Statistical Network Analysis

Prof. Dr. Ingo Scholtes

Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science (CAIDAS)
Julius-Maximilians-Universität Würzburg
Würzburg, Germany

ingo.scholtes@uni-wuerzburg.de

Lecture 03
Community Structures and Node Centrality

November 3, 2021

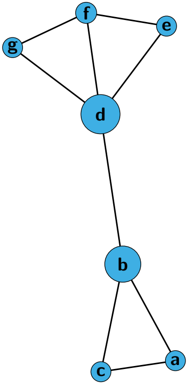
Notes:

- **Lecture L03:** Community Structures and Node Centrality 03.11.2021
- In this lecture, we introduce a first simple algorithm to detect community structures based on a simple measure for partition quality. We further explain how we can use paths to rank nodes by importance.
  - Adjacency matrices, components, and communities
  - Modularity-based community detection
  - Node centrality measures

- **Exercise 01:** Shortest paths, modularity and centralities due 10.11.2021

Motivation

- in L02 we introduced **theoretical foundations** of graph theory and network science
  - mathematical definition of graphs
  - adjacency matrix representation
  - walks, paths, and cycles
  - shortest paths, diameter, avg. path length
  - (strongly) connected components



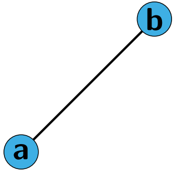
how can we quantitatively analyze networks

- network-level analysis, e.g. components or cluster patterns
- node-level analysis, e.g. centrality of nodes

Notes:

- Last week we explored fundamental concepts and abstractions in network science and graph theory. We defined directed, undirected, and weighted networks, node degrees and introduced walks and paths, adjacency matrices and connected components. These concepts are the basis for advanced methods covered in this and the coming weeks. As a first application of these concepts to network analysis, we will define the partition quality Q, a measure that captures whether we can naturally partition nodes in densely connected communities.
- We conclude this introduction of basic network analytic concepts with an important task, the need to identify "important" or "central" nodes in networks. This has important applications, e.g. to retrieve important (or relevant) documents in information networks, to assess the role of actors in a social organization, or to make statements about critical elements in networked infrastructures.
- As one may have guessed, there is no single notion of importance, there rather is a number of definitions that highlight different aspects of importance. Today we introduce three of basic measures that are often used in (social) network analysis, and which can be simply be defined based on the concepts introduced last week. In a later chapter we complement this view with additional centrality measures that are motivated by dynamical processes and spectral properties.
- All definitions have in common that they provide a measure for the importance of nodes in terms of a real number. Such measures thus define a partial order that we can use to **rank nodes**.

Powers of adjacency matrices



- consider a binary adjacency matrix of a network $G = (V, E)$ with $V = \{a, b\}$ and

$$\mathbf{A} =: \mathbf{A}^1 = \begin{bmatrix} \delta_{aa} & \delta_{ab} \\ \delta_{ba} & \delta_{bb} \end{bmatrix}$$

with $\delta_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise

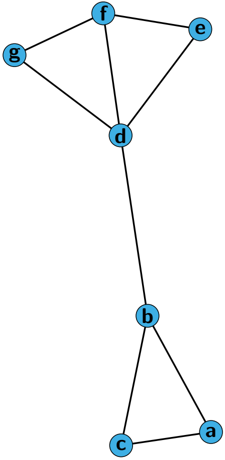
- let us **multiply the adjacency matrix** of G **with itself**

$$\mathbf{A}^2 = \begin{bmatrix} \delta_{aa} & \delta_{ab} \\ \delta_{ba} & \delta_{bb} \end{bmatrix} \cdot \begin{bmatrix} \delta_{aa} & \delta_{ab} \\ \delta_{ba} & \delta_{bb} \end{bmatrix} = \begin{bmatrix} \delta_{aa}\delta_{aa} + \delta_{ab}\delta_{ba} & \delta_{aa}\delta_{ab} + \delta_{ab}\delta_{bb} \\ \delta_{ba}\delta_{aa} + \delta_{bb}\delta_{ba} & \delta_{ba}\delta_{ab} + \delta_{bb}\delta_{bb} \end{bmatrix}$$

Notes:

- Before we introduce partition quality, let us first пересмотреть reconsider the representation of networks in terms of adjacency matrices. A key feature of this mathematical representation is that the multiplication of adjacency matrices naturally relates to the (transitive) notion of walks (or paths) in a network. To better understand this, let us consider an adjacency matrix of a maximally simple network with two nodes a and b. An example for such a network is shown above, but here we do not care about a specific topology. Let us assume that the entries δ_{ab} of the adjacency matrix capture whether an edge from a to b exists in the network, i.e. δ_{ab} is an indicator of the corresponding edge.
- Let us now multiply this adjacency matrix with itself. We apply the rules of matrix multiplication and study the entries of the resulting matrix $\mathbf{A}^2$. We find that those entries count the number of walks of exactly length two between all pairs of nodes, i.e. they are zero if no walk of length two exists and non-zero if one or two such walks exist.
- Example for the top left element: the sum captures the existence of walk $(a, a) \rightarrow (a, a)$ + the existence of walk $(a, b) \rightarrow (b, a)$
- example for the top right element: the sum captures the existence of walk $(a, a) \rightarrow (a, b)$ + the existence of walk $(a, b) \rightarrow (b, b)$

Powers of adjacency matrices



$$
\mathbf{A}^3 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \end{array}
\begin{array}{c}
\begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\
\begin{bmatrix}
2 & 4 & 3 & 1 & 1 & 1 & 1 \\
4 & 2 & 4 & 6 & 1 & 2 & 1 \\
3 & 4 & 2 & 1 & 1 & 1 & 1 \\
1 & 6 & 1 & 4 & 6 & 6 & 6 \\
1 & 1 & 1 & 6 & 2 & 5 & 2 \\
1 & 2 & 1 & 6 & 5 & 4 & 5 \\
1 & 1 & 1 & 6 & 2 & 5 & 2
\end{bmatrix}
\end{array}
$$

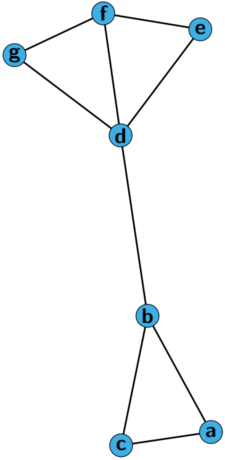
interpretation of $\mathbf{A}^k$

entries A_{ij}^k of k-th power of adjacency matrix count **different walks of exactly length k** between node i and node j

Notes:

- We can test this in our example network from before: Here we have two different walks of length two which start in node f and end in node d. The first one is (f, g, d), the second one is (f, e, d). There are three different cycles of length two which start in node b and end in node b. The first one is the (b, a, b), the second one is (b, c, b), and the third one is (b, d, b).
- For any undirected network without self-loops, the diagonal entries of the squared adjacency matrix $\mathbf{A}^2$ contain the degrees of the corresponding nodes, i.e. $A^2_{ii} = d_i$ This is because
  1. in such a network each undirected link of i yields exactly one cycle of length two from i to i, and
  2. there cannot be other paths of length two that start in i and end in i
- By multiplying the adjacency matrix with itself once more, we now add one to the length of the walks that are counted. Hence, the entries of the matrix $\mathbf{A}^k$ count the walks of exactly length k. Consider the entry $A_{fg} = 5$ in the example of $\mathbf{A}^3$ above: the walks of lengths three between f and g are (f, e, d, g), (f, e, f, g), (f, d, f, g), (f, g, f, g), (f, g, d, g).
- From this, we see that the standard adjacency matrix $\mathbf{A} = \mathbf{A}^1$ is simply a special case that counts the number of walks/paths of length one (which are simply links).
- Looking at the entries of matrix $\mathbf{A}^3$, what else can we say about the topology of our example network?

