panikzettel.htwr-aachen.de

Social Networks^{2019} Panikzettel

Luca Oeljeklaus, Caspar Zecha, Philipp Schröer

Version 1 — 09.05.2025

Contents

Introduction

This Panikzettel covers the lecture Social Networks, held in the summer semesters of 2019 and 2020 by Prof. Dr. Markus Strohmaier.

> We have two Panikzettel on Social Networks! This one was mainly written in the summer semester of 2019. Daniel Sous also wrote one in 2020: Social Networks 2020 Panikzettel.

This Panikzettel is Open Source. We appreciate comments and suggestions at `https://git.rwth-aachen.de/philipp.schroer/panikzettel`.

1 Graphs

A *simple graph* is a tuple $G = \big(V(G), E(G)\big)$, where $V(G)$ is a finite set of *vertices* (or *nodes*) and $E(G) \subseteq \binom{V(G)}{2}$ is a set of *edges* (or *links*).

An *undirected graph* G satisfies for all $v, w \in V(G)$ that $(v, w) \in E(G) \Leftrightarrow (w, v) \in E(G)$. We also just write $vw \in E(G)$ instead of $(v, w) \in E(G)$ for undirected G.

A *directed graph* (or *digraph*) is a graph which is not undirected.

A *weigthed graph* G has a weight function $w : V(G) \to W$ (where W is usually $\mathbb{Q}$ or $\mathbb{R}$) to every edge. A graph which does not is called *binary* (either an edge exists or it does not).

A *simple multigraph* is a graph where $E(G)$ is a multiset (there may be several edges with the same source and target). It may also contain self-loops: there may exist $v \in V(G)$ such that $vv \in E(G)$.

A digraph G contains a cycle if there exists a set of edges $v_1v_2, \ldots, v_{n-1}v_n, v_nv_1 \in E(G)$ for an $n \geq 2$. A digraph is *acyclic* if it does not contain a cycle.

Given some graph G, we generally refer to the number of vertices by n and to the number of edges by m.

We denote by $N_G(v) = \{\, w \mid vw \in E(G) \,\}$ the *neighbourhood* of v in G. We may omit the subscript if it is clear from context. Similarly, we denote by $N_G(v, w) = \{\, y \mid vy \in E(G) \vee wy \in E(G) \,\}$ the set of common neighbours of v and w.

We refer to the average degree of a graph as $\langle k \rangle$. For a vertex v, the degree of v may be written as $\deg_G(v)$, k_v or $|N_G(v)|$. When talking about in- or out-degrees, we generally add an ^{in} or ^{out} superscript.

1.1 Adjacency Matrix

Let $G = (V(G), E(G))$ be a digraph. The *adjacency matrix* $A \in \{0,1\}^{n \times n}$ of G is given by:

$$A_{ij} = \begin{cases} 1, & \text{if } ij \in E(G), \\ 0, & \text{otherwise.} \end{cases}$$

When considering weighted graphs, we let $A_{ij} = w_{ij}$.

The *density* of an undirected graph is given by $\frac{|E(G)|}{\left|\binom{V(G)}{2}\right|} = \frac{2 \cdot |E(G)|}{|V(G)| \cdot (|V(G)|-1)}$.

We call a graph *sparse* if $|E(G)| \approx |V(G)|$. In these cases, an adjacency matrix is an inefficient way of storing the graph. A graph for which the number of edges is significantly larger than the number of vertices is called *dense*. Most observed graphs are sparse.

An alternative representation for sparse graph is the *edge list*, where we simply store a list of edges where every edge is represented by a 2-tuple containing its source and its target vertex (for weighted edges, we may use 3-tuples, where the third stored value is the weight of the edge).

1.2 Clustering Coefficient

For undirected graphs, the *clustering coefficient* is a measure of how connected the neighbourhood of a vertex is. For some $v \in V(G)$ it is given by

$$c_v = \frac{2 \cdot e_v}{\deg_G(v) \cdot (\deg_G(v) - 1)},$$

where e_v is the number of edges between the neighbours of v. The clustering coefficient of a graph is given by:

$$C_G = \frac{1}{|V(G)|} \cdot \sum_{v \in V(G)} c_v.$$

An alternative to the clustering coefficient is the *potential triangle count*, which is given by:

$$P_G = \frac{\text{number of triangles} \cdot 3}{\text{number of connected triples}}.$$

1.3 Small World Property

A graph G satisfies the *small world property* if it has, compared to a random graph G' of the same order, more or less the same average path length L_G and a much higher clustering coefficient.

Definition: Small World Property

Formally, G must satisfy $\sigma = \frac{\frac{C_G}{C_{G'}}}{\frac{L_G}{L_{G'}}} > 1$.

1.4 Components and Paths

We call the largest component in a graph the *giant component*, usually when 90% or more vertices are contained in it.

A digraph is *strongly connected* if for every pair of vertices there exists a path leading from one to the other and vice versa. It is *weakly connected* if this is satisfied for at least one direction.

Components can be identified with *Depth First Search* (DFS) or *Breadth First Search* (BFS).

Algorithm: Depth First Search

Input: A graph G, a vertex v.
Output: All nodes reachable from v.

