent the weights as $W(G) : E(G) \to \mathbb{R}$

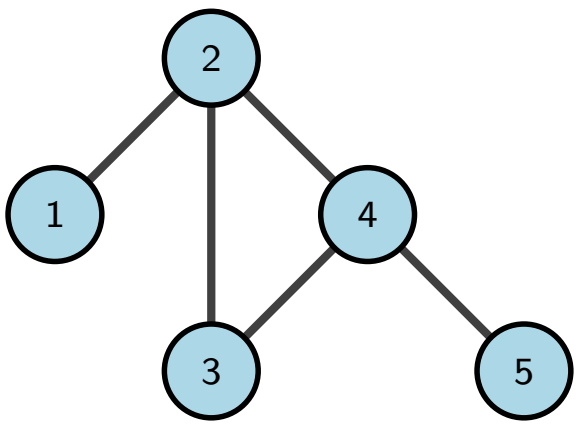


Representation of graphs

- There are a number of different ways to represents a graph
- Each way has different purposes in different contexts
- The most common representations are
  - Edge list,
  - Adjacency matrix,
  - Incidence matrix
- There isn't really a one-size-fit-all representation
  - The representation to use strongly depends on whether our graph has *special properties* and *what applications we are applying it towards*.

Edge list representation

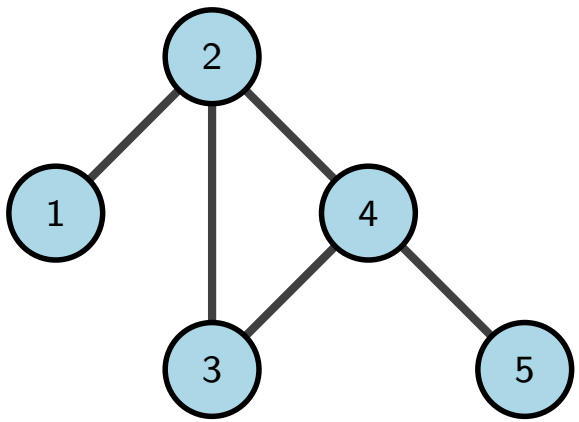
- Consider a simple graph G of order n and let us label the vertices $V(G)$ as $1 \ldots n$.
- Each label is unique, therefore we use them to refer to any vertex unambiguously.
- If we denote an edge between vertices i and j by (i, j) then the edge list representation of G is given by the elements of $E(G)$



- $E(G) = \{(1,2), (2,3), (2,4), (3,4), (4,5)\}$

Adjacency matrix representation

- A better representation for a number of purposes is the *adjacency matrix*
- The adjacency matrix of G, written as $A(G)$ is the n-by-n matrix in which entry $a_{i,j}$ is the number of edges in G with endpoints $\{i, j\}$.
- For simple graphs, the adjacency matrix has entries 0 or 1, with 0s on the diagonal.



$$A(G) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Adjacency matrix representation I

Remark

An adjacency matrix is determined by vertex ordering. For simple graphs, every adjacency matrix is **symmetric**, e.g., $a_{i,j} = a_{j,i}$ for all i, j. Instead, the adjacency matrix of a directed graph is **asymmetric**.

- Colloquially, $a_{i,j}$ indicates that nodes i and j are adjacent.
- If G is a directed graph, then $a_{i,j}$ indicates that there is an edge starting from i and ending to j.
- **The adjacency matrix representation is versatile**
- The weight w of an edge (i, j) can be represented by setting the corresponding matrix entry as $a_{i,j} = w$.

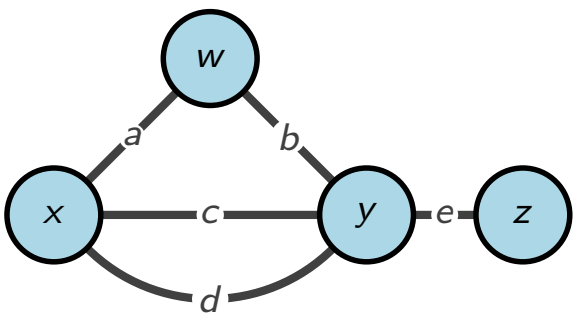
Adjacency matrix representation II

- A multi-edge is represented by setting the corresponding matrix entry $a_{i,j}$ equal to the multiplicity of the edge
  - This representation is not suitable for weighted graphs having multi-edges.
- A self-edge is represented by setting the corresponding matrix entry as $a_{i,i} = 2$
  - Why 2 and not 1? Every self-edge from i to i has two ends, both of which are connected to vertex i, therefore we count them.
  - Another way to look at this is by observing that non-self-edges appear twice in the adjacency matrix of a non-directed graph
    - In fact, in the case of directed graphs, self-edges have entries in the adjacency matrix equal to 1.

Incidence matrix

Definition (Incidence matrix)

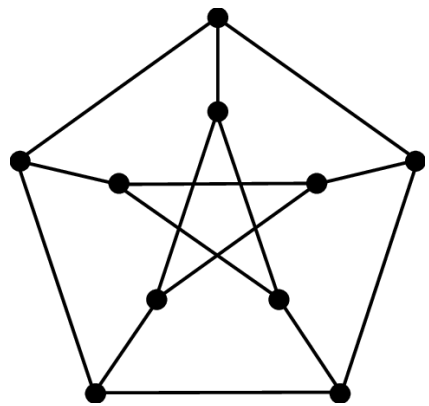
The incidence matrix $M(G)$ is the n-by-m matrix in which each entry $m_{i,j}$ is 1 if the vertex i is an endpoint of the edge j, and otherwise is 0. If vertex i is an endpoint of edge j, then i and j are **incident**.



$$M(G) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Graphs and networks

- [1] dubbed 2004 as "The year of the network".
- Network is a synonym for graph
  - We use the term network when the graph we are dealing it has, in some sense, a semantics.
  - "A network is a graph that represents a complex system" [3]
  - All networks are graphs but there are graphs that are not networks.
  - Network analysis and network science is the field that grew up around using the formalism of graphs to simplify and study more complicated systems.



Bipartite I

Definition (Bipartite graph)

A bipartite graph is a graph containing two kinds of vertices. The edges in a bipartite graph can run only between vertices of unlike types. More formally, a graph G is bipartite if $V(G)$ is the union of two disjoint (possibly empty) independent sets called **partite sets** of G, and the endpoints of each edge of $E(G)$ are not in the same partite set.

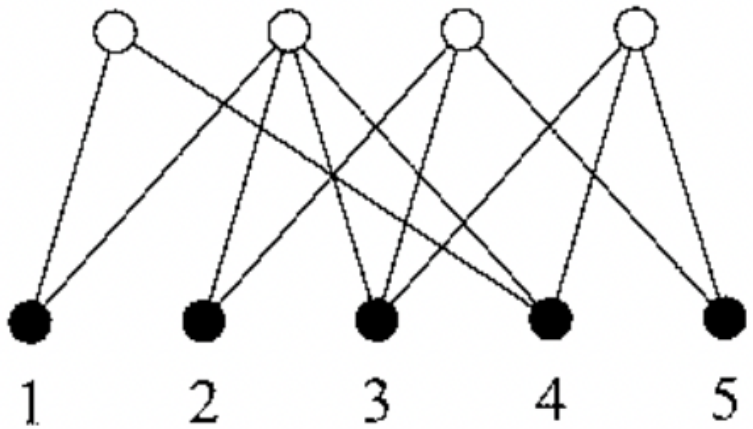


Image from [14]

