1 Theory

This chapter is devoted to the mathematical formalism that is used to model infectious diseases and networks. The author defines a mathematical framework and summarizes relevant results of earlier research in this area. In addition, section xx describes an efficient computer implementation of networks.

1.1 Models of infectious diseases

Observations. Large scale patterns of epidemics have been measured [**?**]. The spread of infectious diseases is something that everyone is familiar with.

Research field: Epidemiology. One goal of epidemiology is to understand the principles behind the spreading process, i.e. the way how a disease is transmitted through a population. In this context, *conceptional* models are used. They make use of simple assumptions for the local (person-to-person) dynamics and focus on the big picture of the process. Conceptual models are very similar to models in theoretical physics, because they focus on the very essence of the problem (here: the macroscopic view, spreading patterns). However, they have to neglect many details of the real problem (here: physiology, symptoms, individual behavior, infection pathways and many more!) in order to have mathematical feasible models.

Another important issue of epidemiology is the *forecast* of epidemic spreading processes. Forecast models incorporate as much information as possible and the main focus is not an understanding of the basic principles.

This section summarizes the mathematical framework that roughly reproduces the behavior of infectious diseases and briefly discusses some major insights.

1.1.1 Development of mathematical epidemiology

The modeling of infection diseases mostly uses the concept of compartment models as explained in section xx. Major contributions to the modern theoretical framework were provided by [**?**], [**?**] and [**?**]. In his review about the mathematics of infectious diseases Hethcote reports a model for smallpox was already formulated in 1760 by D. Bernoulli ([**?**] and references therein). In the early 20th century, people developed mathematical models for epidemics: a discrete time model in 1906 [**?**] and a differential equation model in 1911 [**?**]. The epidemic threshold (section 1.1.4) was found in the 1920s [**?**]

[?]. Starting from Bailey's book [?] in the 1950s, the modeling of infectious diseases became a major scientific research field. Modern models of infectious diseases include vaccination, demographic structure, disease vectors, quarantine and even game theory ([?] and references in [?]). The availability of contact data in recent years led to a strong impact on network analysis on epidemiology. Well known concepts of mathematics (graph theory [?]) and social sciences (social network analysis [?]) have been adopted to disease modeling, since the connections between individuals are related to their epidemic spreading potential [?].

1.1.2 Infection dynamics

The spread of infectious diseases can be modeled in terms of compartment models as described in section xx. We differentiate between *conceptional models* and *realistic disease models*. While the former class is used to provide conceptual results as for the computation of thresholds or to test theories [?], realistic disease models use as many aspects as possible to provide a forecast of the spreading process. Realistic disease models can be very complex and are beyond the scope of this work, thus the author focuses on the use of conceptional models. The following section is inspired by the Lecture notes of J. R. Chasnov [?].

1.1.3 SI model

Let us consider a population of N individuals. In the simplest case, the infection status of each individual is either susceptible or infected and there are no births and deaths on the population. Susceptible individuals become infected, if they are in contact with an infected. This mimics the behavior of an infectious disease without immunization, i.e. infected individuals stay permanently infected.

Provided that α is the rate, under which new susceptible become infected, the SI-model is as follows:

$$\begin{aligned} \frac{dS}{dt} &= -\alpha SI \\ \frac{dI}{dt} &= \alpha SI, \end{aligned} \tag{1.1}$$

where S and I are the numbers of susceptible and infected individuals respectively. The total population is $N = S + I$. Thus, (1.1) can be rewritten as

$$\frac{dI}{dt} = \alpha(N - I)I,$$

i.e. a logistic differential equation. Hence, in the limit $t \to \infty$ the whole population is infected ($I(\infty) = N$).

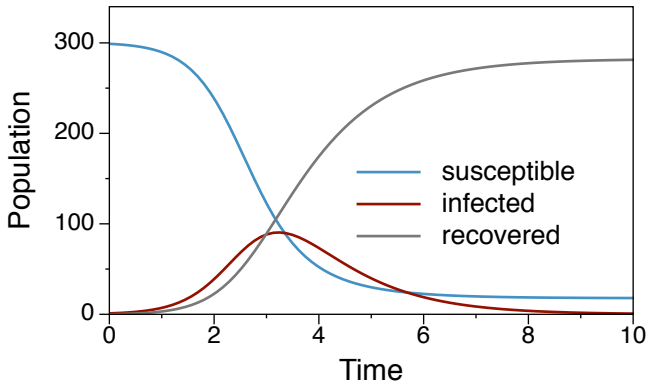


Figure 1.1. Solution of the susceptible-infected-recovered (SIR) model (1.2). The number of infected shows that the spreading process is a single event. Note that a fraction of the population is still susceptible at the end of the process. Parameters: $\alpha = 3$, $\gamma = 1$, $N = 300$, $S_0 = 1$.