Adjacency matrices and linear algebra



$$\mathbf{A}^3 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \end{array} \begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\ \begin{bmatrix} 2 & 4 & 3 & 1 & 1 & 1 & 1 \\ 4 & 2 & 4 & 6 & 1 & 2 & 1 \\ 3 & 4 & 2 & 1 & 1 & 1 & 1 \\ 1 & 6 & 1 & 4 & 6 & 6 & 6 \\ 1 & 1 & 1 & 6 & 2 & 5 & 2 \\ 1 & 2 & 1 & 6 & 5 & 4 & 5 \\ 1 & 1 & 1 & 6 & 2 & 5 & 2 \end{bmatrix} \end{array}$$

example network

entries of third power $\mathbf{A}^3$ of adjacency matrix tell us that

1. network is connected
2. diameter is at most three

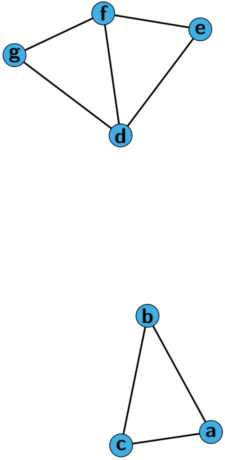
Notes:

- For this specific example, we learn that the **network is connected**, since a path of exactly length three connects all pairs of nodes (there are no zero entries in $\mathbf{A}^3$). This tells us that we can use powers of adjacency matrices to check whether a network is connected or not.

делать вывод

- Since the third power of the adjacency matrix contains no zero entries (and thus walks of length three exist between all pairs of nodes) we can infer that the **diameter of this network** is at most three. In this example network we can even say that the diameter is exactly three, because the last zero entry in the sum $\sum_{k=1}^{l} A^k$ of all matrix powers disappears for $l = 3$. Note that, in general, the diameter of a network can be three even if there are zero elements in the third power of the adjacency matrix. This is because, even though some pairs of nodes are not connected by walks of exactly length three, they can still be connected by shorter paths.
- From the simple example above, we learn something very important: Algebraic methods operating on adjacency matrices can be used to capture non-trivial topological characteristics of networks, like its connectedness or its diameter.
- The reason for this is that paths and walks are transitive, and this transitivity naturally relates to the rules of (repeated) matrix multiplication. We will explore this later in the course (when we deal with spectral properties, eigenvalues, and dynamical processes). We will further see that this assumption of path transitivity may be broken in some networked systems, thus invalidating algebraic methods.

Connected components



$$
\mathbf{A}^3 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \end{array}
\begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\
\begin{bmatrix} 2 & 3 & 3 & 0 & 0 & 0 & 0 \\ 3 & 2 & 3 & 0 & 0 & 0 & 0 \\ 3 & 3 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 5 & 5 & 5 \\ 0 & 0 & 0 & 5 & 2 & 5 & 2 \\ 0 & 0 & 0 & 5 & 5 & 4 & 5 \\ 0 & 0 & 0 & 5 & 2 & 5 & 2 \end{bmatrix} \end{array}
$$

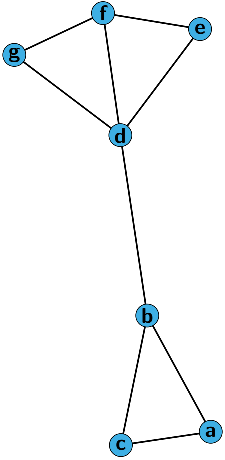
example network

for sufficiently large k and suitably ordered matrix rows and columns, connected components show as **blocks** in $\mathbf{A}^k$

Notes:

- Before studying what else we can see in the example before, let us modify our example by removing the link connecting b to d. Now the network falls apart into two connected components.
- If we now calculate the powers of the adjacency matrix, these two connected components actually show up as **block structures in the matrix powers**.
- This suggests that we could simply detect connected components by searching for block structures in the matrix powers. Unfortunately it is not as easy as that. In the example above, the blocks are only visible because I have ordered the rows/columns in the matrix such that they match the “memberships” of nodes in the connected components. If we were to randomly shuffle the rows/columns (which still represents a network with the same topology), we could not (visually) detect components by blocks.
- For blocks to become apparent we thus need to reorder rows/columns in a meaningful way. We will see how we can do this in a very elegant way, using spectral properties of the adjacency matrix.

What can you see here?



$$\mathbf{A}^3 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \end{array} \begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\ \begin{bmatrix} 2 & 4 & 3 & 1 & 1 & 1 & 1 \\ 4 & 2 & 4 & 6 & 1 & 2 & 1 \\ 3 & 4 & 2 & 1 & 1 & 1 & 1 \\ 1 & 6 & 1 & 4 & 6 & 6 & 6 \\ 1 & 1 & 1 & 6 & 2 & 5 & 2 \\ 1 & 2 & 1 & 6 & 5 & 4 & 5 \\ 1 & 1 & 1 & 6 & 2 & 5 & 2 \end{bmatrix} \end{array}$$

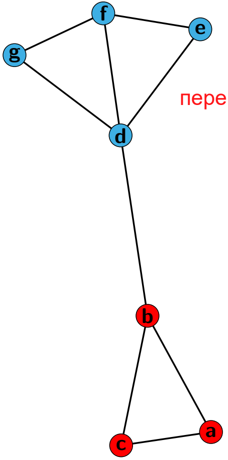
example network

- **blocks** of larger values in $\mathbf{A}^3$ are due to **clusters or communities** in the network
- block structure depends on ordering of rows/columns
- basis for spectral clustering algorithms

Notes:

- Keeping this in mind, we now go back to our example from before, i.e. we add the link from b to d.
- нечеткий
  Here we can see a "fuzzy" block structure that is similar to the one before. We observe blocks of larger values which correspond to the subsets of nodes $\{A, B, C\}$ and $\{D, E, F, G\}$ respectively. The reason for this is that there are many walks of length three between node pairs within $\{A, B, C\}$ and $\{D, E, F, G\}$. However, there are only few walks of length three between node pairs (v, w) where v is from $\{A, B, C\}$ and w is from $\{D, E, F, G\}$.
- густо
  This is because nodes in each of the groups are more densely connected to each other by links than to nodes in the other group. We call such groups of nodes **communities**, **clusters** or **modules** of a network.
- In the example above, we can even see that the two blocks *overlap* in the nodes b and d, which could be counted to either of the two blocks.
- Again, our ability to visually identify these blocks of high values is due to the fact that rows/columns are ordered appropriately. The question how such a reordering of rows/columns can be found automatically is addressed by clustering and community detection algorithms, an important class of **unsupervised machine learning algorithms** for networks. We will learn more about the machine learning perspective on this problem in a later chapter of the course. In the following we will address the problem by maximizing a network-analytic measure that can also be used to quantify the cluster patterns in a network.