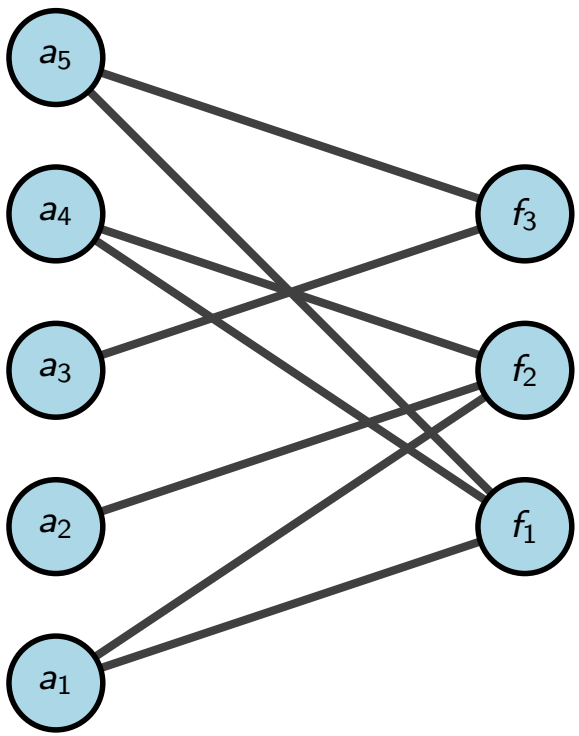
Bipartite II

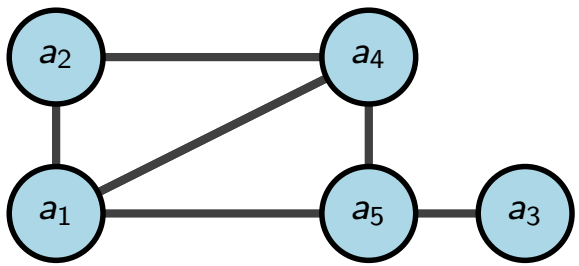
- Bipartite graphs are also called *two-modes networks*.
- It is often convenient to work with connections between vertices of the same type
- To do so, we exploit a bipartite graph to create a *one-mode projection*
  - We can perform a projection onto one of the partite sets and we infer the connections between them via the edges in the bipartite.

Bipartite III

Example

Consider a graph consisting of vertices representing actors and films, and there is an edge between an actor and a film whether the actor performed in that film. A one-mode projection onto the actors can be a graph having actors as vertices, and there is an edge between two actors if they have appeared together in a film.





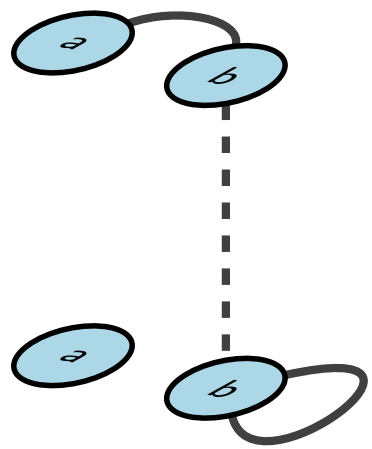
Multilayer I

Definition (Multilayer network)

A multilayer network is a pair $\mathcal{M} = (\mathcal{G}, \mathcal{E})$ where
$\mathcal{G} = \{G_\alpha : \alpha = \{1, \ldots, M\}\}$ is a family of graphs $G_\alpha = (X_\alpha, E_\alpha)$ (called **layers** of $\mathcal{M}$) and $\mathcal{E} = \{E_{\alpha\beta} \subseteq X_\alpha \times X_\beta : \alpha, \beta \in \{1, \ldots, M\}, \alpha \neq \beta\}$ is the set of interconnections between nodes of different layers G_α and G_β with $\alpha \neq \beta$.
The elements of $\mathcal{E}$ are called *crossed layers*, and the elements of each E_α are called *intralayer* connections of $\mathcal{M}$, in contrast with the elements of each $E_{\alpha\beta}$ (where $\alpha \neq \beta$) that are are called *interlayer* connections.

Multilayer II

- Multilayer networks are able to capture the complexity of systems where links have different connotations.
- Multilayer networks allow to encode more information than its single layers taken in isolation.
- There are different special cases of multilayer
  - Multiplex networks are multilayer networks in which each layer has the same set of vertices and the only possible type of interlayer connections are those in which a given node is only connected to its counterpart nodes in the rest of layers.



Hypergraph I

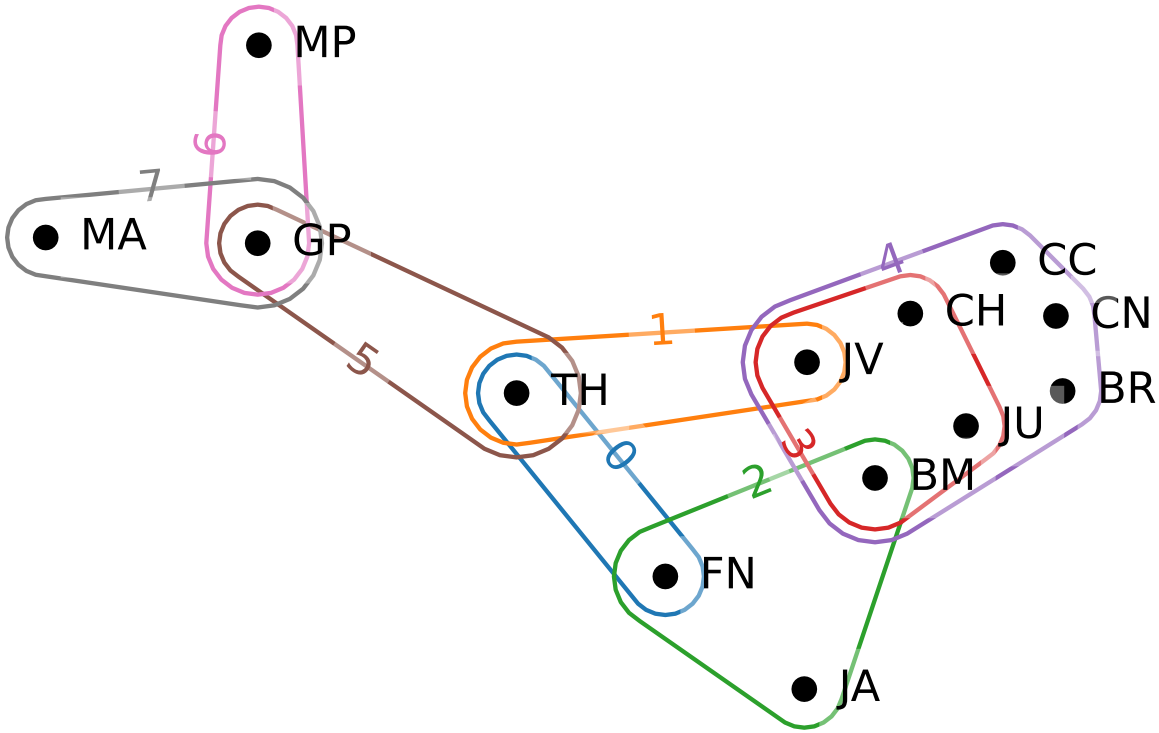
Definition (Hypergraph)

A hypergraph H is a pair $H = (V, E)$ where V is a set of vertices $V = \{v_1, \ldots, v_n\}$ and E is a family of edges $E = (e_1, \ldots, e_m), e_i \subseteq V$.

- Hypergraphs provide a mathematical model of data focused on multy-way relationships
  - Co-authorship $\rightarrow$ Collaboration
  - Friendship/following $\rightarrow$ Social groups
  - Protein interaction $\rightarrow$ Protein complex or pathway

Hypergraph II

- Hypernetwork science is a hot topic right now!



Degree I

Definition (Degree)

Given a graph G, the **degree** of a vertex u in G is the number of edges connected to it. We denote the degree of vertex u by $d(u)$.

- For an undirected graph of order n the degree can be written in terms of adjacency matrix as

$$d(u) = \sum_{j=1}^{n} a_{u,j}$$

- Note that this is still correct if there are self-edges in the graph whose corresponding entry in the adjacency matrix is equal to 2.
- For directed graphs, the degree of a vertex u can also be defined as the sum of its *indegree* and *outdegree*

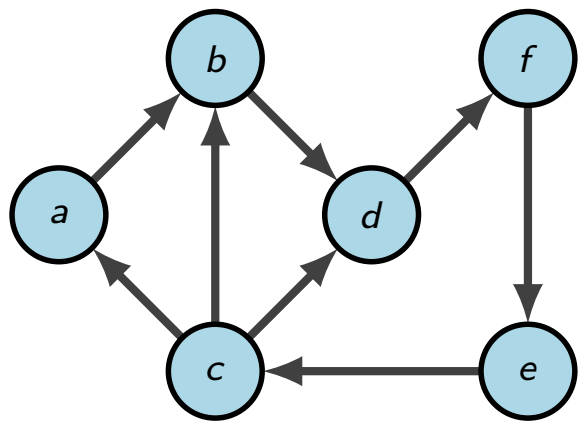
Degree II

Definition (Indegree and outdegree)

Given a directed graph G, the **indegree** of a vertex u, denoted as $d^-(u)$, is the number of edges ending in u, and the **outdegree** of a vertex u, denoted as $d^+(u)$, is the number of edges starting from u.