1. Label v as discovered.
2. For all neighbours w of v:
   a) If w is not labeled as discovered:
      i. Recursively call DFS with G and w.
3. Return discovered nodes.

Algorithm: Breadth First Search

Input: A graph G, a vertex v.
Output: All nodes reachable from v.

1. Label v as discovered.
2. Create queue Q and enqueue v onto Q.
3. While Q is not empty:
   a) Dequeue w from Q:
   b) For all neighbours x of w that are not labeled as discovered:
      i. Label x as discovered.
      ii. Enqueue x onto Q.
4. Return discovered nodes.

Dijsktra's Algorithm

Dijsktra's algorithm is used to compute the shortest path (also called *geodesic path*) from one vertex to all the others. This version here stops when a certain destination node is discovered.

Algorithm: Dijsktra's Algorithm

Input: A graph G, an initial vertex v, a destination vertex z.
Output: A distance function $d : V(G) \to \mathbb{N}$ which assigns every $w \in V(G)$ its distance to v.

1. Create queue Q of all vertices, and set distance $d(v) = 0$ and $d(w) = \infty$ for all other $w \in V(G)$. Label all nodes as undiscovered.
2. While Q is not empty:
   a) Dequeue the node w with smallest tentative distance $d(w)$ from Q and label it as discovered.
   b) If w is the destination node z then terminate.
   c) For each neighbour x of w that has not been labeled as discovered:
      i. Compute the tentative distance $d(x)$: The smallest value between the current distance $d(x)$ and the sum of the distance $d(w)$ plus the weight of the edge (w, x).

The length of the longest shortest path of a graph is called the *diameter* of the graph.

1.5 Centrality Measures

There are a few different graph metrics which are referred to as *centrality measures*. These centrality measures compute a score for every vertex which tells us how central it is to the network it is located in.

Definition: Degree Centrality

For a digraph, the *degree centralities* of a vertex are given by:

$$dc_v^{\text{in}} = \sum_{w \in V(G)} A_{vw} \quad \text{and} \quad dc_v^{\text{out}} = \sum_{w \in V(G)} A_{wv}.$$

For undirected (weighted) graphs it is given by: $dc_v = \sum_{w \in V(G)} A_{vw}$.

The above centralities can be normalised by dividing by $n - 1$.

However, degree centrality is a local measure and does not take into account the location of a vertex in a network.

Definition: Closeness Centrality

The *closeness centrality* of a vertex v is given by:

$$cc_v = \frac{1}{d_v},$$

where $d_v = \frac{1}{n-1} \sum_{w \neq v} d_{vw}$ is the average distance from v to all other vertices. A high closeness centrality implies that a vertex is on average close to all other vertices.

Closeness centrality is already more precise than degree centrality. However, in small diameter networks it often turns out to be ineffective as the range of values is too narrow. It is also very sensitive to variations in the network. Further, it is undefined in networks with multiple components.

Definition: Betweenness Centrality

The *betweenness centrality* of a vertex v is given by:

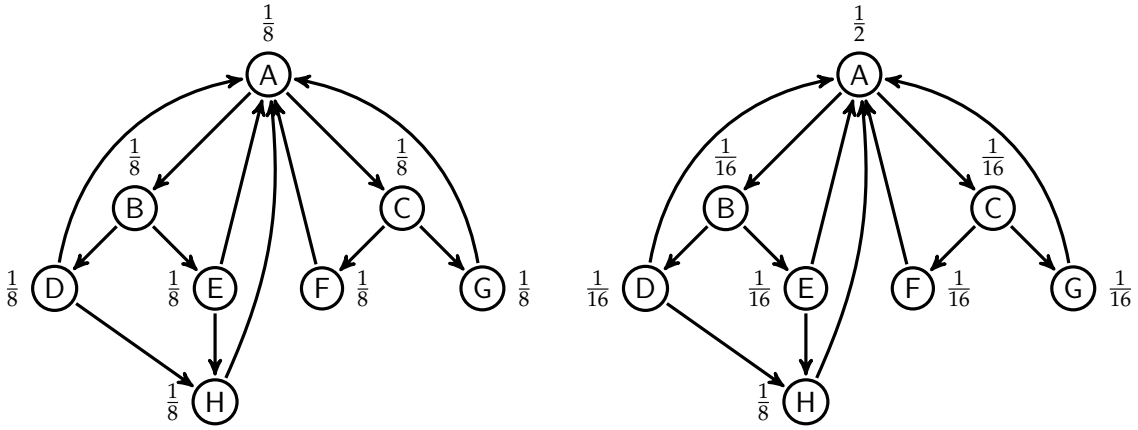
$$bc_i = \sum_{v,w \in V(G)} \frac{\mathsf{sp}(v, i, w)}{\mathsf{sp}(v, w)},$$

where $\mathsf{sp}(v, w)$ is the number of shortest paths from v to w, while $\mathsf{sp}(v, i, w)$ is the number of shortest pathes from v to w that pass through i.

Betweenness centrality, while pretty good, is limited to shortest paths, while generally information can travel long distances. Furthermore, it has time complexity $\mathcal{O}(n^3)$, which is pretty bad.