Community structures in networks



- clusters, modules or **communities** are partitions $C_i \subseteq V$ such that $\bigcup_i C_i = V$
- перекрывать **overlapping communities** if $C_i \cap C_j \neq \emptyset$
- for non-overlapping communities and $v \in V$ we define

$$c_v := i \iff v \in C_i$$

i.e. c_v and c_w are equal if node v and w belong to the same community

community detection problem

- community detection denotes the problem of finding a **"natural" assignment of nodes v to communities c_v** such that nodes v, w with $c_v = c_w$ are **more connected** to each other than nodes x, y with $c_x \neq c_y$
- network equivalent of **cluster analysis**, where **links between nodes capture pair-wise similarities**

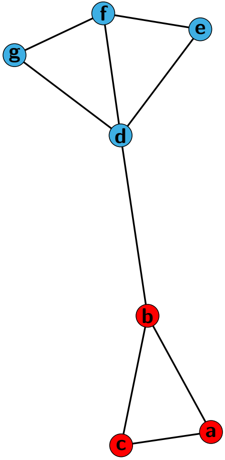
попарно

Notes:

распределение

- This brings us to our first network-analytic problem, namely the problem of finding clusters, modules, or communities in networks. Community detection algorithms try to find a natural assignment of nodes to clusters, modules, or communities such that nodes in the same community are more connected to each other than to nodes in different communities.
- This assignment can take the form of a **partition**, i.e. each node is assigned to exactly one community. We can also consider **overlapping communities**, where a single node can be assigned to multiple communities at the same time. **In this course we will focus on non-overlapping communities**.
- We say that non-overlapping communities should define a “natural partitions” of nodes in the sense that nodes in the same community are more connected to each other than nodes in different communities. This is directly related to the block structure that we have seen in the matrix powers and in this sense community detection can be viewed as a “fuzzy” generalization of connected component calculation.
- To find an “optimal” assignment of nodes to communities (i.e. a partition), we must be able to assign the quality of a partition and there are different ways in which we can do this.

Partition quality



- let $C = \{C_1, \ldots, C_k\}$ be **non-overlapping community partition** of V and let c_v be the community to which $v \in V$ is assigned

- let δ be a **delta function**, i.e.

$$\delta(x, y) := \begin{cases} 1 \text{ iff } x = y \\ 0 \text{ otherwise} \end{cases}$$

partition quality

For an undirected network $G = (V, E)$ with n nodes, m links, adjacency matrix $\mathbf{A}$ and a given community partition C the **partition quality** $Q(G, C)$ is defined as

$$Q(G, C) = \frac{1}{2m} \sum_{i,j \in V} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta(c_i, c_j)$$

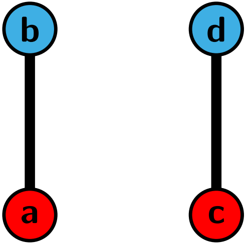
where d_i denotes the degree of node i $\rightarrow$ M Newman, PNAS, 2006

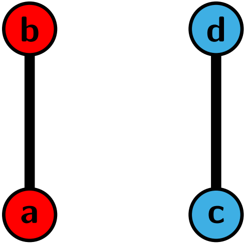
Notes:

- To quantitatively assess the quality of a partition, we introduce a function that is called the **partition quality**. It is defined for undirected networks (with and without self-loops).
- We first use the notation above to denote the (numeric) label c_v of the community assigned to node v. We further define a **delta-function** $\delta(x, y)$ that assumes a value of one whenever $x = y$ and zero whenever $x \neq y$.
- With this, we define the quality $Q(G, C)$ of a given community partition C in a network G as given above. Let us think about the motivation behind this definition. Due to $\delta(c_i, c_j)$, the term in the sum is zero whenever i and j are assigned to different communities, i.e. the expression in the sum is only summed for pairs of nodes that are in the same community.
- $Q(G, C)$ assumes values between a minimum of -0.5 (in which case the partition is contrary to the "natural" partitions in the network) and a maximum of 1 (in which case the partition perfectly captures the "natural" partitions) $\rightarrow$ U Brandes et al., 2008
- In this week's exercise sheet, you will consider two special cases:

  1. a fully connected network with n nodes and no self-loops, where all nodes are in a single community C_1
  2. a network with n nodes, each node only being connected to itself via a self-loop. Each node i is assigned to a different community C_i.

Partition quality: examples





example 1

- partition $C = \{C_1 = \{b, d\}, C_2 = \{a, c\}\}$ is **contrary** to "natural" community structure in the network
- $Q(G, C) = -0.5$

example 2

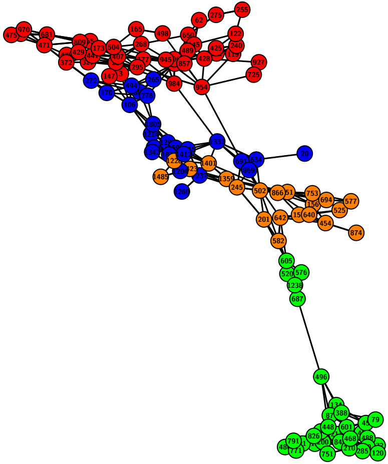
- partition $C = \{C_1 = \{a, b\}, C_2 = \{c, d\}\}$ **matches** "natural" community structure in the network
- $Q(G, C) = 0.5$

Notes:

- To illustrate partition quality, we consider two simple example networks with four nodes (see above) with two different community assignment (colors in left and right example).
- For the left example we have $C_1 = \{b, d\}$ (blue) and $C_2 = \{a, c\}$ (red)

  вопреки
  - Here, community labels are assigned such that they are **contrary to the natural community structure** of the network, i.e. nodes in different communities are connected, while nodes in the same community are not connected.
  - This "negative correlation" between the community labels and the links is expressed by a negative value of $Q(G, C) = -0.5$.
  - As a small exercise, you can confirm that the value of $Q(G, C) = -0.5$ is correct for this network based on the definition of $Q(G, C)$ from the previous slide. What happens if you remove the two links, such that all nodes are disconnected.
- For the right example we have $C_1 = \{a, b\}$ (red) and $C_2 = \{c, d\}$ (blue)
  - Here, community labels are assigned such that they **match the natural community structure** of the network.
  - This "positive correlation" between community labels and the links is expressed by a positive value of $Q(G, C) = 0.5$.
  - As another small exercise, you can confirm that the value of $Q(G, C) = 0.5$ is correct based on the definition of $Q(G, C)$. Can you change the network such that you obtain the maximum value of $Q = 1$?