- For a directed graph of order n the indegree and outdegree can be written in terms of adjacency matrix as

$$d^-(u) = \sum_{i=1}^{n} a_{i,u} \qquad d^+(u) = \sum_{j=1}^{n} a_{u,j}$$

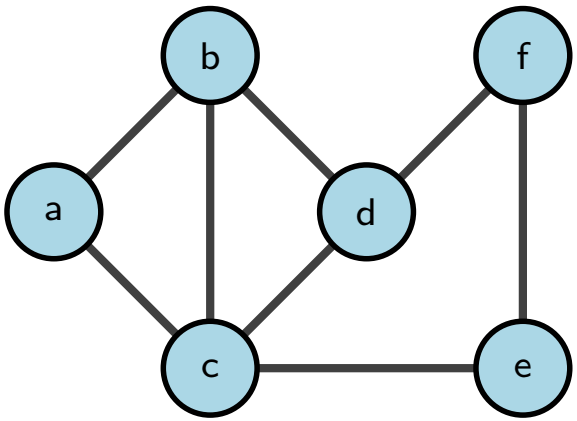


Neighborhood

Definition (Neighborhood)

Given a graph G, the neighborhood of a vertex u, denoted and defined as $neigh(u) = \{v \in V(G) : \{u, v\} \in E(G)\}$ is the set of its adjacent.

- For a directed graph, commonly $neigh(u)$ is defined to consider both vertices connected by an edge starting from or ending to u.



$$neigh(a) = \{b, c\}$$
$$neigh(e) = \{c, f\}$$
$$neigh(d) = \{b, c, e, f\}$$

Density I

- Recall that the maximum possible number of edges in a simple graph of order n is $\binom{n}{2} = \frac{1}{2}n(n-1)$.

Definition (Density or connectance)

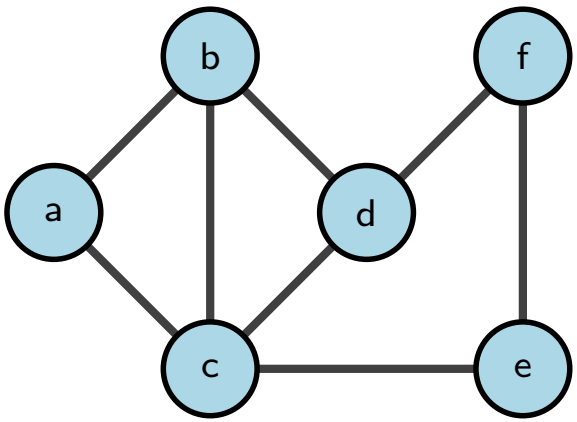
The density ρ of a simple graph is the fraction of the edges that are actually present:

$$\rho = \frac{m}{\binom{n}{2}} = \frac{2m}{n(n-1)}$$

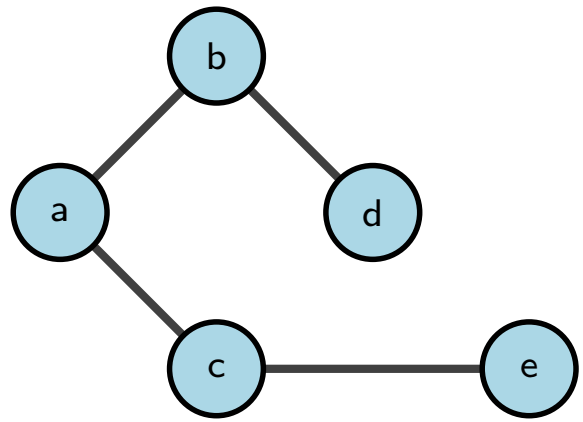
- The density lies in the range $0 \leq \rho \leq 1$
  - A graph for which ρ tends to a constant as $n \to \infty$ is said to be *dense*, that is the fraction of non-zero elements in the adjacency matrix remains constant as the graph becomes large.
  - A graph for which $\rho \to 0$ as $n \to \infty$ is said to be *sparse*, that is the fraction of non-zero elements in the adjacency matrix also tends to zero.

Density II

- The density is an important measure of the degree of connection between the entities in a graph [2, 14]



$\rho = 0.53$



$\rho = 0.4$

Walks and paths I

- Vertices in a graph can be close or distant. In order to quantify this aspect a notion of **distance** is needed.

Definition (Walk)

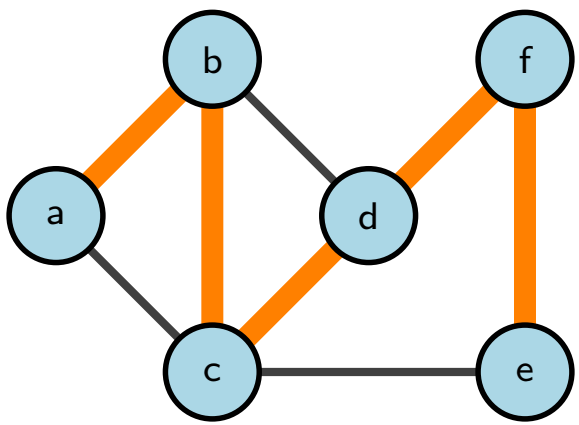
A walk is a finite or infinite sequence of edges which joins a sequences of vertices.
More formally, let G be a graph. A finite walk is a sequence of edges $(e_1, e_2, \ldots, e_{n-1})$ for which there is a sequence of vertices $(v_1, v_2, \ldots, v_n)$ such that $e_i = \{v_i, v_{i+1}\}$ for $i = 1, \ldots, n-1$.
We call the walk *closed* if $v_1 = v_n$, and it is *open* otherwise.

Walks and paths II

Definition (Path)

A path is a walk in which all vertices (and therefore also all edges) are distinct.
The **length** of a path is the number of edges it consists of.

- We can define a walk analogously by the point of view of vertices: a walk is a sequence of vertices $(v_1, \ldots, v_n)$ s.t. v_i is adjacent to v_{i+1} for $1 \leq i < n$.
- A **directed walk** is a walk whose edges are directed (therefore it is only defined for directed graphs).



Walks and paths III

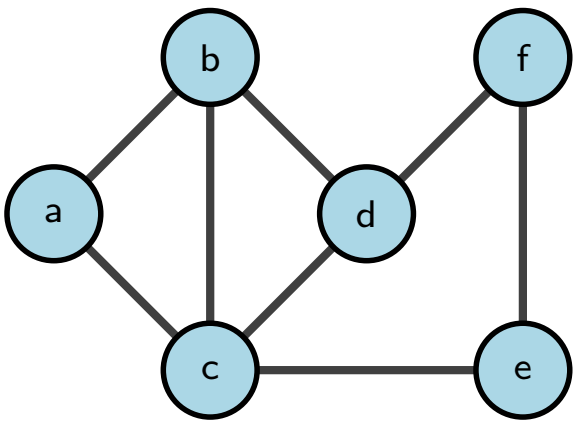
Definition (Connection between vertices)

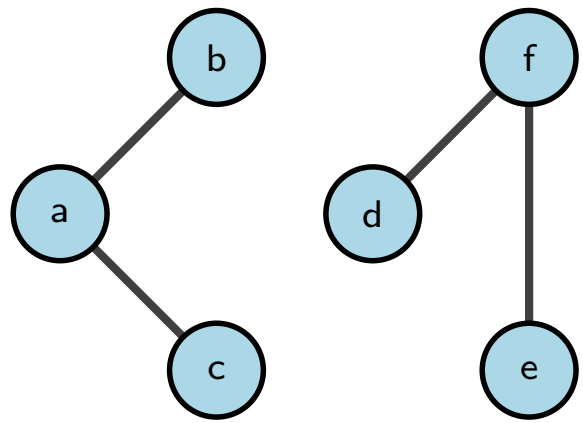
In a simple graph G, two vertices u and v are called **connected** if G contains a path from u to v, and they are **disconnected** otherwise.

Definition (Connectedness of a graph)

A graph is said to be **connected** if every pair of vertices in the graph is connected, and it is said to be **disconnected** otherwise.
A directed graph is called **weakly connected** if replacing all of its directed edges with undirected edges produces a connected (undirected) graph. It is called **strongly connected** if it contains a directed path from u to v and a directed path from v to u for every pair of vertices u, v.