Eigenvector Centrality *Eigenvector centrality* is a measure of the influence of a node and is computed recursively. Initially, every vertex v is assigned $ec_v = 1$. In every step, it is then updated by assigning each vector the sum of the centralities of the neighbours, normalised by the maximum observed centrality.

PageRank *PageRank* works in a similar way to eigenvector centrality. Every vertex is initialised to the value $\frac{1}{n}$ and in every turn, the value of every vertex is divided by its number of out-edges and sent to its neighbours until the values converge.

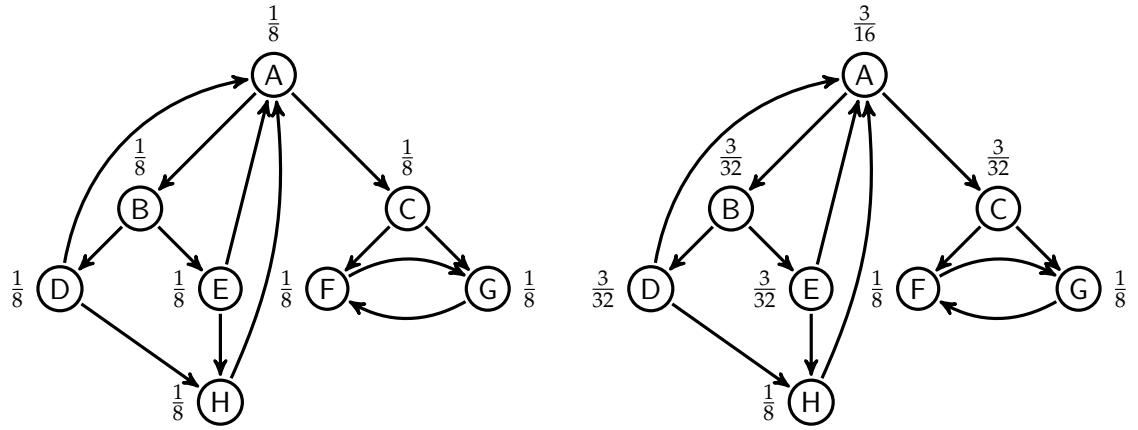


Initialisation First Iteration

In this diagram: The first iteration of PageRank exemplified.

Scaled PageRank In some graphs, PageRank results in all the value being concentrated into few vertices which act as sinks. To counteract this, after every step, *scaled PageRank* scales down the value of each vertex equally by some s and redistributes the rest equally to all nodes such that the sum becomes one again. Formally, this means that:

$$\mathsf{PR}(i) = \frac{1-s}{n} + s \cdot \sum_{\forall j : i \in N_G(j)} \frac{\mathsf{PR}(j)}{\deg_G^{out}(j)}.$$



Initialisation First Iteration

In this diagram: A network in which regular PageRank would let F and G behave as sinks and the first iteration of scaled PageRank with $s = \frac{1}{2}$.

2 Network Models

2.1 Random Graphs

Erdős–Rényi Graphs An *Erdős–Rényi graph* $G(n, p)$ takes two parameters: n, the order of the graph to be generated, and p, the probability for an edge to exist.

Properties	
Mean degree	$c = (n-1) \cdot p$
Degree distribution	$P(deg_G(v) = k) = \frac{c^k}{k!} e^{-c}$
Clustering coefficient	$C = \frac{c}{n-1} = p$
Diameter	$\ell = \frac{\ln n}{\ln c}$
Fraction of vertices in GCC	$S = 1 - e^{-cS}$
GCC critical average degree	$c = 1$

Erdős–Rényi graphs have a giant connected component if their average degree is at least one. Their diameter is relatively small and their clustering is low in comparison to empirical networks.

2.2 Small-World Networks

Recall the small-world property from section 1.3 where graphs with this property have small distance but a high clustering coefficient.

To generate a small-world graph with n nodes, a number x, and probability p we follow these steps:

1. Start with a ring with n nodes and connect every node with its $\frac{x}{2}$ neighbours to each side. Thus, every node has degree x.
2. Go through each edge and reconnect it with probability p: Choose a new node for one end of the edge.

A second way of generating these networks is adding new random edges with probability p, rather than changing the old ones, in the initial ring. This results in the ring structure plus a random graph.

Properties	
Mean degree	$c = xp$
Degree distribution	$P(deg_G(v) = k) = e^{-xp} \frac{(xp)^{k-x}}{(k-x)!}$
Clustering coefficient	$C = \frac{3(x-2)}{4(x-1)}$ (if $p = 0$)
Diameter	$\ell = \frac{n}{2x}$

2.3 The Configuration Model

The *configuration model* is a random graph model with two parameters: $G(n, \vec{k})$, where $\vec{k}$ is a sequence of degrees, one for each node. There are two ways to generate a graph with the configuration model.

Approach 1 is *probabilistic links*: Calculate the probability that a link between nodes i and j exists: $p_{i,j} = \frac{k_i k_j}{2m} = \frac{k_i k_j}{\sum_{l=1}^{n} k_l}$. Then fill the adjacency matrix with 1 or 0 correspondingly. Approach 2: We add stubs equal to the degree in the degree sequence and repeatedly connect two stubs randomly.

The graph has a giant connected component if the condition $\langle k^2 \rangle - 2\langle k \rangle > 0$ is fulfilled.