1.1.4 SIR model

In contrast of the infection dynamics introduced in the previous section, many epidemics allow for an immunization of the individuals. Examples are measles or whooping cough [**?**] [**?**]. In this case, individuals recover from the disease after being infected for a certain time period. The infection scheme has to be extended to susceptible-infected-recovered (SIR) as in the following infection model [**?**]:

$$\begin{aligned} \frac{dS}{dt} &= -\alpha SI \\ \frac{dI}{dt} &= \alpha SI - \gamma I \\ \frac{dR}{dt} &= \gamma I \end{aligned} \tag{1.2}$$

where α is the infection rate and γ is the immunization or recovery rate. There is no analytic solution for the system (1.2), but some fundamental conclusions can be obtained analytically. We show a typical solution of (1.2) in figure 1.1.

The SIR model shows more sophisticated features than the SI model (1.1). To begin with, we analyze the fixed points of the system, i.e. (S_*, I_*, R_*) where

$$\frac{dS_*}{dt} = -\alpha S_* I_* = 0, \quad \frac{dI_*}{dt} = \alpha S_* I_* - \gamma I_* = 0, \quad \frac{dR_*}{dt} = \gamma I_* = 0. \tag{1.3}$$

It follows from the last equation that $I_* = 0$ at the fixed point, where S_* and R_* can be arbitrary. Hence, a fixed point is $(S_*, 0, R_*)$.

Let us analyze the stability of the fixed point in the early phase of an infection. Almost all individuals are susceptible and consequently $I_* = N - S_*$. An outbreak occurs, if and only if $dI/dt > 0$ in this phase, i.e.

$$\frac{dI}{dt} = \alpha S_*(N - S_*) - \gamma(N - S_*) = (N - S_*)(\alpha S_* - \gamma) > 0. \tag{1.4}$$

It follows from (1.4) that the number of infected grows, if

$$\alpha S_*/\gamma > 1. \tag{1.5}$$

Equation (1.5) is extremely important in epidemiology, because it defines a threshold for the unfolding of an infection spreading process. We call this fraction the *basic reproduction number* R_0. Recall that $S_* \approx N$ in the fixed point. Thus it follows that the outbreak condition is

$$R_0 = N\frac{\alpha}{\gamma} > 1. \tag{1.6}$$

The basic reproduction number describes the average number of follow-up infections by each infected individual. It is one of the main goals in epidemiology to bring down the basic reproduction number of a disease below the critical value $R_0 = 1$. This is the reason for the implementation of mass vaccination. As one can immediately see from equation (1.6), this can be done by reducing the infection rate α or by increasing the immunization rate γ. In principle, one could also reduce the size of the initial population S_*. As an example, reducing the infection rate can be done by increasing hygiene standards or appropriate behavior, say wearing warm clothes in winter time to avoid common cold. The immunization rate can be increased by vaccination.

Let us now focus on the late phase of an SIR-infection. In contrast to the SI-model of section 1.1.3 an SIR like outbreak does not necessarily infect the whole population, even if $R_0 > 1$. The reason is that there has to be a critical mass of susceptible individuals in order to keep an infection alive (see equation (1.5)). The total number of infected during an infection given by the number of recovered at the end of the infection, since every recovered has to be in the infected state in the first place. A central measure throughout this work is therefore the *outbreak size* R_∞.

To compute the outbreak size, we consider the second fixed point of (1.2), i.e. the fixed point for $t \to \infty$. At this point there are no infected and a fraction of the population is recovered. Hence the fixed point is $(N - R_\infty, 0, R_\infty)$. A simple way to obtain the outbreak size R_∞ is to use equations (1.2) and compute

$$\frac{dS}{dR} = -\frac{\alpha}{\gamma}S$$

and separate the variables [**?**]. This yields

$$\int_{S_*}^{N-R_\infty} \frac{dS}{S} = -\frac{\alpha}{\gamma}\int_{R_*}^{R_\infty} dR.$$

We integrate from the initial condition at $t = 0$ to the final condition at $t \to \infty$, where $S_\infty = N - R_\infty$. Using that $R_* = 0$ at $t = 0$ gives

$$R_\infty = S_* - S_* e^{-\frac{\alpha}{\gamma}R_\infty}. \tag{1.7}$$

This transcendental equation can be solved numerically using a Newton-Raphson technique. The outbreak size R_∞ only takes finite values for $\alpha/\gamma > 1$. A solution of equation (1.7) is shown in figure 1.2