Walks and paths IV

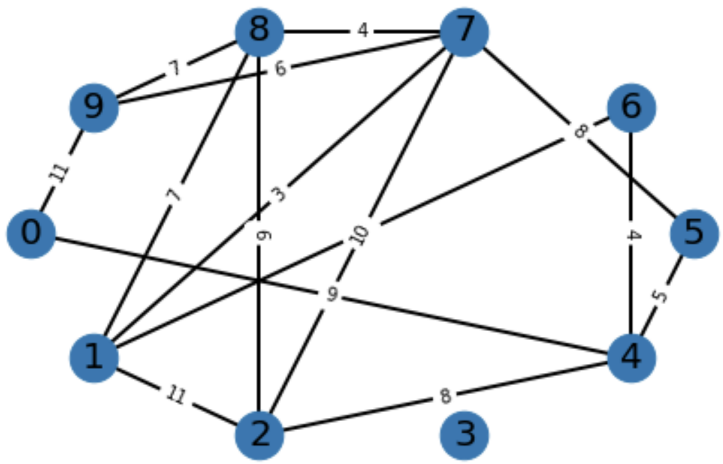




Shortest path

Definition (Shortest path)

Given a graph G and a real-valued weight function $f : E(G) \to \mathbb{R}$, the **shortest path** between two vertices u and u' of G is the path $P = (v_1, v_2, \ldots, v_n)$, where $v_1 = u$ and $v_n = u'$, that over all possible n minimizes the sum $\sum_{i=1}^{n-1} f(e_{i,i+1})$.
In case of $f : E(G) \to \{1\}$, the shortest path is the path with fewest edges.



Shortest path between 0 and 8 is $P = (0, 9, 8)$

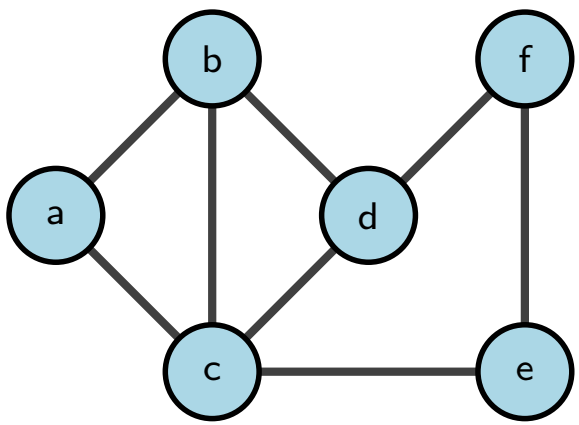
Distance and related concepts I

Definition (Distance)

The distance between two vertices u and v in a graph is the length of a shortest path connecting them. We denote it as $d(u, v)$. If two vertices are disconnected, then conventionally the distance is defined as infinite.

Definition (Eccentricity)

The eccentricity $\epsilon(u)$ of a vertex u is the greatest distance between u and any other vertex: $\epsilon(u) = max_{v \in V(G)}\ d(u, v)$



$d(a, e) = 2$

$d(a, f) = 3$

$\epsilon(d) = 2$

$\epsilon(f) = 3$

Distance and related concepts II

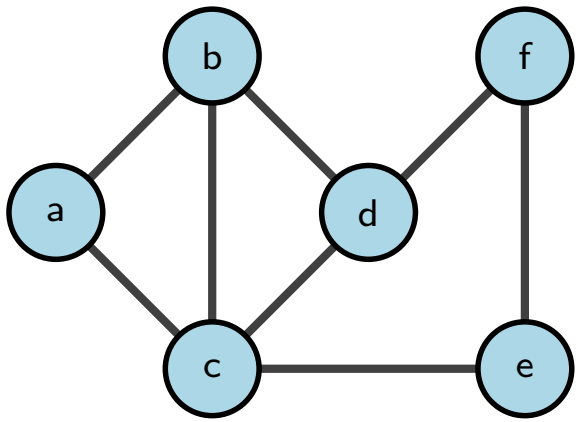
Definition (Radius)

The radius r of a graph is the minimum eccentricity of any vertex in the graph: $r = min_{u \in V(G)}\ \epsilon(u)$

Definition (Diameter)

The diameter dm of a graph is the maximum eccentricity of any vertex in the graph: $dm = max_{u \in V(G)} \epsilon(u)$

- These measures allow one to capture several aspects inherent to the analyzed networks [11, 8]



$r = 2$

$dm = 3$

Distance and related concepts III

Definition (Characteristic path length)

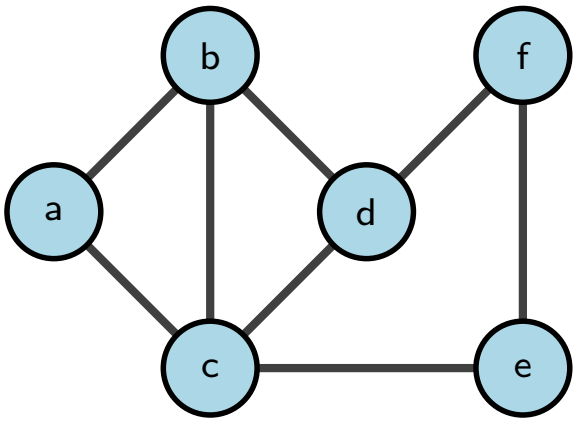
Let G be a graph of order n and let $d(u, v)$, where $u, v \in G(V)$, denote the shortest distance between u and v. Assume that $d(u, v) = 0$ if u and v are disconnected. Then, the **characteristich path length** l_G is

$$l_G = \frac{1}{n(n-1)} \sum_{u,v \in V(G), u \neq v} d(u, v)$$

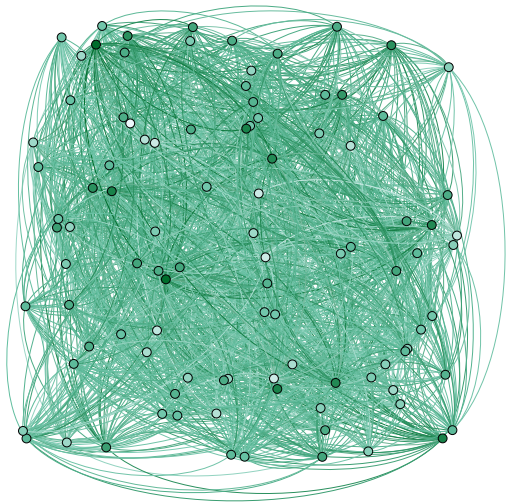
.

- Also called *average path length*
- It is considered a robust measure of network topology [2]
  - It is a measure of efficiency of information transport on a network

Distance and related concepts IV



$l_G = 1.53$



A Erdős-Rényi random graph of order 100 and $1/3$ edge probability, $l_G = 1.66$

Efficiency, and global and local efficiency I

Definition (Efficiency [10])

The efficiency of a pair of vertices u and v in a graph G is the multiplicative inverse of the shortest distance between the vertices, that is $ef(u, v) = \frac{1}{d(u,v)}$.

Definition (Global efficiency [10])

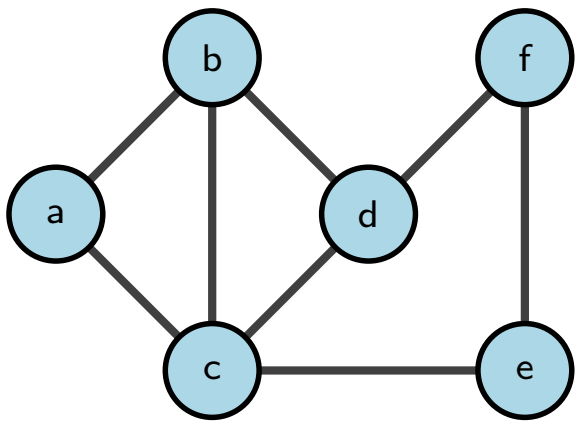
The average global efficiency of a graph G of order n is the average efficiency of all pairs of vertices, that is
$ef_{glob}(G) = \frac{1}{n(n-1)} \sum_{u,v \in V(G), u \neq v} ef(u, v)$

Definition (Local efficiency [10])

The local efficiency of a graph G of order n w.r.t. a vertex u is the average global efficiency of the local subgraph consisting only of nodes in $neigh(u)$.