Properties	
Mean degree	?
Degree distribution	Specified
Clustering coefficient	$C = \frac{1}{n}\frac{(\langle k^2\rangle - \langle k\rangle)^2}{\langle k\rangle^3}$
Diameter	$\ell = \mathcal{O}(\log n)$

2.4 The Barabási-Albert Model

We introduce the *scale-free property* which is also called the *power-law*. The functional relationship between the node degree x and the frequency f is $f(x) = ax^{-k}$. This results in many nodes with low degree and few nodes with high degree. The distribution follows the principle of "the rich get richer".

The *Barabási-Albert model* generates scale-free graphs. Generation is done in two phases:

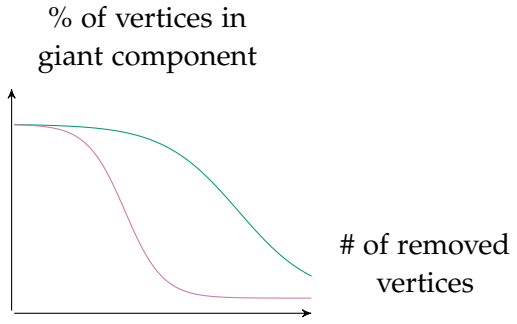
0. Start with m_0 nodes and connect them arbitrarily, such that every node has at least one link.
1. Growth step: At each timestep add a new node and connect it with $m(\leq m_0)$ links that connect to m nodes already in the network.
2. Preferential attachment: The probability $\Gamma(k)$ that a link of the new node connects to node i depends on the degree k_i: $\Gamma(k_i) = \frac{k_i}{\sum_j k_j}$

Many real-world networks are scale-free but many are also not.

2.5 Network Robustness

We now look at the robustness of networks to node and edge failures.

Robustness of Erdős–Rényi Graphs On the right, we approximate the share of vertices remaining in the giant component of Erdős–Rényi graphs when successively removing random vertices and when always removing the vertex with the highest degree. We observe that succesively removing vertices of large degree very quickly breaks the network, and that this type of network isn't robust against random attacks either.



Robustness of Scale-Free Networks Scale-free networks on the other hand are quasi-immune to random attacks as the only real way of breaking them up is to hit the hubs, which is unlikely as they are comparatively few. However, they are thus much more susceptible to precise attacks on the hubs. We can explain this using the *Molloy-Reed criterion* below.

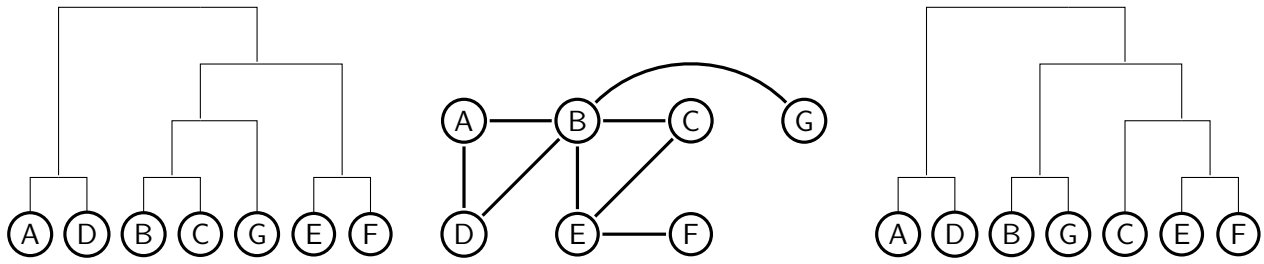
Definition: Molloy-Reed Criterion

The *Molloy-Reed criterion* states that a graph has a giant component if $\frac{\langle k^2\rangle}{\langle k\rangle} \geq 2$.

3 Mesoscopic Structure

3.1 Detecting Communities

Hierarchical Clustering *Hierarchical clustering* is a technique used to arrange nodes of a graph in a so-called *dendrogram* (see examples) such that vertices which are more likely to be in the same community are connected earlier on. The advantage is that here we can first analyse the structure of a graph *before* deciding where to actually split the graph into communities. In the following, we present two heuristics which generate such dendrograms.



In this diagram: A graph G (middle) and two dendrograms, one obtained by the Ravasz algorithm (left), and one obtained by the Girvan-Newman Algorithm (right).

Algorithm: Ravasz

Input: A graph $G = (V(G), E(G))$.
Output: A dendrogram for G.

1. Set up the similarity matrix $X^0 \in \mathbb{R}^{|V(G)| \times |V(G)|}$ $\forall\, v, w \in V(G)$:

$$X^0_{vw} = \frac{|N_G(v,w)| + \Theta(A_{vw})}{\min(k_v, k_w) + 1 - \Theta(A_{vw})},$$

   where Θ is the *Heaviside function*: $\Theta(x) = 0$ if $x \leq 0$, $\Theta(x) = 1$ otherwise.
2. Assign every node to a community of its own.
3. While there is more than one community:
   a) For all pairs of communities A and B, compute the *average linkage similarity* given by:

$$als(A,B) = \frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} X^0_{ab}.$$

      (This is just the average pairwise similarity between vertices in A and B.)
   b) Merge the pair of clusters where $als(A,B)$ is maximal.
4. Generate the dendrogram in the order in which the communities were merged, i.e. if two singleton communities are merged, they are connected at the lowest step in the dendrogram.