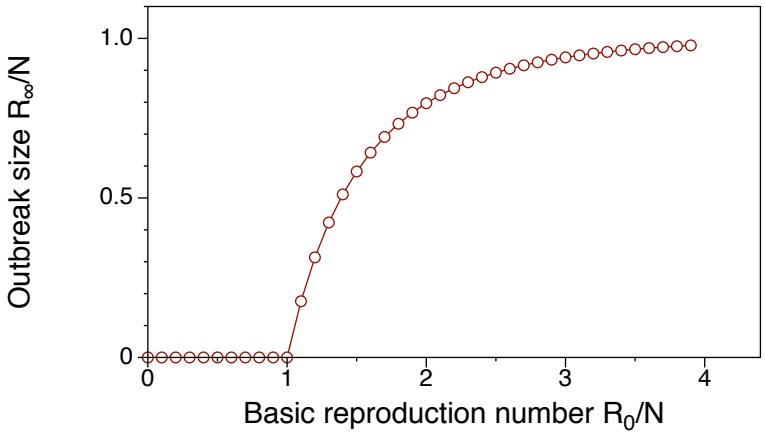


Figure 1.2. Relative outbreak size vs. basic reproduction number. The outbreak size takes finite values only for $R_0/N > 1$. Note that even for supercritical R_0 the outbreak size is in general smaller than the total population.

It should be noted that an SIR epidemic is a single event, i.e. it possesses a *characteristic time scale*. The analysis of the late phase of an epidemic also gives information about these time scales. Let us consider the second equation of (1.2).

$$\frac{dI}{dt} = \alpha SI - \gamma I \tag{1.8}$$

In the late phase of an SIR-type epidemic, the fraction of infected is small. Given sufficiently large values of R_0, the fraction of recovered is also small in this phase (see figure 1.2). Thus, we neglect the quadratic term in (1.8). This gives $\frac{dI}{dt} = -\gamma I$, which has the solution

$$I(t) = I(0)e^{-\gamma t}.$$

Hence, the infection decays exponentially for large t and the inverse recovery rate $1/\gamma$ defines the characteristic time of the epidemic.

1.1.5 Force of infection

The model presented in section 1.1.4 describes only the very basic behavior of epidemic dynamics, and is therefore a conceptual model. However, it is one of the main objectives

in epidemiology to have an understanding of the exact infection rates in the process. Infection rates their selves can cause complex infection dynamics.

The term αI used in section 1.1.4 is a special, very simple case of an infection rate. More generally, we have to replace αI by an abstract infection rate λ containing more information about the interaction between susceptible and infected individuals [**?**]. Thus, the equation for the infected becomes

$$dI/dt = -\lambda S - \gamma I.$$

The rate λ is called the *force of infection*. In principle, this parameter can be arbitrarily complex, because it contains detailed information about the mixing properties of the population. This information could be given as contact networks, demographic contact structures, etc.

In most cases, detailed information about mixing is not available. Instead, we assume *random mixing* of the population, i.e. every individual can be in contact with every other individual. This yields a transmission rate [**?**]

$$\lambda = \tau n \frac{I}{N} \equiv \beta \frac{I}{N}, \tag{1.9}$$

where τ is the transmission rate, n is the effective contact rate and I/N is the fraction of infectious contacts. It is therefore reasonable to replace the infection term α in (1.2) by β/N to explicitly include the force of infection. Nevertheless, the results presented in section 1.1.4 remain qualitatively the same.

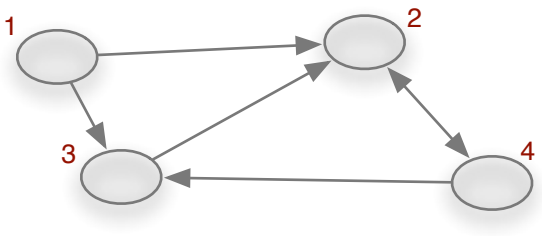
Although the force of infection gives a more reasonable description of the infection process, the assumption of random mixing remains inappropriate for many real world systems. Due to the availability of contact data, the random mixing assumption can be improved in terms of contact networks. Even if the exact data of an epidemic system is not available, research on complex networks allows us to give more realistic models about mixing. In the next section, we briefly report important results in complex network research and focus on the interplay between networks and epidemics in section xx.

1.2 Network theory

As we have seen in the previous section, standard epidemic models make use of the random mixing assumption. This assumption seems reasonable, if no further information about the contact structure within a population is available, because it gives a worst case scenario of the infection dynamics. Even an overestimation of the outbreak size can be corrected by introducing smaller, effective disease parameters. However, the random mixing assumption does not allow for non homogenous mixing, i.e. each individual is considered equal. Nevertheless, the equality of individuals is not a reasonable assumption for many epidemic substrates. Examples are contact structures of humans, livestock trade, vehicles as disease vectors of links between computers.