Efficiency, and global and local efficiency II

- The key distinction here is that characteristic path length measures efficiency in a system where *only one packet of information is being moved through the graph*, while the global efficiency measures the efficiency of *parallel communication*.



$$ef(a, f) = 0.33$$
$$ef(a, e) = 0.5$$
$$ef_{glob}(G) = 0.75$$

How to compute paths

- We given several definitions for paths and types of paths.
- How does one compute them?
- The **shortest path problem** is the problem of finding a path between two vertices in a graph s.t. the sum of the weights of the edges it consists of is minimized.
- Several variants
  - Single-source shortest path problem (from a vertex to all others)
  - Single-destination shortest path problem (from all vertices to a specific one)
  - Single-pair shortest path problem (for a single pair of vertices)
  - All-pairs shortest path problem (for all pairs of vertices)
- And several algorithms
  - Dijkstra's algorithm (single-source w/ non-negative weights)
  - Bellman-Ford algorithm (single-source w/ negative weights)
  - Floyd-Warshall algorithm (all pairs)
  - and others [5, 12, 17]

Segregation metrics

- Segregation metrics help to understand the presence of groups and quantify the degree of cohesiveness of the vertices within a graph.
- Concepts such like **network homophily** help Identifying patterns in which ties are more likely to exist between nodes similar to each other.
- Such aspects become crucial when networks represent complex systems resembling real phenomena.

Triadic closure

- In social network theory, the **triadic closure** is the property among three nodes u, v and w, that if the links uv and uw exist, there is a **tendency** for the new link vw to be formed [15].
- Practically speaking, it is the tendency for people who share connection in a social network to become connected.
- How can we measure the prevalence of triadic closure in a network?

Local clustering coefficient I

Definition (Local clustering coefficient)

Given a graph G and a vertex u, the local clustering coefficient $LCC_G(u)$ is the fraction of pairs of the vertex's neighbors that are connected with each other. Formally

$$LCC_G(u) = \frac{2|\{e_{v,w} : v, w \in G(V), e_{v,w} \in E(V)\}|}{d(u)(d(u)-1)}$$

For a directed graph G', the $LCC_{G'}(u)$ does not consider the multiplicative factor of 2.

- We assume that for a vertex u with $d(u) < 2$, $LCC_G(u) = 0$.

Local clustering coefficient II

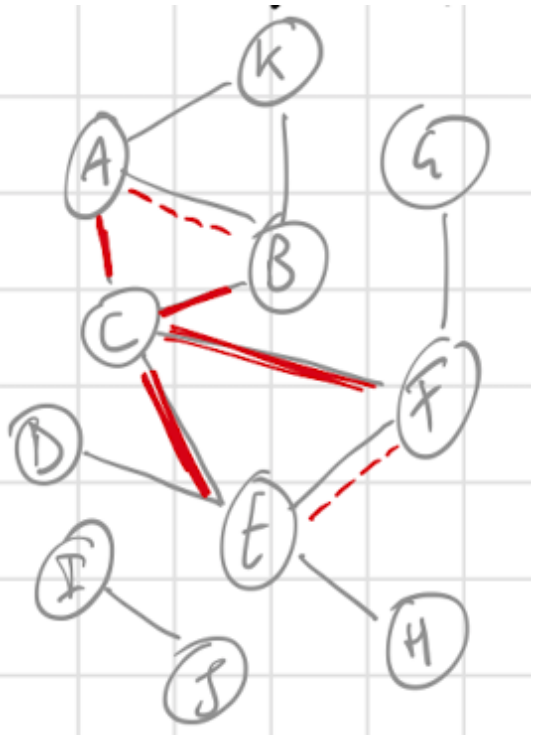
- EXAMPLE

# of C's friends = $d_C = 4$ (solid red)

# of pairs of C's friends = $\frac{d_C(d_C-1)}{2} = 6$

# of pairs of C's friends with one friend = 2 (dashed red)

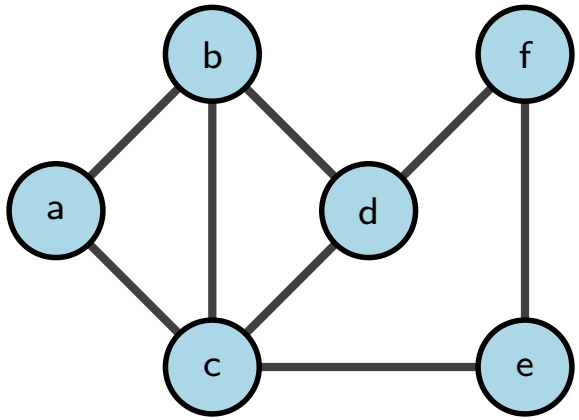
Local Clustering Coefficient of $C = \frac{2}{6} = \frac{1}{3}$



Global clustering coefficient I

- The LCC gives an indication of the *embeddedness* of single vertices.
- The global version of the clustering coefficient, instead, was designed to give an overall indication of the clustering in the network [16].
- There are two approaches to compute the global clustering coefficient GCC_G of a graph G:
  - Approach (1): computing the average $LCC_G(u)$ for all $u \in V(G)$,
  - Approach (2): computing the percentage of *open triads* that are triangles in a graph. This is also called **transitivity** of a graph and can be defined in different ways
    - $T(G) = \frac{\#\text{ of closed triplets}}{\#\text{ of all triplets}}$
    - $T(G) = \frac{3 \times \#\text{ of triangles}}{\#\text{ of all triplets}}$
    - $T(G) = \frac{\sum_{u,v,w} a_{u,v} a_{v,w} a_{w,u}}{\sum_u d(u)}$ (with $T(G) = 0$ when $\sum_u d(u) = 0$)

Global clustering coefficient II



$$T(G) = 0.38$$

Modularity I

- Modularity is a measure of the structure of graphs.
- It measures the strength of division of a network into modules (also called communities, groups or clusters).
- Graphs having high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules.
- It is used in tasks such as **community detection** and, in general, in tasks in which we need to intra-group cohesiveness.

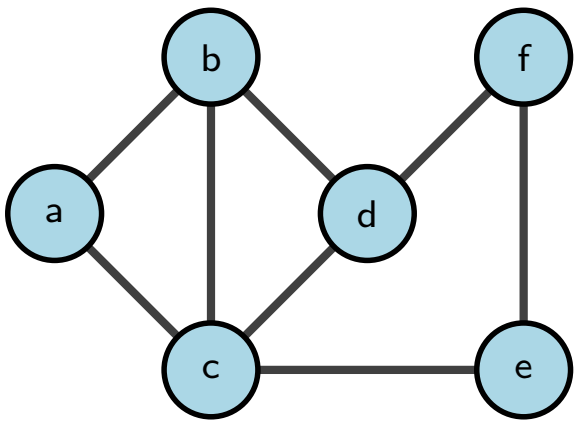
Modularity II

- More precisely, modularity is the fraction of the edges that fall within the given groups minus the expected fraction if edges were distributed at random.
- Considers a graph G of order n and size m s.t. it can be partitioned into two communities using a membership variable s.
  - If a vertex u belongs to community 1, $s_u = 1$
  - of, if u belongs to community 2, $s_u = -1$
- We define the difference between the actual number of edges between vertices u and v and the expected number of edges between them as $a_{u,v} - \frac{d(u)d(v)}{2m}$
- Then, summing over all vertex pairs gives the modularity

$$Q = \frac{1}{2m} \sum_{u,v} \left[a_{u,v} - \frac{d(u)d(v)}{2m} \right] \frac{s_u s_v + 1}{2}$$

Modularity III

- Modularity ranges between -1 and 1, with 1 meaning optimal modularity.
- However, the maximum achievable modularity for a given graph isn't necessarily 1 but possibly lower, depending on the structure.
- In practice, assignments with Q values above 0.3 are typically considered a successfully identified cluster structure [4].