Algorithm: Girvan-Newman

Input: A graph $G = (V(G), E(G))$.
Output: A dendrogram for G.

1. While $E(G) \neq \varnothing$:
   a) For every edge, compute $vw \in E(G)$ the *link betweenness* L_{vw} (analogous to node betweenness centrality, but for links).
   b) Remove the edge with max L_{vw}. If there are multiple, choose one randomly.
2. Generate the dendrogram in the reverse order in which the edges were removed, i.e. if an edge is removed last, the corresponding vertices are connected at the lowest step of the dendrogram.

The Ravasz algorithm is called an *agglomerative algorithm* as we successively merge vertices into larger and larger communities, while the Girvan-Newman algorithm is a *divisive algorithm*, where we break up the network successively and observe which parts stay connected the longest. By varying the heuristics we use we can obviously obtain different results, for example by instead of computing the average linkage similarity in the Ravasz Algorithm, choosing the communities with the largest overall linkage.

3.2 Cutting a Dendrogram

There are multiple heuristics for cutting a dendrogram. For example, we can for try to split such that the coverage, i.e. the ratio between in-community edges versus total number of edges is maximised. Another more complex method is to maximise the modularity of a partition.

3.3 Modularity

The *modularity of a partition* is given by

$$M = \sum_c \left(\frac{|E_c(G)|}{|E(G)|} - \left(\frac{k_c}{2 \cdot |E(G)|} \right)^2 \right),$$

where k_c is the sum of the degrees within a community c and $E_c(G)$ are the edges.

Algorithm: Greedy Modularity Maximisation

Input: A graph G and a corresponding dendrogram.
Output: A partition of $V(G)$.

1. Consider the partition P where every vertex is an individual community.
2. Compute M, the modularity of the current partition.
3. While $|P| > 1$:
   a) Store M.
   b) For every one merge of the dendrogram, compute the change ΔM of the modularity.
   c) Follow the merge where ΔM is the largest.
   d) Compute M, the modularity of the current partition.
4. Return the visited partition of the largest modularity.

The modularity of a partition can also be computed in another way:

$$M = \sum_c \left(e_{cc} - k_c^2 \right),$$

where for every community c, e_{cc} is the ratio of in-community edges to all edges, while k_c is the ratio of the sum of vertex degrees in c to the overall sum of vertex degrees ($2m$).

Label Propagation *Label propagation* is a very fast and local algorithm to detect communities. First, every vertex is initialised to have a unique label. Then, in every step (and generally in a random order), every vertex is assigned the label of which it has the most neighbours. Ties are resolved by a random choice. This algorithm can find any number of communities of any size and is very fast. However, its results are not unique and it can get stuck in oscillations, for example in bipartite subgraphs.

3.4 Judging the Quality of Detected Communities

Rand Index Assume you are given two partitions X and Y of a same graph, where X is the actual partition and Y is one obtained by some algorithm, the *Rand index* $\mathcal{R}$ is a measure of how well the computed partition fares compared to the actual one. It is given by

$$\mathcal{R} = \frac{n_{00} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}}.$$

n_{00}: number of pairs vertices in different communities in both X and Y.
n_{01}: number of pairs vertices in different communities in X but not Y.
n_{10}: number of pairs vertices in same communities in X but not Y.
n_{11}: number of pairs vertices in same communities in both X and Y.

Homophily Given a graph and two types of labels with occurring probabilities p and $1-p$ respectively, the real probability for an edge to have both its endpoints of the same label is $p^2 + (1-p)^2$, while the probability for an inter-label edge is $2p(1-p)$. We say that a network tends to show *homophily* if the share of inter-label edges is significantly less than $2p(1-p)$. On the other hand, if it is significantly larger, we talk about *heterophily*.

In general, modularity is a good indicator for homophily.

Two influencing factors for homophily in social networks are *influence* and *selection*: people tend to become more like the people around them and people tend to connect more to people which are already like themselves.

Degree Assortativity The *degree assortativity* of a graph is given by:

$$r = \frac{S_1 S_e - S_2^2}{S_1 S_3 - S_2^2},$$

where $S_e = 2 \cdot \sum_{vw \in E(G)} k_v k_w$ and for $p \in \{ 1, 2, 3 \}$, $S_p = \sum_{v \in V(G)} k_v^p$.

A network with high degree assortativity ($r > 0$) tends to have a structure where hubs connect to hubs and low degree vertices tend to connect to low degree vertices. We call this a *core-periphery* structure. In a network with high degree *disassortativity* ($r < 0$), hubs tend to connect to small degree vertices and vice-versa. Erdős–Rényi and Barabási-Albert graphs in general have $r \approx 0$.

Degree Correlation Function The *degree correlation function* $k_{nn}(k)$ returns the average degree of an arbitrary node we reach by selecting a random node of degree k and following a random link, or:

$$k_{nn}(k) = \sum_{k'} k' \cdot P(k'|k).$$

Friendship Paradox The *friendship paradox* states that most vertices have less neighbours on average than their neighbours. Given a random graph with average degree $\langle k \rangle$ and selecting a random vertex, we can show that his neighbours have $\frac{\langle k^2 \rangle}{\langle k \rangle}$ neighbours on average.