Figure 1.3. A simple directed network. The corresponding adjacency matrix is

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$



1.2.1 Network matrix representations

A network is a system consisting of nodes that are connected by edges. Edges can be undirected, directed and weighted. In principle, a network can consist of edges of different types. This can be represented by multiple networks sharing the same set of nodes, but different edges.

Networks are called graphs in mathematical literature. A graph $G = (V, E)$ is a set of nodes (or vertices) V and edges (or arcs) E, where each edge is given by the tuple of nodes it connects, i.e. $e_1 = (u, v) \in E$ connects nodes u and v. An edge (u, v) being present in an undirected network implies an edge (v, u). Apparently, this does not hold in directed networks. Edges of weighted networks carry additional meta information about their weight. This meta information can be their importance, capacity, number of transported items or the geographical distance between nodes u and v.

Graphs can be represented by different graph matrices, where different matrix representation emphasize typical properties of the network. The most common graph matrix is the *adjacency matrix* $\mathbf{A}$ with entries

$$a_{ij} \equiv (\mathbf{A})_{ij} = \begin{cases} 1 & \text{if } i \text{ is connected to } j \\ 0 & \text{else.} \end{cases} \tag{1.10}$$

An adjacency matrix contains the edges of the graph and can be seen of the most fundamental graph representation. Figure 1.3 shows a simple example of a directed graph and its adjacency matrix. The corresponding matrix would be symmetric in the undirected case. Weighted networks can be represented by weight matrices, where the values of the entries in (1.10) are not restricted to zero and one.

The adjacency matrix of an undirected network is symmetric, because every non-zero entry $a_{ij} = 1$ implies an edge into the opposite direction, $a_{ji} = 1$. Entries on the main diagonal a_{ii} correspond to nodes with self loops, i.e. nodes with edges pointing back to themselves. The i-th row the adjacency matrix contains non-zero entries $a_{ij} = 1$, whenever node i is connected to node j. Hence, every row can be interpreted as the neighborhood of one node. This holds for undirected and for directed networks. The columns of $\mathbf{A}$ give the same information as the rows. In the directed case, however, rows contain the out-neighborhood of each node and columns contain the in-neighborhood,

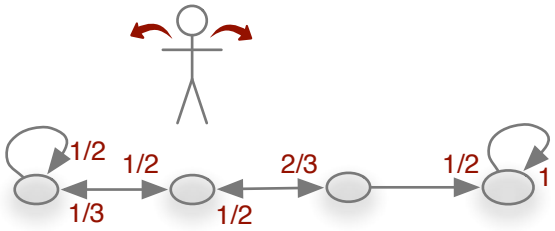


Figure 1.4. Trajectory of a toddling drunk man as an example of a Markov chain. At every location there is a probability for the drunkard to go left or right. The node rightmost node is an absorbing state and could model a park bench. Weights at arrowheads mark the transition probability. (inspired by [?]).

respectively.

Information about paths of a certain length can be obtained using the powers of the adjacency matrix. The adjacency matrix gives information about the number of paths of length 1 between node pairs. Evidently, the number of paths of length 2 between two nodes i and j is given by $(\mathbf{A}^2)_{ij}$. This applies also to paths of arbitrary length n using the elements of $\mathbf{A}^n$.

An important example for weighted network matrices is a *Markov chain*. A Markov chain is a random process without memory and with discrete state space and discrete time. It is called time-homogenous, if the transition rates are constant. Time-homogenous Markov chains can be represented as weighted networks and the corresponding weighted adjacency matrix is the *transition matrix*. Transition matrices are stochastic matrices, i.e. the elements of every row sum up to unity. Each node represents a different state of the system and the edges are weighted with the probabilities to transition into other states adjacent to these edges. It is obvious that a transition matrix representation is useful to describe random walks on networks. An example of such a process is shown in figure 1.4. The underlying network represents a line of locations, where the drunkard can be located. At every time-step there is a certain probability to move to another location. The state of the random walker can be described by a probability vector $\mathbf{p}$, where the initial state of figure 1.4 is $\mathbf{p} = (0, 1, 0, 0)$. The transition matrix $\mathbf{M}$ is a weighted adjacency matrix as it follows from the figure. Given a state $\mathbf{p}_t$ at time t, the state of the next time step is given by $\mathbf{p}_{t+1} = \mathbf{p}_t\mathbf{M}^T$. The equilibrium state $\mathbf{p}_{\text{eq}}$ follows in the limit $\lim_{n\to\infty} \mathbf{p}_0(\mathbf{M}^T)^n$, i.e. the equilibrium state is given by the dominant eigenvector of $\mathbf{M}$.