- $s_a = s_b = s_e = 1$, $s_c = s_d = s_f = -1$ gives $Q = -0.13$
- $s_a = s_b = s_c = 1$, $s_d = s_e = s_f = -1$ gives $Q = 0.11$

Centralities I

- Segregation metrics help to understand the presence of groups.
- However, each vertex in a graph has its own *importance*, which is tied to the concept of **centrality**.
- *Which are the most important or central vertices in a graph?*
  - There are many possible definitions of importance, and correspondingly may **centrality measures** for graphs.
  - Each centrality measure induces a *ranking* of the vertices.
  - Identifying central vertices is the fundamental steps of many other tasks [6, 2, 14]

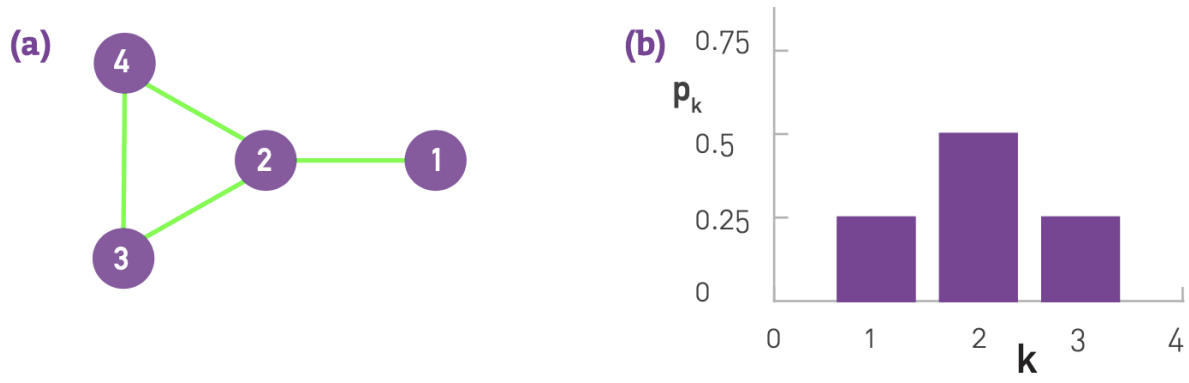
Degree centrality

- The simplest centrality measure in a network is just the degree of a vertex, that is the number of edges connected to it.
- **Assumption: important nodes have many connections**.
- Although very simple, it can be very illuminating in several contexts
  - In a social network, it is reasonable to suppose that individuals who have connections to many other might have more influence, more access to information, or more prestige than those who have fewer connections.
  - In the evaluation of scientific papers, the number of citations a paper receives from other papers (i.e., its indegree), gives a crude measure of whether the paper has been influential or not.

How to visualize centralities

- In case of small graphs, a visualization with vertices and edges, and an indication of the centrality value of the vertex is sufficient.
- In case of large graphs, it is convenient to plot the distribution of the centrality.

Plotting the degree distribution



- If we decide to depict the distribution of the degree centrality, we can exploit a histogram.
- The network in (a) has order $n = 4$
- In (b) we have $p_1 = 1/4$ (one of the four vertices has degree $d(1) = 1$), $p_2 = 1/2$ (two vertices have $d(3) = d(4) = 2$), and $p_3 = 1/4$ (as $d(2) = 3$)
- Here, $p_k = \frac{n_k}{n}$ is the normalized histogram value for degree k, where n_k is the number of vertices of degree k.

Image from [2]

Closeness centrality

- Closeness centrality measures the mean distance from a vertex to other vertices [14].
- **Assumption: important vertices are close to other nodes.**
- Suppose $d_{u,v}$ is the length of the shortest path from u to v, then the mean shortest path length from u to v, averaged over all vertices v in the graph, is

$$l_u = \frac{1}{n} \sum_v d_{u,v}$$

- The mean distance l_u is not a centrality measure since that it gives *low* values for more central vertices and high values for less central ones. *It could work* but it is commonly rewritten as its inverse, that is

$$C_u = \frac{1}{l_u} = \frac{n}{\sum d_{u,v}}$$

- For directed graphs we have two options for computing it:
  1. Consider only vertices that u can reach $C_u = \frac{|R_u|}{\sum_{v \in R_u} d_{u,v}}$ where R_u is the set of all vertices reached by u
  2. Consider only vertices that u can reach and normalize for the fraction of vertices that v can reach $C_u = \frac{|R_u|}{n-1} \frac{|R_u|}{\sum_{v \in R_u} d(u,v)}$

Betweenness centrality I

- Betweenness centrality measures the extent to which a vertex lies on paths between other vertices.
- **Assumption: important vertices connect other nodes, i.e., they act like hubs.**
- Suppose $\sigma_{v,w}$ is the number of shortest paths between vertices v and w, and suppose $\sigma_{v,w}(u)$ is the number of shortest paths between nodes v and w passing through u, then the betweenness centrality of a vertex u is

$$C_u = \sum_{v,w \in V(G)} \frac{\sigma_{v,w}(u)}{\sigma_{v,w}}$$

- Note that here we include all the shortest paths starting from and ending to u
  - In most applications we do not include them, therefore $v, w \in V(G), v \neq u, w \neq u$.

Betweenness centrality II

EXAMPLE

$$C(D) = \delta_{A,B}(D) + \delta_{A,C}(D) + \delta_{A,E} + \delta_{A,F} + \delta_{A,G} +$$
$$\delta_{B,C} + \delta_{B,E} + \delta_{B,F} + \delta_{B,G} +$$
$$\delta_{C,E} + \delta_{C,F} + \delta_{C,G} +$$
$$\delta_{E,F}(D) + \delta_{E,G}(D) +$$
$$\delta_{F,G}(D) = 9$$

⇒ node D lies on all the shortest paths between {A,B,C} and {E,F,G} and there are no alternative paths for these pairs. Hence D has the highest betweenness.

Components I

- As stated previously, a graph could be disconnected, i.e., there exists a pair of vertices that are not connected.
- In such case, there are several *pieces* of the graph that are disconnected.
- These pieces are called **components**.

Components II

Definition (Subgraph)

A subgraph of a graph G is another graph formed from a subset of the vertices and edges of G. The vertex subset must include all endpoints of the edge subset, but may also include additional vertices. More formally, a subgraph G' of G is a graph s.t. $V(G') \subseteq V(G)$

Definition (Component)

A component of a simple graph is a connected subgraph that is not part of any larger connected subgraph.

- The components of a graph can be constructed in linear time [9]
- Components are useful in many contexts and applications, both from a theoretical and practical point of view
- As we'll see later, several measures and properties are defined on connected graphs. Therefore, we'll learn how to deal with components.

Structures I

- There are several kinds of graph that are somewhat special, in the sense that their formation is often connected to model different aspects and contexts of a complex system.
- We have seen a special graph already that is the *bipartite graph*
  - It helps modeling systems in which there are two distinct sets of entities (e.g., authors and their papers)
- When dealing with complex systems, we can look for structures in order to understand whether they carry a semantics related to the context of analysis