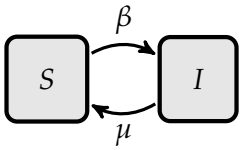
4 Dynamics and Spreading

Networks can be used to model spreading, for example how diseases spread in a population. The population is modelled by a graph of people as nodes and interactions as vertices.

4.1 Infection Models

The *characteristic time* of a given model is the number of steps needed to infect about $\frac{1}{e} \approx 36\%$ of the nodes. The *basic reproductive number* is the average number of individuals an infected node will infect before recovering in a completely susceptible population.

SIS Model



Characteristic Time	$\tau = \frac{1}{\mu \cdot (R_0 - 1)}$
BRN	$R_0 = \frac{\beta \langle k \rangle}{\mu}$
Spreading Rate	$\lambda = \frac{\beta}{\mu}$
Epidemic Threshold	$\lambda_c = \frac{\langle k \rangle}{\langle k^2 \rangle}$

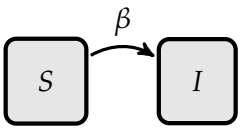
If $R_0 > 1$, then every node infects at least one other before becoming susceptible again. The epidemic is endemic. In terms of the spreading rate, the epidemic spreads if $\lambda > \lambda_c$.

If $R_0 < 1$, then every nodes infects less than one other before becoming susceptible again. The epidemic dies out. In terms of the spreading rate, the epidemic dies out if $\lambda < \lambda_c$.

Observe that for power law networks, λ_c diverges when $n \to \infty$, which implies that in these networks, even very slow viruses can spread successfully there.

In this model, either the infection dies out, or after enough time some fraction of the population (given by $1 - \frac{\mu}{\beta \langle k \rangle}$) is infected.

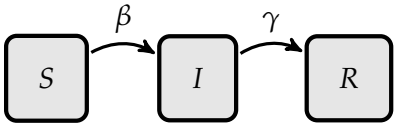
SI Model



Characteristic Time $\tau = \frac{1}{\beta \langle k \rangle}$

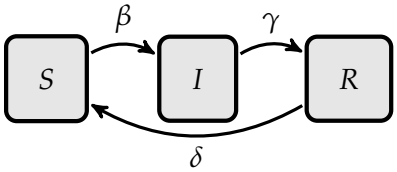
In this model, all nodes become infected given enough time.

SIR Model



In this model, all nodes recover given enough time.

SIRS Model



Analytically not solvable.

Legend	
S	Set of susceptible nodes
I	Set of infected nodes
R	Set of recovered/removed nodes
$\beta \in [0,1]$	In each step, likelihood of an infected node infecting one of its neighbours.
$\mu \in [0,1]$	In each step, likelihood of an infected node becoming susceptible again.
$\gamma \in [0,1]$	In each step, likelihood of an infected node recovering.
$\delta \in [0,1]$	In each step, likelihood of a recovered node to become susceptible again.

Vaccination Strategies An interesting questions is, "how can we stop the spread of infections in real networks effectively and at a limited cost?" Three approaches have been discussed in the lecture:

- Random vaccination: Ineffective, as power-law networks are very robust against random node attacks.
- Hub vaccination: Theoretically ideal as it would break the network. Impractical however, as this requires knowledge of hubs.
- Random neighbour vaccination: Sample the population and vaccinate, for each node, a random neighbour (see Friendship Paradox). Balances the two previous approaches.

4.2 Cascading Behaviour

In this part, we discuss the emergence of group behaviour based on individual thresholds. For a population given by $P = \{1, \ldots, n\}$, let $\theta(i) = x, i \in P$ be the behaviour threshold for individual i, meaning i joins a certain group behaviour only if at least x others are already participating.

Let $f(x)$ be the corresponding frequency distribution while $F(x)$ corresponds to the cumulative distribution. Denoting discrete time steps as $t \in 1, 2, \ldots$, let $r(t)$ denote the number of participants at time t. We can then calculate $r(t+1) := F(r(t))$: the number of people whose threshold is below the number of current participants will join in the next time step.

An *equilibrium* is reached once $r(t) = r(t+1)$: when no new people join the behaviour, either because they have a higher threshold or because everyone is participating.

If we now assume that f is a normal distribution (pertinent, as in reality this is often approximately the case), we often observe that there is a tipping point tied to the standard deviation of f. Below it, only very few participate, while above it, almost everyone participates in the behaviour.

Network Behaviour Assume a given network G, in which every node can choose between two alternatives, A and B such that, for every edge $vw \in E(G)$, if v and w both chose A or B, they get both payoffs $a > 0$ or $b > 0$ respectively, whereas if they chose different alternatives, they both get payoff 0. Initially, everybody uses alternative B, while a select few choose A for some reason. Then in a given time step every $v \in V(G)$ would adopt A if

$$p \cdot |N_G(v)| \cdot a \geq (1-p) \cdot |N_G(v)| \cdot b,$$

where p is the share of neighbours having adopted A (simple payoff maximisation). The formula can be shortened to $p \geq \frac{b}{a+b}$.