As a special case of transition matrices, the author would like to name the *Google matrix*. It describes a random walk on a network, but allows for shortcuts to any node in the network with a certain probability. The eigenvectors of Google matrices are used for the computation of node rankings according to the PageRank-Algorithm [?].

Finally, the *Laplace-matrix* of a network is an appropriate representation to model diffusion processes. For undirected networks the Laplace-matrix is defined as

$$\mathcal{L} = \mathbf{D} - \mathbf{A}, \tag{1.11}$$

where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}$ is a diagonal matrix containing the degree

$d_i = \sum_j a_{ij}$ of each node. The definition (1.11) has strong analogies to the discrete Laplace-Operator [**?**]. Consequently, they can be used to model diffusion processes on graphs in analogy to Laplace operators in continuous systems (see section xx).

The spectra of adjacency and Laplace matrices contain information about the evolution/history of networks [**?**].

1.2.2 Network measures

Before we address ourselves to models of real world networks, we have to introduce methods to measure structural properties of networks. On the micro scale, this can be done in terms of *node centrality* measures. These measures are very important to assess the importance of single nodes in the network. On the macroscopic side, we are interested in the large-scale properties of networks, i.e. percolation, distributions of centralities, connected components or other large scale structures.

Network terminology

Let $G = (V, E)$ be a graph consisting of a set of nodes V and a set of edges E. Every route across a graph along its edges without repeating nodes is called a *path*. Each path is given by an ordered set of the nodes traversed, i.e. $(v_1, v_2, \ldots, v_l)$, with $v_i \in V$ and all edges are in E, $v_i, v_{i+1} \subseteq E$. If there is a path from every node in the network to any other node, the network is called *connected*. In directed networks, we have to consider two types of connectedness. A directed network is strongly connected, if there is a directed path between all node pairs and weakly connected, if the node pairs would be connected ignoring the direction of edges.

The *distance* between two nodes is the length of the shortest path between them. Every closed path is called a *cycle*. Graphs that do not contain cycles are called *trees*. The neighborhood of a node u is the set of all nodes adjacent to it and the size of the neighborhood is the *degree* of the node. Hence, a node v is in the neighborhood of u, if $(u, v) \in E$. We distinguish between in-degree and out-degree in directed networks. Finally, $G_0 = (V_0, E_0)$ is a *subgraph* of $G = (V, E)$, if $V_0 \subseteq V$ and $E_0 \subseteq E$.

Microscopic measures

Given a network, an important question is, if some nodes are more important as other nodes. Therefore, we summarize measures of the *centrality* of nodes. The idea if centrality mainly goes back to social network analysis [**?**, **?**], but has been widely adopted and extended in network science. I restrict myself to those measures, that are indispensable when describing networks. A more exhaustive overview of centrality measures is found in the review article [**?**] or in online documentations of network analysis software, e.g. [**?**, **?**]. In the following, N denotes the order of the network (the number of nodes) and m the number of edges.

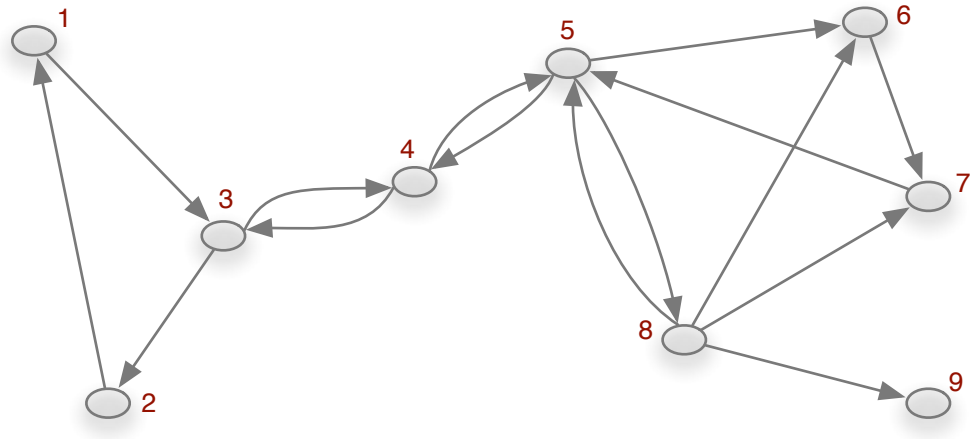


Figure 1.5. A directed network for the demonstration of different centrality measures.