Structures II

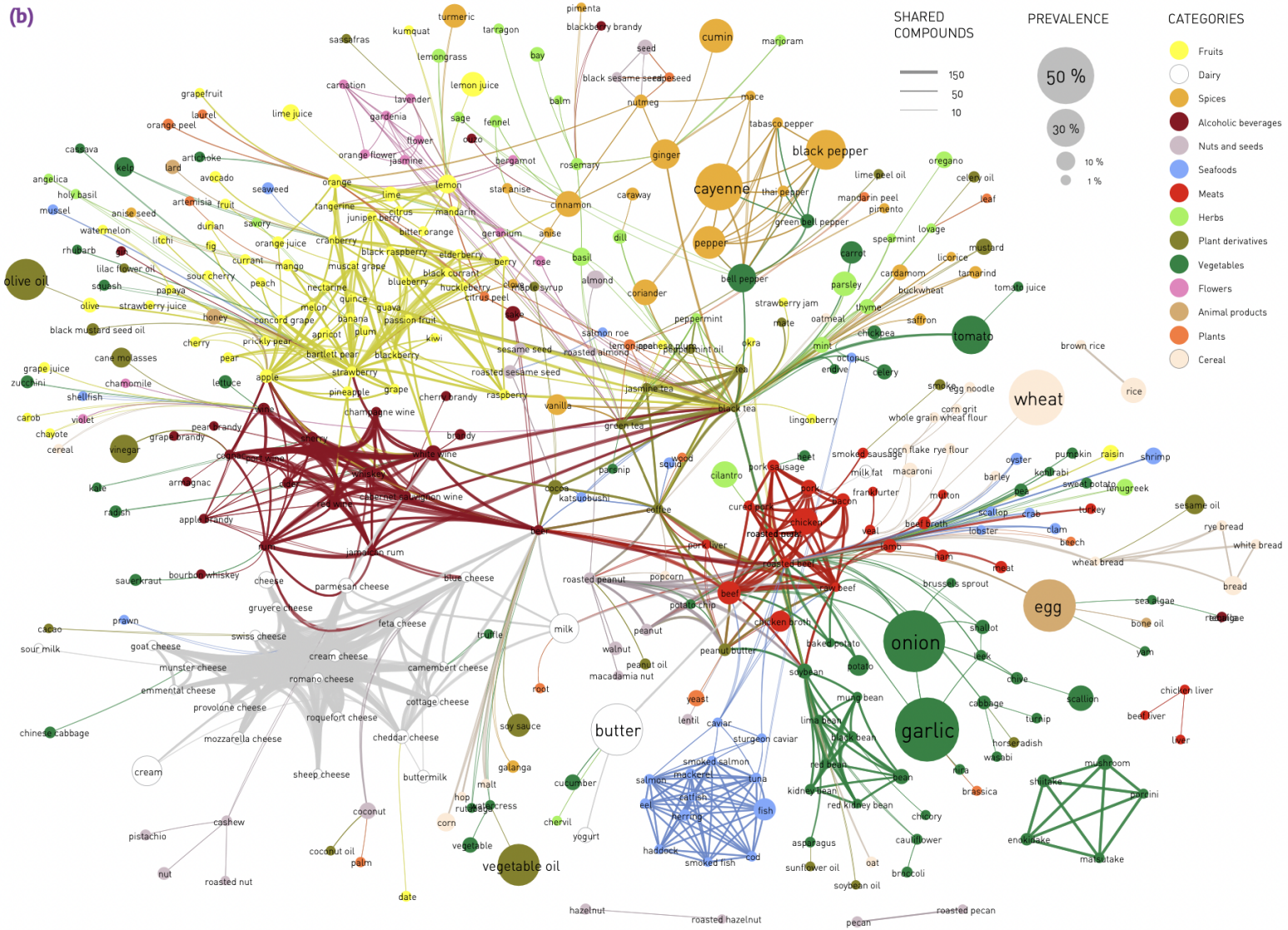


Image from [2]

Clique I

- A clique is a subset of vertices of an undirected graph s.t. every two distinct vertices in the clique are adjacent.
- The task of finding whether there is a clique of size k in a graph is NP-complete.
- Cliques are used in a variety of contexts:
  - In social networks, to model groups of people all of whom know each other.
  - In bioinformatics, to cluster genere expression or to model ecological niches.
- Also, several relaxations of the notion of clique have been proposed:
  - k-cores, k-plex, n-clan, n-club, k-truss, ... [7]

Clique II

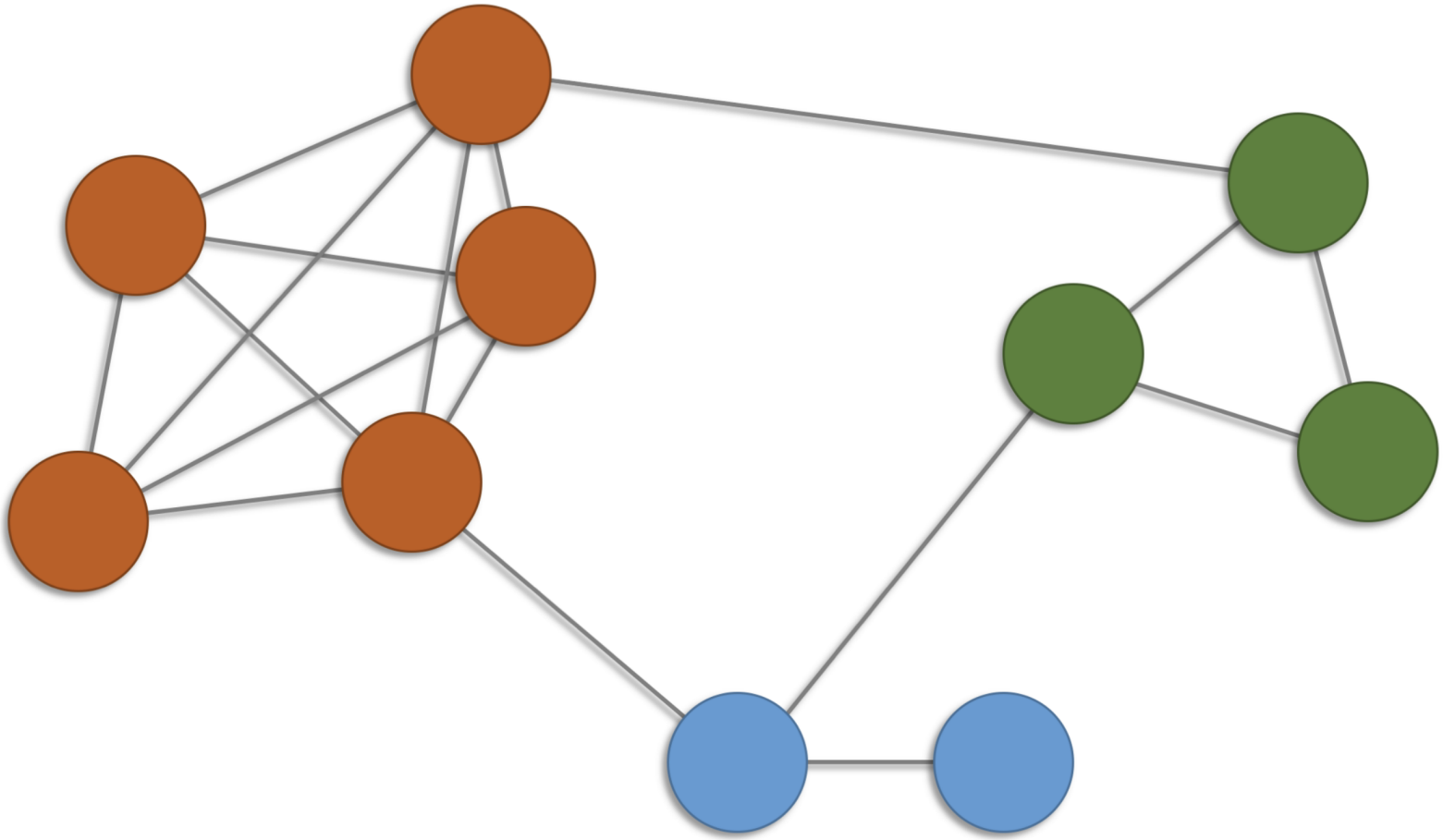


Image from http://compbio.ucsd.edu/communities-and-cliques/

k-cores I

- A k-core is a maximal subgraph that contains vertices of degree k or more.
- The task of extracting a k-core from a graph can be carried out in linear time.
- k-cores are extensively used in several contexts
  - In social networks, to identify key nodes in a network or to measure the influence of users
  - In biology, to predict features of functional-unknown proteins and to characterize nodes in a protein network that are both necessary and evolutionarily conserved
  - In economics, to simulate the spreading of a crisis in global economic networks