We call it a *complete cascade* if after a certain number of time steps, all nodes switch from B to A. A complete cascade can be stopped by a B cluster of density at least $1-p$: a set of nodes such that

every node in it has selected B and has at least a fraction of p neighbours who have also selected B. It is easy to prove that only such clusters can stop a complete cascade.

4.3 Weak Ties

In this section, we discuss a discovery by Mark Granovetter about job acquisition. He observed that although many people got their jobs through social contacts, most didn't come through close friends (strong ties), but rather through acquaintances (weak ties).

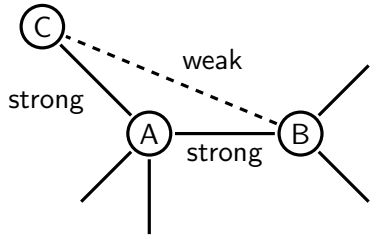
Triadic Closure *Triadic closure* is the term used to describe the phenomenon in social networks in which nodes with a shared neighbour have a stronger than average likelihood of in turn developing a shared link, which in turn provides both with social opportunities.

Local Bridge A *local bridge* is an edge in a network such that its endpoints have no common neighbours.

Strong Triadic Closure Property The *STCP* (which is actually an **assumption**) states that if a vertex has a strong tie to two other vertices, then these two share a tie, either weak or strong.

It implies that every local bridge must have one weak tie. A proof by contradiction is drawn on the right. Two strong ties would imply a weak tie between the other vertices, thus the local bridge isn't a bridge anymore.

Further, the STCP implies that weak ties are those that connect more distant parts of a network and that removing them is more likely to destroy a network than removing strong ties.



In this diagram: A local bridge that isn't one.

This has been observed across a variety of networks, and is among others a nice property to reduce the spread of infections as weak ties are less likely to transmit them.

5 Link Prediction

In this section we discuss how we can, given a network, predict which new links might form in the future and which tools we have to predict these.

5.1 Neighbourhood Approaches

A simple method is to assign pairs of vertices scores based on their neighbourhoods. These scores are convenient as they are easy to understand and easy to implement as they only require local information. However, they are less accurate than other, more complex techniques.

Definition: Common Neighbours

The *common neighbour score* returns, for two $v, w \in V(G)$, the number of neighbours they have in common:

$$S_{CN}(v, w) = |N_G(v) \cap N_G(w)|.$$

Definition: Jaccard

The *Jaccard score* for two $v, w \in V(G)$ is given by:

$$S_J(v, w) = \frac{|N_G(v) \cap N_G(w)|}{|N_G(v) \cup N_G(w)|}.$$

Informally, we measure how many of their neighbours they have in common compared to their overall number of neighbours.

Definition: Adamic-Adar

The *Adamic-Adar score* values shared neighbours of small degree more. Formally:

$$S_{AA}(v, w) = \sum_{u \in N_G(v) \cap N_G(w)} \frac{1}{\log |N_G(u)|}.$$

For example, two people who regularly visit the same small bar are much more likely to meet than two people who regularly attend a same large concert venue.

Definition: Preferential Attachment

The *preferential attachment score* assumes that two hubs are likely to become connected at some point:

$$S_{PA}(v, w) = |N_G(v)||N_G(w)|.$$

5.2 Path Approaches

A somewhat more sophisticated approach would be to use path information to predict new links. However, even these are not very good and computationally very expensive.

Definition: Distance

The *distance score* for two vertices $v, w \in V(G)$ is the negation of the distance of the two:

$$S_D(v, w) = -d(v, w).$$

Definition: Katz Similarity

The *Katz similarity score* sums over all paths of length ℓ, where $\beta \in [0, 1]$:

$$S_{KS}(v, w) = \sum_{\ell=1}^{\infty} \beta^{\ell} \left| \mathsf{paths}_{vw}^{(\ell)} \right|.$$

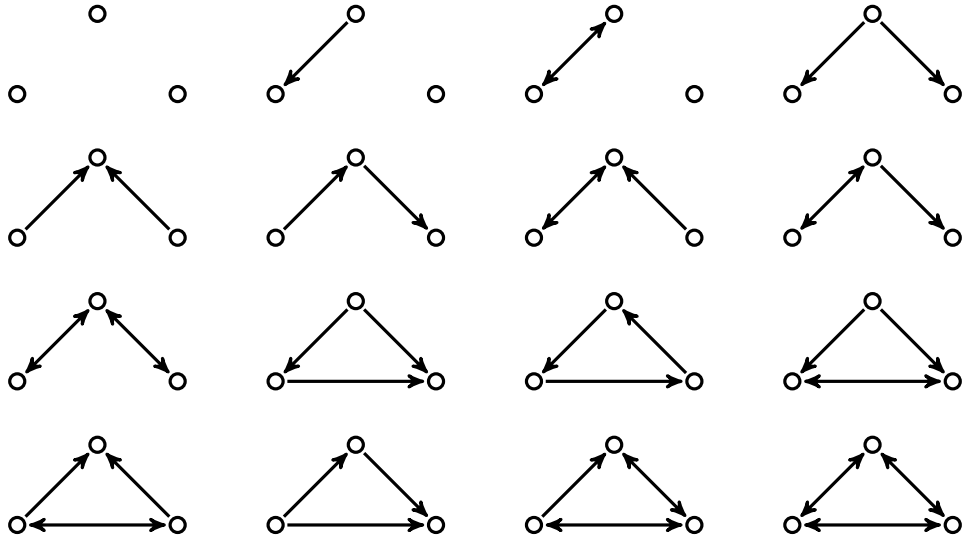
By tuning β, it is possible to adjust how much value to put on longer paths.