Modularity maximization



empirical example

social network constructed from contact traces between high school students → www.sociopatterns.org

modularity-based community detection

For given network $G = (V, E)$ find partition $\hat{C}$ with **maximum partition quality** Q_{opt}, i.e.

$$Q_{opt}(G) := \max_{C} Q(G, C)$$

and

$$\hat{C} := \operatorname{argmax}_{C} Q(G, C)$$

- yields **one** heuristic method to detect communities in networks

community detection algorithms

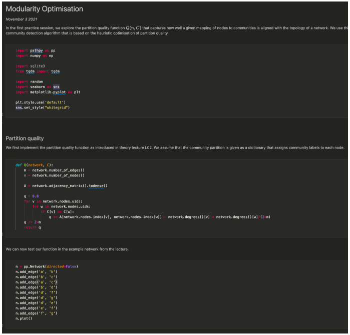
- Kernighan-Lin → BW Kernighan and S Lin 1970
- Girvan-Newman → M Girvan and M Newman 2002
- Greedy optimization → A Clauset et al. 2004
- Clique percolation → G Palla et al. 2005
- WalkTrap → P Pons and M Latapy 2006
- Spectral partitioning → M Newman 2006
- InfoMap → M Rosvall and C Bergstrom 2008

Notes:

- The benefit of the function $Q(G, C)$ is that – apart from evaluating the quality of a partition – we can use it to heuristically find the community partition with maximum partition quality. This is an NP-hard problem, so we need to apply heuristic отжиг optimization algorithms to find near optimal solutions, e.g.simulated annealing, hillclimbing, genetic algorithms, etc. In the first practice session we will consider a simple algorithm to solve this problem.
- Apart from modularity maximization, there are a number of other algorithms that address the community detection problem. These algorithms do not necessarily use the partition quality function to find communities, which results in a different definition of what a community is. Nevertheless all of them share the idea that nodes within one community should be more connected to each other than to nodes in other communities. We will discuss some of those methods in a later chapter, as well as in будущий the forthcoming course *Machine Learning for Complex Networks*, which is currently planned for the next summer semester.

Practice session

- we show how you can use `pathpy` to calculate **partition quality**
- we implement a simple **stochastic optimization algorithm** to calculate the maximum partition quality for a given network
- we show how we can **visualize detected communities** in `pathpy`



Modularity Optimisation

November 3 2021

In the first practice session, we explore the partition quality function $Q(n, C)$ that captures how well a given mapping of nodes to communities is aligned with the topology of a network. We use thi community detection algorithm that is based on the heuristic optimisation of partition quality.

```python
import pathpy as pp
import numpy as np

import sqlite3
from tqdm import tqdm

import random
import seaborn as sns
import matplotlib.pyplot as plt

plt.style.use('default')
sns.set_style("whitegrid")
```

Partition quality

We first implement the partition quality function as introduced in theory lecture L02. We assume that the community partition is given as a dictionary that assigns community labels to each node.

```python
def Q(network, C):
    m = network.number_of_edges()
    n = network.number_of_nodes()

    A = network.adjacency_matrix().todense()

    q = 0.0
    for v in network.nodes.uids:
        for w in network.nodes.uids:
            if C[v] == C[w]:
                q += A[network.nodes.index[v], network.nodes.index[w]] - network.degrees()[v] * network.degrees()[w]/(2*m)
    q /= 2*m
    return q
```

We can now test our function in the example network from the lecture.

```python
n = pp.Network(directed=False)
n.add_edge('a', 'b')
n.add_edge('b', 'c')
n.add_edge('a', 'c')
n.add_edge('b', 'd')
n.add_edge('d', 'f')
n.add_edge('d', 'g')
n.add_edge('d', 'e')
n.add_edge('e', 'f')
n.add_edge('f', 'g')
n.plot()
```

practice session

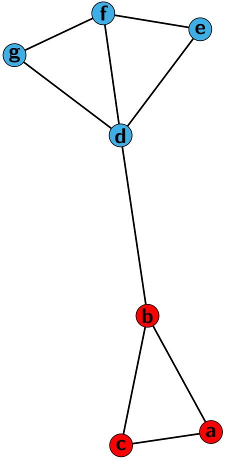
see notebook 03-01 in `gitlab` repository at

→ https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2021_wise_sna_notebooks

Notes:

- In the first practice session of this week, we will show how we can calculate partition quality for a given community assignment and a `pathpy` network. We further show how we can use a simple (and unfortunately not very optimization algorithm) to detect partitions that maximize the partition quality. We finally show how we can use colours to visualize the detected community structures in networks.

Modularity



example

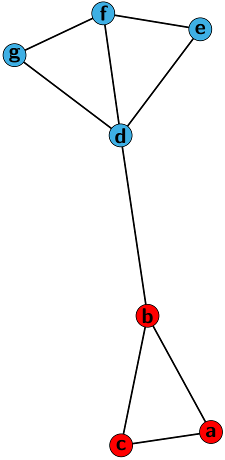
maximum modularity $Q_{opt} \approx 0.36 \ll 1$

- for any network $G = (V, E)$ we can compute the maximum partition quality $Q_{opt}(G)$
- while $Q(G, C)$ is a property of network and partition, $Q_{opt}(G)$ **is a network property**
- $Q_{opt}(G)$ is called **modularity** of a network
- to interpret modularity we must compare Q_{opt} to the **maximum possible modularity** Q_{max} in **given network** G **and partition** C

Notes:

- Using heuristic optimization algorithms we can find a partition such that the partition quality is maximal. But are the detected communities meaningful? Does the network topology exhibit any community structure in the first place?
экспонат
- This is not easy to answer. Whether a given community partition is "significant" in a statistical sense is an important open research problem that we are working on at our chair. For this, we need to answer the question whether the nodes in a community are more connected to each other than we would expect "at random", which motivates random graph models. → L04: Random Graph Ensembles
- For any given partition C the partition quality $Q(G, C)$ is a property of both the network and the partition. However, Q_{opt} is the maximum across all partitions, which turns it into a property of the network. This value is thus called **the modularity of the network**. It captures how well we can partition nodes in communities.
- In our example we obtain $Q_{opt} \approx 0.36$, which is much smaller than the maximum value of 1. But which values of Q_{opt} indicate the presence of "strong" communities structure? Sometimes, a rule-of-thumb threshold of 0.3 is used, which is not satisfactory as the maximum modularity of a network depends on the number of nodes and links.
практическое правило
- In our example, Q_{opt} is smaller than the theoretical maximum value of 1 due to two reasons: (i) there is a link (b, d) that connects nodes in different communities and (ii) there are links missing between nodes in the same community. The latter is simply because there are not enough links in the network. Can we differentiate between these two effects?
различать

Maximum partition quality Q_{max}



example

for $C = \{\{a, b, c\}, \{d, e, f, g\}\}$ we get $Q_{max}(G, C) \approx 0.48 \ll 1$

- consider network G and the optimal community partition

$$C = \{C_1 = \{a, b, c\}, C_2 = \{d, e, f, g\}\}$$

- what value of Q would we get if all links connected nodes in the same community?