k-cores II

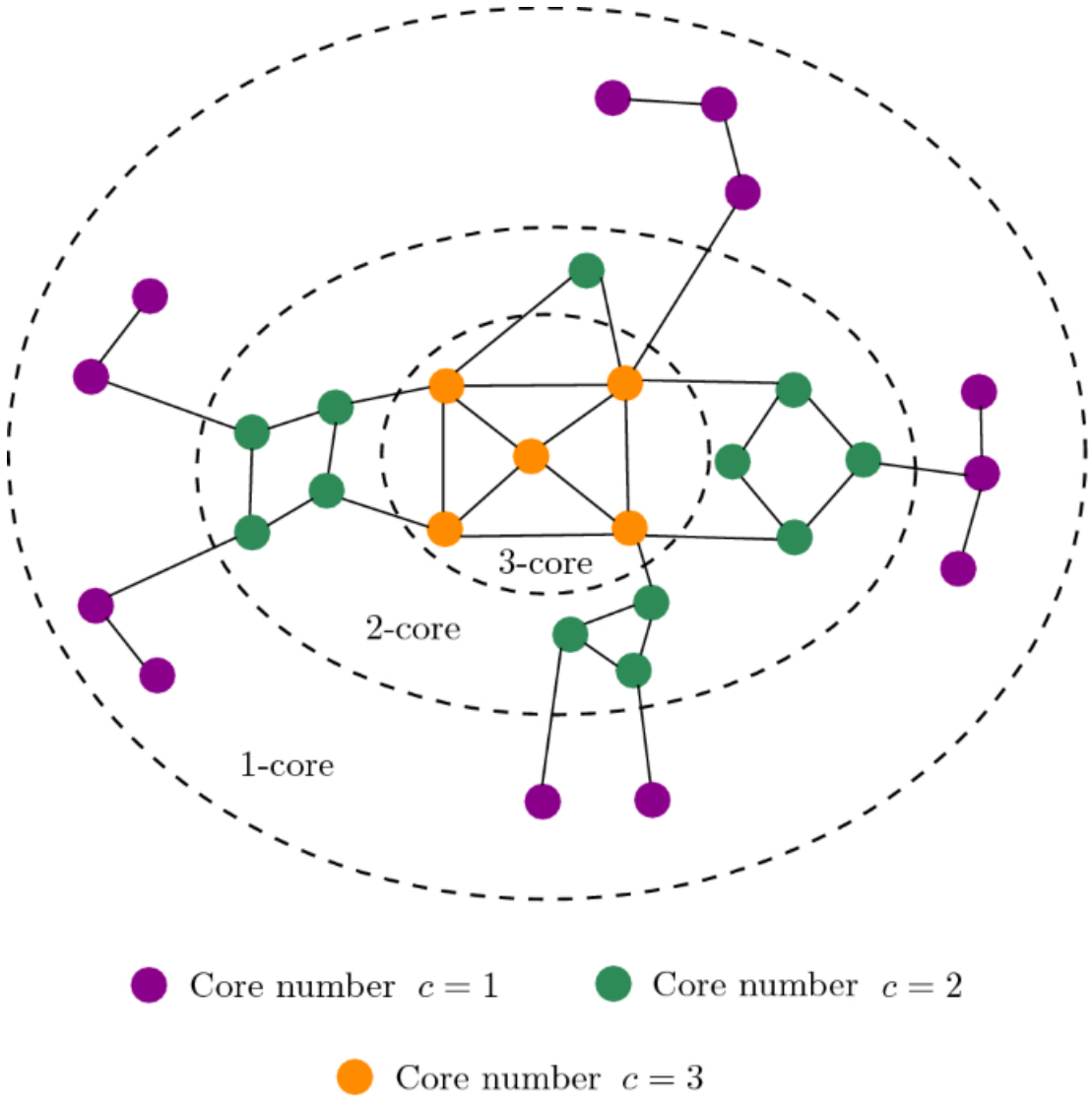


Image from [13]

Tasks on graphs

- Graph structures are interesting both from a theoretical and practical point of view
  - Graph theory is a field in mathematics focusing on the study of graph.
  - The paper by Leonhard Euler on the Seven Bridges of Königsberg (1736) can be seen as the first paper in history of graph theory.
  - Since then, graph theory evolved in a gigantic way and it is today one of the most prolific and interesting fields of research.
- Modeling complex systems with graphs then became an obvious step
  - Graphs representing real systems are objects where order coexists with disorder [7].
  - Network science is just one outcome of that step
- There are countless tasks on graphs/networks, and especially in computer science and machine learning, some of the most famous are
  - Node classification
  - Link prediction
  - Community detection

Node classification

- **Node classification**, also called **label prediction**, is the task of predicting an attribute of each vertex in a graph.
- Examples are labelling each vertex with a categorical class (binary or multiclass classification), or predicting a continuous value (regression).
- The task is supervised or semi-supervised, where the model is trained using a subset of vertices that have ground-truth labels.
- Such task can also be tackled by unsupervised approaches, such as random walk-based methods.

Link prediction

- A **link prediction** task predicts an attribute of edges in a graph.
- Examples are predicting whether an edge that is not already in the graph should exist, or labelling existing edges with a categorical class, or predicting a continuous value.
- It is a common task in dynamic graphs, i.e., graphs whose vertices and edges may vary in time.

Community detection

- **Community detection** is the problem of identifying the modules and, possibly, their hierarchical organization in a graph by only using the information encoded in the graph topology.
- Sometimes also called *graph clustering*, it is a very intuitive problem. However, it is not well defined.
  - The main elements of the problem themselves, i.e., the concepts of community or cluster, are not rigorously defined.
  - Therefore, there are plenty of works in literature which focus on the most disparate contexts and approaches for community detection.
- Traditional methods for community detections span from graph partitioning to spectral clustering, as well as modularity-based methods and divisive algorithm [7].

Working with graph data

- Where to find graph data?
- How to handle graph data?
- How can I visualize my graphs?
- Is there anything else out there?

Where to find graph data?

- SNAP (Stanford Network Analysis Project) datasets
  - `http://snap.stanford.edu/data/index.html`
  - Several kinds of networks (social, communication, citation, web graphs, road networks, etc...)
  - Each dataset is associated to a peer-reviewed paper.
- Network Repository
  - `https://networkrepository.com`
  - An interactive repository with several categories and hundreds of dataset for each of them.

How to handle graph data?

- The mentioned repositories contain graph data in several formats
  - CSV, list of edges, binary matrices, etc...
- To handle such data, we can use existing libraries
- NetworkX (`https://networkx.org/`)
  - Python package for the creation, manipulation, and study of the structure, dynamics, and functions of networks.
  - Very intuitive, it contains a number of algorithms ready-to-use.
  - Not very useful for large-scale networks.
  - Hands-on!

How can I visualize my graphs?

- If we are working with sufficiently small networks, being able to visualize them can help in getting a "big picture" of the system.
- The visualization features in NetworkX can help but they are not suitable for larger graphs
- Gephi (`https://gephi.org/`)
  - It is a visualization and exploration software for networks.
  - It also allows to perform exploratory data analysis (EDA), as well as link analysis, common social network analyses, centralities computation, etc.

Is there anything else out there?

- graph-tool (`https://graph-tool.skewed.de/`)
  - Python module for manipulation and statistical analysis of graphs.
- igraph (`https://igraph.org/`)
  - A collection of network analysis tools programmable in R, Python, C/C++ and Mathematica.
- NetworKit (`https://networkit.github.io/`)
  - Open-source toolkit for large-scale network analysis. It is a Python module written in C++.
- snap (`https://snap.stanford.edu/snap`)
  - General purpose, high performance system for analysis and manipulation of large networks. Written in C++, it also provides a Python interface.
- Cytoscape (`https://cytoscape.org/`)
  - Open source software platform for visualizing complex networks.

References I


Siam. year of the network.
`https://archive.siam.org/news/news.php?id=76`.
Accessed: 2012-09-07.


Albert-László Barabási.
Network science.
Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 371(1987):20120375, 2013.


Austin R Benson, David F Gleich, and Desmond J Higham.
Higher-order network analysis takes off, fueled by classical ideas and new data.
arXiv preprint arXiv:2103.05031, 2021.


Aaron Clauset, Mark EJ Newman, and Cristopher Moore.
Finding community structure in very large networks.
Physical review E, 70(6):066111, 2004.

References II


Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein.
Introduction to algorithms.
MIT press, 2022.


Kousik Das, Sovan Samanta, and Madhumangal Pal.
Study on centrality measures in social networks: a survey.
Social network analysis and mining, 8(1):1–11, 2018.


Santo Fortunato.
Community detection in graphs.
Physics reports, 486(3-5):75–174, 2010.


Per Hage and Frank Harary.
Eccentricity and centrality in networks.
Social networks, 17(1):57–63, 1995.


John Hopcroft and Robert Tarjan.
Algorithm 447: efficient algorithms for graph manipulation.
Communications of the ACM, 16(6):372–378, 1973.

References III


Vito Latora and Massimo Marchiori.
Efficient behavior of small-world networks.
Phys. Rev. Lett., 87:198701, Oct 2001.


David Lusseau.
The emergent properties of a dolphin social network.
Proceedings of the Royal Society of London. Series B: Biological Sciences, 270(suppl_2):S186–S188, 2003.


Kairanbay Magzhan and Hajar Mat Jani.
A review and evaluations of shortest path algorithms.
International journal of scientific & technology research, 2(6):99–104, 2013.


Fragkiskos D Malliaros, Christos Giatsidis, Apostolos N Papadopoulos, and Michalis Vazirgiannis.
The core decomposition of networks: Theory, algorithms and applications.
The VLDB Journal, 29(1):61–92, 2020.

References IV


Mark Newman.
Networks.
Oxford university press, 2018.


Duncan J Watts.
Six degrees: The science of a connected age.
WW Norton & Company, 2004.


Duncan J Watts and Steven H Strogatz.
Collective dynamics of 'small-world'networks.
nature, 393(6684):440–442, 1998.


F Benjamin Zhan and Charles E Noon.
Shortest path algorithms: an evaluation using real road networks.
Transportation science, 32(1):65–73, 1998.