More complex machine learning classifiers (logistic regression, SVMs, or NNs) yield much better results. They are however much less transparent in their results. Further, use of meta-data and community structures can yield improved results.

6 Directed and Signed Networks

6.1 Directed Networks

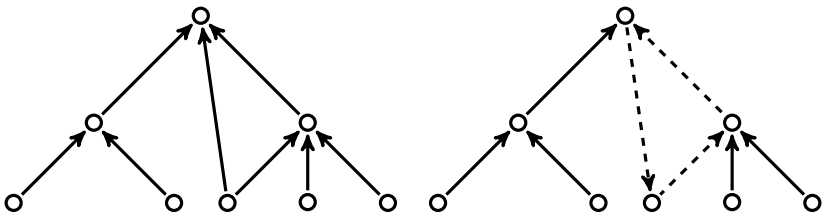
Triadic Census We can do a *triadic census* that evaluates every vertex triple in a graph. The result is a $T \in \mathbb{N}^{1\times 16}$ vector, counting how often each of the 16 triads below occured.



In this table: The 16 triad motifs. All in all, there exist 64 realisations of these triads when distinguishing the vertices.

Flow Hierarchy *Flow hierarchy* is a metric which measures how hierarchical a graph is, and is more nuanced than the binary hierarchical/not hierarchical. Formally, the flow hierarchy is given by the size of the fraction of edges which are not part of a cycle:

$$h(G) = \frac{\sum_{e\in E(G)} \begin{cases} 1, \text{ if } e \text{ is not in a cycle.} \\ 0, \text{ otherwise.} \end{cases}}{|E(G)|}.$$

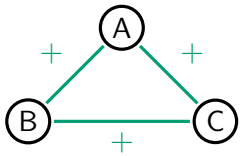


hierarchical graph
$h(G) = 1$

non-hierarchical graph
$h(G) = \frac{5}{8}$

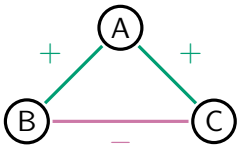
6.2 Signed Networks

There are four configurations of signed triangles to model three-way relations, some of them more stable than the others. A network is called *structurally balanced* if **all** its triangles are balanced (they are either of the first or the third type).



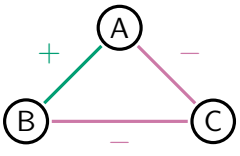
balanced

All three get along.



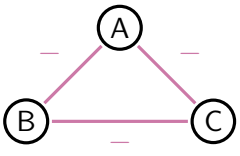
stressful

B and C try to get A to side with them against the other.



balanced

A and B are friends against a common enemy.



stressful

The enemy of my enemy is my friend...

7 Association Networks

Definition A graph is an *association network* if each vertex $v \in V(G)$ has an attribute vector $x_v \in X^n$ with $x_v = (x_{v,1}, \ldots, x_{v,n})$.

7.1 Measuring Node Similarity

Pearson Correlation The *Pearson correlation* is useful for measuring the similarity for vertices v, w with continuous attributes and is given by:

$$\rho_{vw} = \frac{\mathsf{Cov}(x_v, x_w)}{\sqrt{\mathsf{Var}(x_v) \cdot \mathsf{Var}(x_w)}} = \frac{\sum_{v,w}(A_{vw} - k_v k_w / 2m) x_v x_w}{\sum_{v,w}(k_v \delta_{vw} - k_v k_w / 2m) x_v x_w}.$$

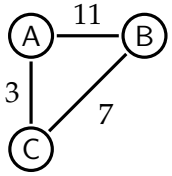
8 Filtering

Minimum Spanning Tree A very simple technique for filtering a network is to compute a minimum spanning tree. For example, we can use Kruskal's algorithm[1] in that we sort all edges by decreasing similarity and then add them successively back to the network while making sure that the result is a tree.

Though MSTs are very easy to compute and somewhat useful, they have the disadvantage of having no cycles or clustering.

Disparity Filter The *disparity filter* approach aims to remove a subset of insignificant edges by computing, for every vertex, the fractional edge weight of all outgoing edges:

$$p_{vw} = \frac{w_{vw}}{\sum_{d=1}^{k_v} w_{vd}}.$$



$p_{AB} = \frac{11}{14}$, $p_{BA} = \frac{11}{18}$.

[1] See e.g. our DSAL Panikzettel (German) for Kruskal's algorithm.

Algorithm: Disparity Filter

Input: A graph $G = (V(G), E(G))$.
Output: A filtered graph $G' = (V(G'), E(G'))$.

1. Select a threshold $\alpha \in [0, 1]$.
2. For every vertex, compute the relative significance score of every outgoing edge, i.e. the weight of the outgoing edge versus all of the vertex' outgoing edge's weights.
3. Keep all edges which are significant for either of its endpoints, i.e. whose significance is larger than α.
4. Return the resulting subgraph.