- **theoretical maximum of partition quality** for corresponds to $A_{ij} = 1 \Rightarrow \delta(c_i, c_j) = 1$

maximum partition quality Q_{max}

$$Q_{max}(G, C) = \frac{1}{2m}\left(2m - \sum_{i,j \in V} \delta(c_i, c_j)\frac{d_i d_j}{2m}\right)$$

Notes:

- Rather than comparing the value Q_{opt} to the maximum value of 1 that is possible in *any* network, we should compare it to the maximum value Q_{max} that is possible for the network in question (thus accounting for the number of links that actually exist). For this we adapt the definition of Q such that whenever $A_{ij} = 1$ we set $\delta(c_i, c_j) = 1$, i.e. we "pretend" that all links are between nodes in the same community.
- While $A_{ij} = 1 \Rightarrow \delta(c_i, c_j) = 1$, the opposite is not true, i.e. $A_{ij} = 0 \not\Rightarrow \delta(c_i, c_j) = 0$. The reason for this is that, due to the sparseness of the network, there are pairs of nodes in the same community, which are nevertheless not connected by a link. Our Q_{max} value should account for this.
- To account for this we first move $\delta(c_i, c_j)$ in the definition of the partition quality into the brackets, i.e.

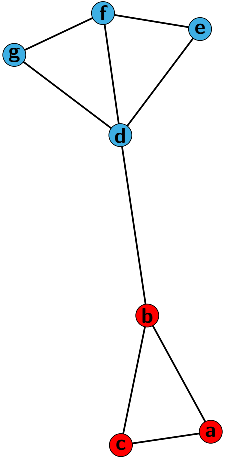
$$Q(G, C) = \frac{1}{2m} \sum_{i,j} \left(\delta(c_i, c_j) A_{ij} - \delta(c_i, c_j) \frac{d_i d_j}{2m} \right)$$

- Since $A_{ij} = 1 \Rightarrow \delta(c_i, c_j) = 1$ we have $\delta(c_i, c_j) A_{ij} = A_{ij}$ and the expression above becomes

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

- The sum of all elements in the adjacency matrix is $2m$, so we get the above expression for Q_{max}, which is smaller than 1 if not all possible links exist.

Community assortativity coefficient



example

$$\frac{Q_{opt(G)}}{Q_{max(G,\hat{C})}} \approx 0.77$$

- assume we found a partition $\hat{C}$ with optimal partition quality Q_{opt} for network G
- ratio between $Q_{opt}(G)$ and $Q_{max}(G, \hat{C})$ quantifies how close a network is to the network with maximum possible modularity

community assortativity coefficient

for an undirected network G with modularity Q_{opt} and optimal community partition $\hat{C}$ the **community assortativity coefficient** is given as

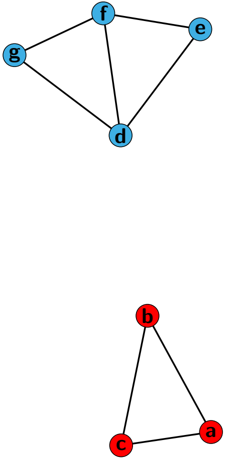
$$\frac{Q_{opt}(G)}{Q_{max}(G, \tilde{C})}$$

- term "assortativity" refers to a preference of nodes to be connected to "similar" nodes

Notes:

- With this definition of Q_{max} we can calculate the so-called community assortativity coefficient, which divides the optimal partition quality Q_{opt} by the (theoretically possible) maximum partition quality Q_{max}.
- In general, assortativity is the preference of nodes to connect to other nodes that are – in some way – similar according to themselves.
- In social systems, this network characteristic is often the result of *homophily*, the tendency of humans to create social ties to others similar in age, gender, profession, etc. Here we define a node's community as its characteristic, i.e. the community assortativity coefficient captures the nodes' preference to connect to nodes in the same community.
- In the example shown above, there is almost perfectly *assortativity* w.r.t. communities (except for the link (b, d))
- The community assortativity coefficient normalizes the partition quality to a range between 0 and 1. Also, this quantity is independent of the number of links in the network. This allows us to judge more easily whether a network exhibits community structures or not.

Maximum community assortativity coefficient



- if we remove link (b, d) in our example, we obtain a network in which ...
  - all existing links connect nodes within the same community
  - not all possible links within communities exist

example

$$Q_{opt} \approx 0.47$$

$$Q_{max} \approx 0.47 \qquad \frac{Q_{opt}}{Q_{max}} = 1$$

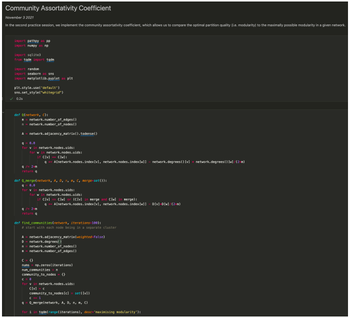
- resulting network has **maximum assortativity coefficient**

Notes:

- To better understand the community assortativity coefficient, we can now remove the only link (b, d) from our previous example that "violates" the optimum partitioning. If we do this, we get a value for $Q_{opt} = 0.47$ that is only marginally larger than before (0.36).
- Computing Q_{max} for this example confirms that this network exhibits the maximum modularity that is possible in a network of this size. We obtain a community assortativity coefficient of one, which tells us that the network has indeed a strong modular structure.
- Note however, that this comes at the price that we lose the ability to differentiate between this network, and a network which has a higher density of links within communities (and which would thus generate a higher value for Q_{opt} as well as a higher value for Q_{max}).

Practice session

- we show to calculate the **maximum partition quality** Q_{max}
- we implement a function to calculate the **community assortativity coefficient**
- we apply community detection to **empirical networks**



practice session

see notebook 03-02 in `gitlab` repository at

→ https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2021_wise_sna_notebooks

Notes:

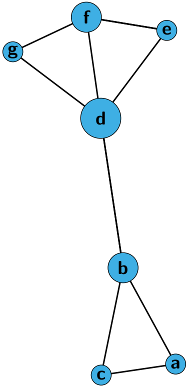
- In the second practice session, we will implement a function that calculates the maximum partition quality that can be theoretically achieved in a network. We will use this to calculate the community assortativity coefficient. We finally apply our knowledge to detect and evaluate community structures in three empirical networks.

Node centrality measures

- important basic task in network analysis is to **identify important nodes**
  - recommender systems
  - information retrieval
  - social network analysis
  - robustness modelling
    прочность

node centrality measures

- For a network $G = (V, E)$ **node centrality** $c : V \to \mathbb{R}$ is a function or measure that can be used to assess the **importance of nodes** in a network.
- For $v \in V$, centrality indicators $c(v)$ provide a partial order than can be used to **rank nodes by importance**.