Degree. The simplest centrality measure is the degree k of a node, which is the number of its neighbors. In directed network, we distinguish between in-degree k^- and out-degree k^+. The degree follows immediately from the adjacency matrix, i.e.

$$k^-(i) = \sum_j a_{ji} \quad \text{and} \quad k^+(i) = \sum_j a_{ij}$$

is the in- and out-degree of node i, respectively. As an example, node 8 in figure 1.5 has $k^+(8) = 4$ and $k^-(8) = 1$. In weighted networks, the degree is computed in the same way and is called in-weight and out-weight of a node.

The degree centrality is used in a huge variety of applications. One of its most important applications is to measure the heterogeneity of network connections, i.e. the existence of hubs in the network. Hubs are nodes with a degree much larger than the rest of the system. The heterogeneity of networks can be measured in terms of degree distributions (see section xx).

Closeness. The closeness of a node is the reciprocal average distance to all other nodes in the network. It can be normalized, so that the closeness is 1, if all other nodes are reachable within one step and 0 in the limit of infinite distances to all other nodes. The closeness of a node i in a network of order N is defined as follows:

$$c(i) = N - 1 \sum_j \frac{1}{d_{ij}} \tag{1.12}$$

where d_{ij} is the distance between nodes i and j. Some tools for an efficient computation of shortest-path distances are introduced in section 1.2.3. It should be noted that the

distance between two nodes is defined to be infinite, if the underlying network is not connected. In this case, the corresponding terms $1/\infty$ do not make contributions in equation (1.12).

The closeness centrality is capable to identify nodes with short average pathways to other parts of the network. Identifying high closeness nodes is therefore reasonable for network navigation. This holds in particular, if the exact route to the destination is unknown, because nodes with high closeness are probable to reach many destinations quickly.

Betweenness. In order to identify nodes that act as bridges between two subgraphs, the measure of betweenness was developed. In figure 1.5, node 4 plays such a role. It is characteristic for these nodes to contain a relatively large number of shortest paths that have to cross them. Therefore, betweenness of a node i is defined as

$$b(i) = \sum_{s \neq i \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}} \tag{1.13}$$

where σ_{st} is the number of shortest paths between nodes s and t and $\sigma_{st}(i)$ is the number of shortest paths between s and t going through node i. The computation of betweenness is expensive using equation (1.13). Therefore, an efficient algorithm was introduced by Brandes [**?**].

Note that bridge nodes might look inconspicuous in the first place, e.g. they could have only two links. Removing node 5 in figure 1.5, for instance, would divide the network into two disjoint subgraphs with nodes $V_1 = (1, 2, 3)$ and $V_2 = (5, 6, 7, 8, 9)$ respectively. Therefore, removing nodes of high betweenness from the network has been proven useful in order to divide networks into smaller components [**?**, **?**].

Eigenvector centrality. The idea of eigenvector centrality can be easily realized by considering Markov chains as in section 1.2.1. Frequent multiplication of the transition matrix $\mathbf{M}$ with a random vector gives the largest eigenvector of $\mathbf{M}$. This relation is known as power method or van Mises iteration [**?**]. The dominant eigenvector of the transition matrix gives the equilibrium state of the system. Using this state as a measure of centrality assigns every node with the probability to find a random walker here after a long period. The principle behind the dominant eigenvector of an adjacency matrix $\mathbf{A}$ is that important nodes are likely to be connected to other important nodes. This recursive concept is reflected in the equation

$$x_i = \frac{1}{\lambda} \sum_j a_{ij} x_j,$$

where x_i is the centrality of i, $\sum_j a_{ij}x_j$ is the centrality of the neighborhood of i and λ is a constant. This equation can be rewritten as

$$\mathbf{Ax} = \lambda\mathbf{x}. \tag{1.14}$$

It follows from the Perron-Frobenius-Theorem that λ must be the largest eigenvalue of $\mathbf{A}$ in order to guaranty all entries of $\mathbf{x}$ to be positive [**?**]. The theorem guaranties unique solutions only to adjacency matrices of connected networks. Hence, eigenvector centrality is only defined for connected graphs. Nevertheless, the eigenvector centrality can be computed for each component separately, if a graph is not connected [**?**].

Some important variants of eigenvector centrality are the PageRank and HITS algorithm [**?**, **?**].

Node components and range. The component of a node is the set of nodes it is connected to by a path of any finite length. We call the size of this set the *range* of a node [**?**]. In directed networks, we distinguish between the out-component and in-component of a node. The size of the former is its range and the size of the latter is its reachability.