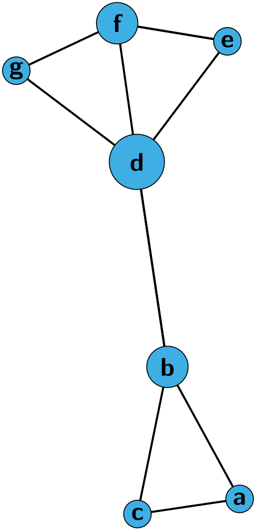


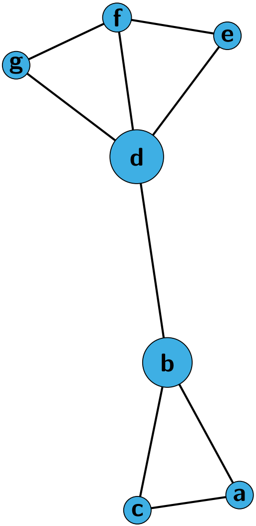
how do we assess the **importance of nodes**?

Notes:

- In the previous have defined modularity, a measure that captures whether we can naturally partition nodes in densely connected communities.
- We conclude this introduction of basic network analytic concepts with an important task, the need to identify “important” or “central” nodes in networks. This has important applications, e.g. to retrieve important (or relevant) documents in information networks, to assess the role of actors in a social organization, or to make statements about critical elements in networked infrastructures.
- As one may have guessed, there is no single notion of importance, there rather is a number of definitions that highlight different aspects of importance. Today we introduce three of basic measures that are often used in (social) network analysis, and which can be simply be defined based on the concepts introduced last week. In a later chapter we complement this view with additional centrality measures that are motivated by dynamical processes and spectral properties.
- All definitions have in common that they provide a measure for the importance of nodes in terms of a real number. Such measures thus define a partial order that we can use to **rank nodes**.

How important is a node?





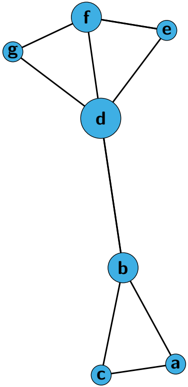
Notes:

неизбежный

- The notion of *importance* inevitably refers to the *role* of a node, which depends on the *function* of a network. Any realistic investigation of *node importance* thus requires a model for the actual function of a system (e.g. based on a dynamical process). Indeed, later in the course we see that we can define measures which capture the importance of nodes with respect to a particular dynamical process, e.g., random walks or surfing in a network.
- Nevertheless, we can also take a purely topological/structural perspective on node importance, e.g. based on the number of connections of nodes or based on shortest paths that result from the topology of those connections. We refer to such measures as *centrality measures* or *centrality indices*, capturing the fact that they measure how “central” a node is in the topological space defined by the network.

Degree-based centralities

- we can define centrality of node $v \in V$ based on the number of incident links
- **degree centrality** $d(v) = d_v$ for undirected networks
- **in- or out-degree centrality** $d_{in}(v)$ or $d_{out}(v)$ for directed networks
- degree-based centralities are **local measures of importance**
  - **number** of incident links
  - independent of **topology** of links



example

- $d(f) = 3$
- $d(b) = 3$
- $d(d) = 4$

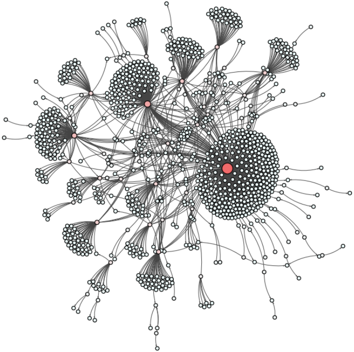
Notes:

- While there are many different approaches in the literature, one of the most straight-forward measures that we can think of **defines importance based on the number of links incident to nodes**. We call this the **degree centrality** of a node. Apart from the simple degree d_v in undirected and unweighted networks, for directed and/or weighted networks we can also use the (weighted) in- or out degree. Depending on the semantics of links (and weights) this can lead to meaningful measures for the importance of nodes.
- Whatever variant of degree centrality we use, it inevitably leads to a **local notion of importance**, i.e. the importance of a node only depends on the number of links of a node rather than to which specific nodes it is connected. In other words, as long as we keep the degrees, **we can change the topology of a network and still get the same degree centralities**.
- This approach has its limitations, as shown in the example on the right. Intuitively, we might say that node b is more important or central than node f but both have the same degree and thus the same degree centrality.
- On the positive side, the **degree centrality can be calculated very efficiently** and it often provides first insights. Moreover, the fact that degree centrality is not influenced by the topology of links (i.e. which nodes they interconnect) makes it a useful baseline against which we can compare other measures.

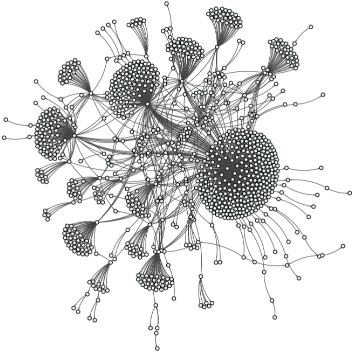
In-degree vs. out-degree centrality

example: KDE community

network of **directed** communication interactions between users and developers of OpenSource Software project KDE



out-degree centralities



in-degree centralities

Notes:

- As an **exemplary application of degree centralities**, we consider the directed communication network of the Open Source Software (OSS) project KDE. Links represent *directed* interactions (e.g. the forwarding of information or the assignment of tasks from one member of the community to another member. In the two figures above, the colors and sizes of nodes indicate their out-degree (left) and in-degree (right) centrality, i.e. small blue nodes have small centrality while large, red nodes have high centrality.
- A large **in-degree** means that a community member is being forwarded information or being assigned tasks by *many different other members*. We find that the in-degree centralities are distributed rather homogeneously, all nodes have similar in-degree (max. in-degree is 10, minimum in-degree is 0).
- For the **out-degree** (left) we have a very different picture. Out-degrees of nodes are broadly distributed, with the maximum out-degree of a node being 416 and the minimum being 0. A large out-degree means that a community member forwards information or assigns tasks to *many different other members*. We see that this is the case for a small number of "central" community members.

шире

- The fact that the out-degree distribution is much broader than the in-degree distribution can intuitively be explained by the fact that it is **easier to delegate tasks** (or information) to many other users, **than to complete tasks** assigned by a large number of other users.