Apparently, the range of a node is of major importance for any epidemiological problem on a network, because it defines an upper bound for the size of any outbreak starting at this very node. Although the range measure is rather simple, it can show an interesting distribution. The shape of its distribution is inherently related to percolation properties of the network.

Macroscopic measures

In order to get the big picture about a network, I discuss measures that capture large scale properties of networks.

Degree Distribution. In principle, the distribution of any centrality measure could yield insights into the macroscopic network structure. As a matter of fact, the distribution of a networks degrees became a major criterium for the classification into different network types. If all nodes of a graph have the same degree, the graph is called *regular*. Lattices are regular graphs. In this case, the degree distribution collapses to a single peak without statistical variation. A *random network* is generated by generating a set of N nodes and connect node pairs with a certain probability p. This procedure yields a graph with a Poisson degree distribution (see section xx),

$$P(k) = \frac{(Np)^k e^{-Np}}{k!} \tag{1.15}$$

i.e. there is variation in the degrees, but there still remains a *typical degree* in the system.

Observations of real-world networks have shown that their nodes show a degree variation over several orders of magnitude. The corresponding degree distributions are called *scale-free*, because they do not show a typical degree. Instead, the network has nodes with only a few neighbors and hubs with very large degrees. These networks are called scale-free networks and their degree distribution takes the form

$$P(k) \propto ck^{-\gamma} \tag{1.16}$$

for sufficiently large values of k. c is a normalization constant and $2 < \gamma < 3$ for most observed networks [**?**]. The structural difference between random and scale-free networks is sketched in figure 1.6.

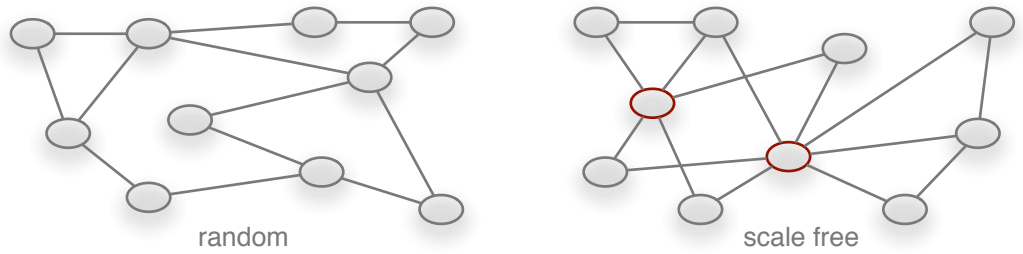


Figure 1.6. Structural difference between random networks and scale-free networks. All nodes have a similar degree in the random network, while the scale-free network shows hubs with a significantly larger degree than the average. Hubs are highlighted in red.

Scale-free networks have attained remarkable attention in the last years and many real-world networks have been conjectured as scale-free [**?**, **?**]. Important consequences of this classification were found to be a change in the threshold behavior of epidemic processes [**?**] and their topological resilience to node failures [**?**].

Clustering coefficient. The idea of the clustering coefficient comes from social networks. It measures, whether a network contains a significantly large number of triangles. This behavior is conjectured to be typical for social networks and has the simple meaning: "a friend of your friend is likely to be your friend". The clustering coefficient C is the number of connected triples ($A \to B \to C \to A$) divided by the actual number of triples ($A \to B \to C$) in the network. Using the adjacency matrix $\mathbf{A}$, the clustering coefficient can be computed as follows:

$$C = \frac{\mathrm{tr}(\mathbf{A}^3)}{\mathrm{sum}(A^2) - \mathrm{tr}(\mathbf{A}^2)}, \tag{1.17}$$

where $\mathrm{tr}(\mathbf{A})$ denotes the trace of $\mathbf{A}$ and $\mathrm{sum}(\mathbf{A}) = \sum_{ij} a_{ij}$ is the sum over all elements of $\mathbf{A}$.

The clustering coefficient plays an essential role in the small-world model of networks ([?], section 1.2.4) and has been found to be an important property of social networks [?].

Connected components. A connected component $G_{cc} = (V_{cc}, E_{cc})$ is a subgraph of graph $G = (V, E)$, where there is a path between any node pair in V_{cc}. The number of connected components lies between 1 and the number of nodes in the network.

Accessibility.

1.2.3 Implementation

In order to efficiently implement networks and their analysis on a computer, it is necessary to use data structures adjusted to these problems. A short and transparent introduction to data structures and algorithms is in the book of Skiena [?]. In this section, I discuss some essential data structures appropriate for network analysis and give a brief description of fundamental algorithms. The purpose of this section is not to list different algorithms and source code, but rather to sketch the basic ideas behind the data structures and algorithms. For source code of data structures and algorithms, the reader is encouraged to the lecture of [?].