Betweenness centrality

- **betweenness centrality** of node v is defined as

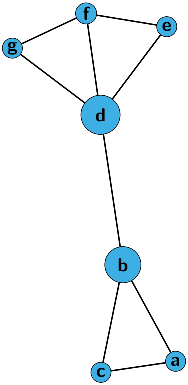
$$C_B(v) := \sum_{s,t \in V-\{v\}, s \neq t} \frac{N_{st}(v)}{N_{st}}$$

  where N_{st} is number of **shortest paths** from s to t and $N_{st}(v)$ is number of shortest paths from s to t passing through v

- in general we have $0 \leq C_B(v) \leq n^2$

- **normalized betweenness centrality** can be defined as

$$C_{\bar{B}}(v) := \frac{C_B(v) - \min_i C_B(i)}{\max_i C_B(i) - \min_i C_B(i)} \in [0, 1]$$



example

- $C_B(f) = 1$ $\quad C_{\bar{B}}(f) \approx 0.05$
- $C_B(b) = 16$ $\quad C_{\bar{B}}(b) \approx 0.83$
- $C_B(d) = 19$ $\quad C_{\bar{B}}(d) = 1$

Notes:

- Degree centrality does not depend on the topology, i.e. to *whom* nodes are connected but only on *how many* links a node has. One way to use the topology of a network to define centrality measures is based on the shortest paths between all pairs of nodes. Since many processes can be assumed to follow (approximately) shortest paths, such **path-based centrality measures** capture a notion of centrality that is meaningful in a variety of systems, e.g. in terms of information propagation and gossiping, routing and navigation, etc.
- An example for a path-based centrality measure is **betweenness centrality** → Linton Freeman, 1977 . It is defined as a sum of fractions in $[0, 1]$, where each term gives the *fraction* of shortest paths between a pair of nodes s and t that pass through node v.
- For node b in the example network, the shortest paths between pairs of nodes $s \neq t$ are: $(a, c), (a, b, d), (a, b, d, e), (a, b, d, f), (a, b, d, g), (c, b, d), (c, b, d, e), (c, b, d, f), (c, b, d, g), (d, e), (d, f), (d, g), (e, f), (e, f, g), (f, g)$ as well as all of the reverse paths (undirected network). b is on all shortest paths (fraction 1) for 16 of the node pairs, i.e. $C_b(b) = 16$.
- The maximum betweenness centrality of a node is bounded above by n^2 and thus grows with the network size (see self-study question). We sometimes use a **normalized betweenness centrality** (see above), which ensures that the minimum value is zero and the maximum value is one.
- Betweenness centrality is a **non-local measure of importance**, i.e. the centrality of a node depends not only on the number of neighbors, but also how these neighbors (and their neighbors) are connected to other nodes. An efficient method to calculate it has been developed in → U Brandes, 2001

Closeness centrality

- **closeness centrality** of node v is the inverse of the average shortest path distance to all other nodes

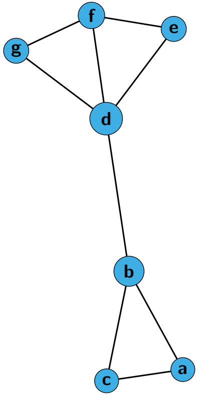
$$C_C(v) = \frac{n-1}{\sum_{w \in V - \{v\}} \text{dist}(v, w)} \in [0, 1]$$

  where n is the number of nodes

- for disconnected networks we have

$$C_C(v) = \frac{n-1}{\infty} := 0 \quad \forall v \in V$$

- betweenness and closeness are **path-based centrality measures** that depend on the **topology** of links



example

- $C_C(f) \approx 0.55$
- $C_C(b) \approx 0.67$
- $C_C(d) = 0.75$

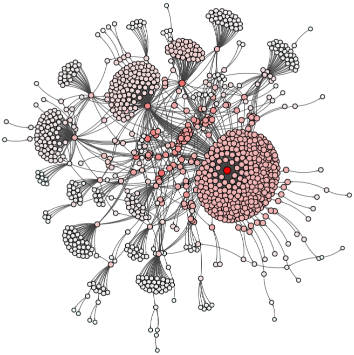
Notes:

- Betweenness centrality emphasizes nodes that lie on shortest paths between **other nodes** but it does not capture how easily a node itself can reach other nodes. Another example for a path-based centrality is **closeness centrality**. It measures how close (in terms of topological distance) a node is to all other nodes. Closeness centrality is another **non-local measure of importance**, i.e. it depends on the topology of the network. It was originally defined in $\longrightarrow$ A Bavelas, 1950
- As an example, we calculate the closeness centrality of node d in our example and find:
  - $dist(a, d) = 2$
  - $dist(b, d) = 1$
  - $dist(c, d) = 2$
  - $dist(e, d) = 1$
  - $dist(f, d) = 1$
  - $dist(g, d) = 1$
- With $n = 7$ we thus have $C_c(d) = \frac{n-1}{2+1+2+1+1+1} = \frac{6}{8} = 0.75$.
- The **maximum closeness centrality** for the center in a star network is 1, since in this case the distance to all $n - 1$ other nodes is 1 and we have $\frac{n-1}{n-1}$. The closeness centrality of any node in a fully connected network is also 1, while the closeness centrality of any node in a disconnected network is typically defined as 0.

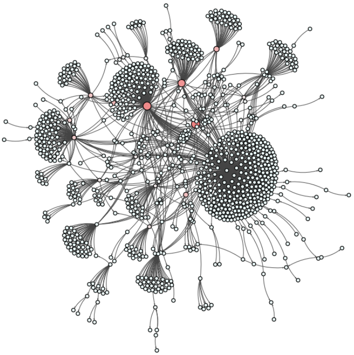
Closeness vs. betweenness centrality

example: KDE community

network of **directed** communication interactions between users and developers of OpenSource Software project KDE



closeness centralities



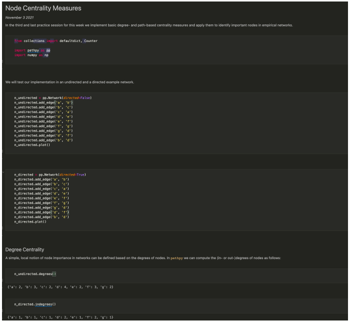
betweenness centralities

Notes:

- We apply those measures to the KDE collaboration network. We again visualize centralities by node colors and sizes, i.e. small blue nodes have small betweenness/closeness centrality and large, red nodes have high betweenness/closeness centrality.
- Closeness centrality (left) naturally relates to the distance of a node to the "*center*" of the network, i.e. it quite literally allows us to identify "central" and "peripheral" parts of the network. Nodes in the periphery have small closeness centrality, while those in the center of the network have high closeness centrality (even though they can have a small degree or they may not lie on many shortest paths). In the example, the central community member on the right slightly "shifts" the center of the network to the right.
- Note that the positions of nodes in the visualization have been computed with a force-directed layout algorithm. It considers links as springs which generate attractive forces between connected nodes, adding a repulsive force between all pairs of nodes. Initializing nodes with a random position, and updating node positions according to these forces generates a positioning of nodes which "naturally" matches the network topology. It moves communities of nodes close to each other and naturally moves nodes with high closeness in the center of the visualization.
- **Betweenness centrality** gives a very different notion of importance. We see that we identify some nodes that have both small in- and out-degree. They are nevertheless important because they lie on a large fraction of shortest paths between all other nodes. Removing those nodes would affect many other pairs of nodes in the sense that those pairs lose at least one shortest path.

Practice session

- we show how to calculate **betweenness and closeness centrality**
- we use different notions of centrality to **rank nodes** in empirical networks
- we show how we can **visualize node centralities** in `pathpy`



practice session

see notebook 03-03 in `gitlab` repository at

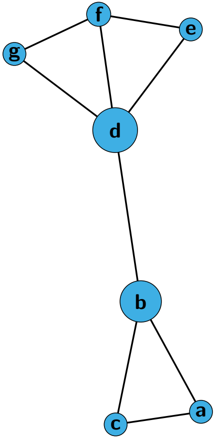
→ `https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2021_wise_sna_notebooks`

Notes:

- In the third and final practice session accompanying this week's lecture, we implement betweenness and closeness centrality based on the all-pairs shortest path algorithms introduced in the previous lecture. We then use this rank nodes in empirical networks and we will use node sizes to visualize node centralities with `pathpy`.