Matrix implementation. To begin with, we consider the implementation of adjacency matrices as introduced in section 1.2.1. Matrix implementations work also for the other matrices introduced in section 1.2.1. All adjacency matrices are by definition square matrices. Their entries are either 0 or 1, and can take any floating number value for weighted networks. In this work, we neglect negative edge weights. The number of nodes in most complex network datasets is relatively large. Starting with small networks (100 nodes, conference contacts [?]), complex networks can be gigantic (0.5 billion nodes, twitter tweeds [?]). Note that the size of the adjacency matrices scales with the square of the networks size, hence large networks intractable for straightforward computer-based matrix analyses.

Nevertheless, there is one feature, that most adjacency matrices of real-world networks have in common: they are *sparse*, i.e. the vast majority of their entries are zeros[1]. Since zeros do not contribute to matrix operations as products or additions, it is reasonable to use data structures ignoring zeros. These data structures are called `sparse matrices`. Their advantages is (1) they save much memory and (2) computations are faster, because operations with zeros involved are not executed. Sparse matrix data structures are available in most modern computer languages (e.g. Matlab, Python: `scipy` library,

[1] Typically, the number of edges in the network is of the same order as the number of nodes.

C/C++: `boost` library). They perform well for all problems based on adjacency matrices, e.g. degree or eigenvector centrality. However, matrix methods are not suitable for the computation of many other network measures, such as betweenness, closeness or network navigation.

Graph implementation. The weakness of matrix representations of networks is that it is rather complicated to *traverse* a network using matrices. A traversal is a procedure like: start at a node, visit all of its neighbors, from each neighbor visit its neighbor and so forth, until there are no more new nodes to traverse. This is a searching process. These processes are used in many implementations of graph theoretic methods.

An alternative implementation to the adjacency matrix is an *adjacency list*. It stores the neighbors of every node and is implemented in terms of a linked list. Using the example network of 1.3, we get the following adjacency list:

$$
\begin{aligned}
&1 \to 2, 3\\
&2 \to 4\\
&3 \to 2\\
&4 \to 2, 3.
\end{aligned}
$$

In order to traverse the graph starting at node 1, we can choose any of the neighbors of 1 and repeat the process until we have traversed all nodes. One possible traversal starting at 1 would be $1 \to 3 \to 2 \to 4$.

During a traversal process, one can decide to either exploit the whole neighborhood of a node first and then traverse the next generation or choose a neighbor of every traversed node at every step. These two essential searching processes are called breadth-first-search (BFS) and depth-first-search (DFS), respectively. The difference between the two lies in the order of traversed nodes. Figure 1.7 shows resulting search trees of the two methods. Starting at node 1, the traversal $1 \to 3 \to 2 \to 4$ would be found using a DFS-search, while a BFS-search would yield $1 \to 2 \to 3 \to 4$. It should be noted that in general there exist multiple BFS and DFS trees for each starting node.

Both search algorithms are used in many applications. BFS is efficient to compute shortest paths in unweighted networks. With every generation in a BFS tree, the distance from the starting node is incremented by 1, and thus the set of nodes with a certain distance from the starting node can be directly read from the BFS tree (see figure 1.7). Shortest paths in weighted networks can be identified using a the algorithm of Dijkstra [**?**]. DFS can be used to identify connected components in directed graphs (see next section).

Implementations of graph structures as discussed above are for example available in the libraries `networks` (Python), `igraph` (C, Python, R), `Lemon` and `Boost` (C++).

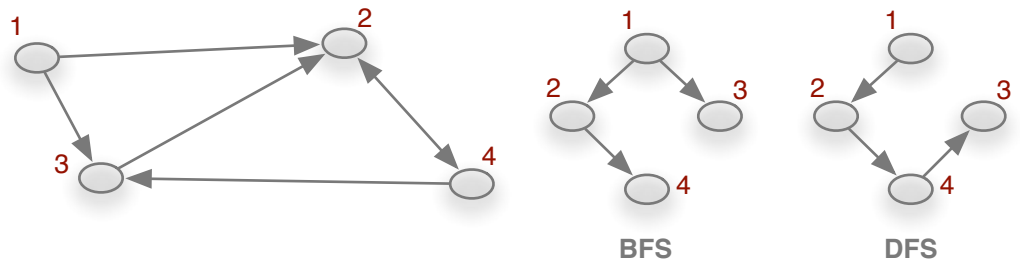


Figure 1.7. Breadth-first-search and depth-first-search trees in the directed network of fig. 1.3. Searches are started at node 1.

1.2.4 From random mixing to structure: Network models