In summary

- we highlighted relations between **paths in networks** and **algebraic operations** on adjacency matrices
- we introduced **partition quality** and its application to **community detection**
- we showed how to quantitatively assess the **importance of nodes** in a network



open questions

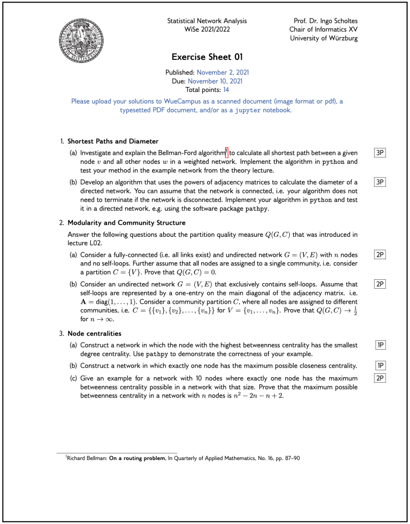
- how can we distinguish **topological patterns** in empirical networks from **random fluctuations**
- what are the **probabilistic foundations** of network analysis?

Notes:

- In today's lecture we introduced the relation between paths and walks in networks and algebraic operations like, e.g. multiplication or powers, on adjacency matrices. This simple and intuitive relation is the basis for advanced network analytic methods, using e.g. eigenvalue spectra of matrix representations of networks, that we will introduce in the following weeks.
- We have further introduced the notion of **community structures** as well as a simple method to detect community structures by maximizing the partition quality Q. This simple problem highlights an important issue: whatever result we obtain for a concrete empirical network (e.g. the modularity of a network) we need to compare it against a random baseline in order to determine whether our finding is "surprising" enough to constitute an actual discovery. We will address such questions based on random graph models that we will introduce in the next lecture.

Exercise sheet 01

- **first exercise sheet** is available on WueCampus
  - implement algorithms to calculate the diameter of networks
  - deepen your understanding of modularity-based community detection
  - explore betweenness and closeness centrality
- solutions are due **November 10th** (via Moodle)
- solutions will be discussed in exercise on November 24th
- present your solution to earn bonus points



Statistical Network Analysis
WiSe 2021/2022

Prof. Dr. Ingo Scholtes
Chair of Informatics XV
University of Würzburg

Exercise Sheet 01

Published: November 2, 2021
Due: November 10, 2021
Total points: 14

Please upload your solutions to WueCampus as a scanned document (image format or pdf), a typesetted PDF document, and/or as a `jupyter` notebook.

1. **Shortest Paths and Diameter**
   (a) Investigate and explain the Bellman-Ford algorithm[1] to calculate all shortest path between a given node v and all other nodes w in a weighted network. Implement the algorithm in `python` and test your method in the example network from the theory lecture. [3P]
   (b) Develop an algorithm that uses the powers of adjacency matrices to calculate the diameter of a directed network. You can assume that the network is connected, i.e. your algorithm does not need to terminate if the network is disconnected. Implement your algorithm in `python` and test it in a directed network, e.g. using the software package `pathpy`. [3P]
2. **Modularity and Community Structure**
   Answer the following questions about the partition quality measure $Q(G, C)$ that was introduced in lecture L02.
   (a) Consider a fully-connected (i.e. all links exist) and undirected network $G = (V, E)$ with n nodes and no self-loops. Further assume that all nodes are assigned to a single community, i.e. consider a partition $C = \{V\}$. Prove that $Q(G, C) = 0$. [2P]
   (b) Consider an undirected network $G = (V, E)$ that exclusively contains self-loops. Assume that self-loops are represented by a one-entry on the main diagonal of the adjacency matrix. i.e. $\mathbf{A} = \text{diag}(1, \ldots, 1)$. Consider a community partition C, where all nodes are assigned to different communities, i.e. $C = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$ for $V = \{v_1, \ldots, v_n\}$. Prove that $Q(G, C) \to \frac{1}{2}$ for $n \to \infty$. [2P]
3. **Node centralities**
   (a) Construct a network in which the node with the highest betweenness centrality has the smallest degree centrality. Use `pathpy` to demonstrate the correctness of your example. [1P]
   (b) Construct a network in which exactly one node has the maximum possible closeness centrality. [1P]
   (c) Give an example for a network with 10 nodes where exactly one node has the maximum betweenness centrality possible in a network with that size. Prove that the maximum possible betweenness centrality in a network with n nodes is $n^2 - 2n - n + 2$. [2P]

[1]Richard Bellman: **On a routing problem**, In Quarterly of Applied Mathematics, No. 16, pp. 87–90

Notes:

Self-study questions

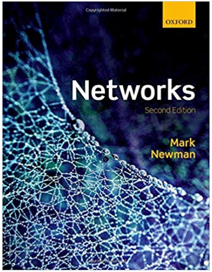
1. Why do entries of the k-th power of adjacency matrices count walks of length k?
2. Explain how we can use the sum of adjacency matrix powers $\sum_{k=1}^{l} \mathbf{A}^k$ to compute the diameter of a network.
3. Implement an algorithm that determines the modularity Q_{opt} for a given network.
4. Explain the difference between $Q_{max}(G, C)$ and $Q_{opt}(G, C)$.
5. Give an example for a network with maximum community assortativity coefficient but modularity smaller than one.
6. Define the betweenness and closeness centrality in an undirected network.
7. Construct a network in which the node with highest betweenness centrality is the one with the smallest degree centrality?
8. For a fully connected network with n nodes and no self-loops and a single community C_1, show that $Q \to 0 (n \to \infty)$.
9. For a network with n nodes, only connected by self-loops (represented by 1-elements on the diagonal) and each node i assigned to its own community C_i, show that $Q \to \frac{1}{2} (n \to \infty)$.

Notes:

References

reading list

- M Newman: **Networks**, Oxford University Press, 2010
  → Chapters 6 & 11
- D Easley, J Kleinberg: **Networks, Crowds, and Markets: Reasoning about a highly interconnected world**, Cambridge University Press, 2010 → Chapter 2
- V Latora, V Nicosia, G Russo: **Complex Networks: Principles, Methods, and Applications**, Cambridge University Press, 2017 → Chapters 1 & 9
- L Freeman: **A set of measures of centrality based on betweenness**, Sociometry, 1977
- U Brandes: **A faster algorithm for betweenness centrality**, Journal of Mathematical Sociology, 2001
- A Bavelas: **Communication patterns in task-oriented groups** Journal of the Acoustical Society of America, 1950
- G Sabidussi: **The centrality index of a graph**. Psychometrika, 1966
- M Newman: **Modularity and community structure in networks**, PNAS, 2006
- U Brandes et al.: **On Modularity Clustering**, IEEE Transactions on Knowledge and Data Engineering, 2008



Notes: