13 Network Models

Nadine Baumann and Sebastian Stiller

The starting point in network analysis is not primarily the mathematically defined object of a graph, but rather almost everything that in ordinary language is called 'network'. These networks that occur in biology, computer science, economy, physics, or in ordinary life belong to what is often called 'the real world'. To find suitable models for the real world is the primary goal here. The analyzed real-world networks mostly fall into three categories.

The biggest fraction of research work is devoted to the Internet, the WWW, and related networks. The HTML-pages and their links (WWW), the newsgroups and messages posted to two or more of them (USENET), the routers and their physical connections, the autonomous systems, and several more are examples from this scope of interest.

In biology, in particular chemical biology, including genetics, researchers encounter numerous structures that can be interpreted as networks. Some of these show their net structure directly, at least under a microscope. But some of the most notorious of the biological networks, namely the metabolic networks, are formed a little more subtly. Here the vertices model certain molecules, and edges represent chemical reactions between these molecules in the metabolism of a certain organism. In the simplest case, two vertices are connected if there is a reaction between those molecules.

Sociological networks often appear without scientific help. We think of cronyism and other (usually malign) networks in politics and economy, we enjoy to be part of a circle of friends, we get lost in the net of administration, and networking has become a publicly acknowledged sport. The trouble – not only but also – for scientists is to get the exact data. How can we collect the data of a simple acquaintance network for a whole country, or even a bigger city? But for some networks the data is available in electronic form. For example, the collaboration of actors in movies, and the co-authorship and the citation in some research communities, partly owe their scientific attraction to the availability of the data.

Many – but not all – of these examples from different areas have some characteristics in common. For example metabolics, the WWW, and co-authorship often form networks that have very few vertices with very high degree, some of considerable degree and a huge number of vertices with very low degree. Unfortunately, the data is sometimes forced to fit into that shape, or even mischievously interpreted to show a so called power law. Often deeper results are

U. Brandes and T. Erlebach (Eds.): Network Analysis, LNCS 3418, pp. 341–372, 2005.

not only presented without proof, but also only based on so called experimental observations.

Yet one feature can be regarded as prevalent without any alchemy: Most of the real-world networks are intrinsically historical. They did not come into being as a complete and fixed structure at one single moment in time, but they have developed step by step. They emerged. Therefore, on the one hand, it makes sense to understand the current structure as the result of a process. On the other hand, one is often more interested in the network's future than in one of its single states. Therefore several models have been developed that define a graph, or a family of graphs, via a process in the course of which they emerge.

The mathematical models for evolving networks are developed for three main intentions. First of all, the model should meet or advocate a certain intuition about the nature of the development of the real-world network. Secondly, the model should be mathematically analyzable. A third objective is to find a model that is well suited for computational purpose, i.e., to simulate the future development or generate synthetic instances resembling the real network, for example to test an algorithm.

There are several overviews in particular on models for Internet and WWW networks (see [164, 68] for a more mathematically inclined overview). Some of these papers already exceed this chapter in length. It hardly pays and it is virtually impossible to mention all models, experimental conjectures, and results. We rather intend to endow the reader with the necessary knowledge to spark her own research. We proceed in four sections.

In the first section the founding questions, driving ideas, and predominant models are summarized. Then, in the second section, we compile some methods that are deemed to or have proven to be fruitful in analyzing the structure of a network as a whole. Third, we broaden our scope in order to exemplify the great variety of models for evolving networks. The last section is devoted to the state of the art generators for networks that resemble the Internet.

Up to further notice we consider graphs as directed graphs. Some graph processes may generate multigraphs which should be clear from the context.

13.1 Fundamental Models

13.1.1 The Graph Model ($G_{n,p}$)

First we want to discuss the historical starting point of random graph theory. More precisely, we define the graph model $(G_{n,p})$.

A graph model is a set of graphs endowed with a probability distribution. In this case the graphs under consideration are undirected. The following three graph models stochastically converge to each other as $n \to \infty$:

1. The first way to generate a random graph is to choose a graph uniformly at random among all graphs of given vertex number n and average vertex degree z.

2. Alternatively, choose every edge in a complete graph of n vertices with probability p to be part of $E(G)$, where $\frac{2p\binom{n}{2}}{n} = p(n-1) =: z$ is the expected average degree. This model is denoted by $(G_{n,p})$.
3. In the third method, n vertices v_i are added successively, deciding for each v_i and for each $j < i$ whether to put $\{v_i, v_j\}$ in the edge set or not with probability p.

The last one is an interpretation of the second as a graph process. See Section 13.1.4 for more details about graph processes. The first is of course more restrictive, because the average degree is fixed and not just expected, as in the two other models. Still these models converge. Thereby the first model may be more intuitive, but the second is often more suitable for analysis. These three aspects are also important for the other models we will discuss in this section: Some models capture best our intuition about the real world, others are superior in mathematical tractability. Third, networks in the real world very often are structures which rather emerged from a process than popped up as a whole.

There is a myriad of literature and highly developed theory on the $(G_{n,p})$ and related models. It turns out that a graph chosen according to that distribution, a graph 'generated' by that model, shows a number of interesting characteristics with high probability. On the other hand, this graph model has, precisely because of these characteristics, often been disqualified as a model for real-world networks that usually do not show these characteristics. For example, without deep mathematical consideration one can see that the majority of the vertices will have almost or exactly the average degree. For many networks in the real world this is not the case. Still our interest in this model is more than historical.

We should state at least one fundamental and very illuminating result on $(G_{n,p})$-graphs. Let $G_{n,p}$ denote a fixed graph generated by one of these models.

Theorem 13.1.1. *Let m_ω be the expected number of arcs in a $G_{n,p}$, i.e., $m_\omega = p\binom{n}{2}$. If $m_\omega = \frac{n}{2}(\log n + \omega(n))$, then for*
$\omega \to -\infty$ $G_{n,p}$ is disconnected with high probability, and for
$\omega \to \infty$ $G_{n,p}$ is connected with high probability.

This chapter will extensively treat degree sequences. Therefore we state the following immediate fact about the probability distribution p of the degree k of a vertex in a $(G_{n,p})$-graph. We use z or z_1 to denote the average degree of a vertex in the graph under consideration.

$$p(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k} \approx \frac{z^k \exp(-z)}{k!}$$

After this classical mathematical model let us turn to a topic strongly inspired by the real-world, the concept of a Small World.

13.1.2 Small World

One of the starting points of network analysis is a sociological experiment conducted to verify the urban legend that anyone indirectly knows each other by

just a few other mediators. To scrutinize this assumption Milgram [421] asked several people in the US to deliver a message just by passing it on to people they knew personally. The senders and mediators knew nothing about the recipient but his name, profession, and the town he lived in. These messages reached their destination on average after roughly five or six mediators, justifying the popular claim of six degrees of vicinity. The world, at least in the US, appears to be small.

The notion of 'Small World' has become technical since, usually encompassing two characteristics: First, the average shortest path distances over all vertices in a small world network has to be small. 'Small' is conceptualized as growing at most logarithmically with the number of vertices. In this sense $(G_{n,p})$ graphs (see Section 13.1.1) are small even for small values of p, and the sociological observation would come as no surprise. But in a vicinity-network – like the one the sociological experiment was conducted on – a huge fraction of people one knows personally, also know each other personally. Mathematically speaking a network shows the worldly aspect of a small world if it has a high clustering coefficient. Whereas in an $(G_{n,p})$ graph the clustering coefficient obviously tends to zero. (The clustering coefficient gives the fraction of pairs of neighbors of a vertex that are adjacent, averaged over all vertices of the graph. For a precise definition of the clustering coefficient and a related graph statistic, called transitivity, see 11.5.)

A very popular abstract model of small world networks, i.e., a graph with clustering coefficient bounded from below by a constant and logarithmically growing average path distance, is obtained by a simple rewiring procedure. Start with the kth power of an n-cycle, denoted by C_n^k. The kth power of a cycle is a graph where each vertex is not only adjacent to its direct neighbors but also to its k neighbors to the right and k neighbors to the left. Decide for each edge independently by a given probability p whether to keep it in place or to rewire it, i.e., to replace the edge $\{a, b\}$ by an edge $\{a, c\}$ where c is chosen uniformly at random from the vertex set.

The description contains a harmless ambiguity. Viewing the rewiring process as iteratively passing through all vertices, one may choose an edge to be rewired from both of its vertices. It is not a priori clear how to handle these ties. The natural way to straighten this out is the following: Visit each vertex iteratively in some order, and make the rewiring decisions for each of the *currently* incident edges. Therefore, strictly speaking, the model depends on the order in which the vertex set is traversed. Anyway, the reader should be confident that this does not affect the outcome we are interested in, namely the average shortest path distance and the clustering coefficient C. For small p the clustering coefficient stays virtually that of C_n^k. To be more precise, for small k and p and large n: $C(G_{rewired}) = C(C_n^k)(1 - \frac{p}{2k})$, as the pth fraction of an average of the $2k$ neighbors' contribution is removed from the numerator. On the other hand, the average path distance in such a graph decreases quickly (as p increases) from the original $\frac{n}{4k}$ (on average one has to walk a quarter of the circle by steps of length

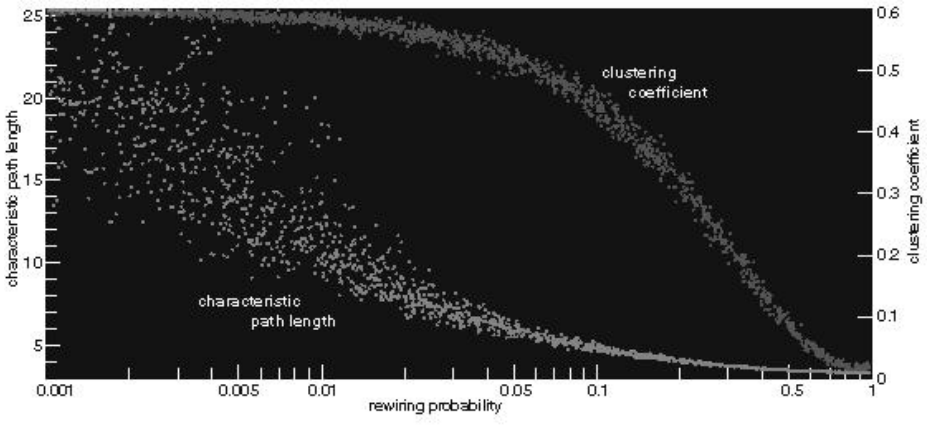

Fig. 13.1. Clustering coefficient and path lengths for the small world model by Watts-Strogats. Found at: `http://backspaces.net/PLaw/`. Results are from 2,000 random graphs, each with 300 vertices and 900 edges

k, except for maybe the last) to small values, claimed [573] to be in $\mathcal{O}(\log n)$ (compare Figure 13.1).

Unfortunately, these figures were obtained and verified empirically only. The chart suggests that calculation of the second moment of the distributions would be desirable, as the lower cloud of points, i.e., the average shortest path distances, appear far less stable. Maybe the most important problem with such experimental figures is that they can hardly account for the difference between, for example, a logarithmic or a $\sqrt{n}$ behavior.

A weakness of the rewiring model, and thus of the whole definition of small world graphs, is that, by fixing the number of random edges and enlarging k, the clustering coefficient can be kept artificially high, whereas the path distances on average only depend on the number of random edges relative to n. An increase in small deterministic edges does not contribute to the average path distance, except for a constant: On average the number of steps to go from one long-range edge to the other becomes smaller only by a constant factor. Sociologically speaking, having many friends in the neighborhood brings you only a constant closer to the Dalai Lama.

13.1.3 Local Search

Revisiting the sociological experiment, one may not be satisfied that the theoretical explanation only accounts for the existence of short average shortest paths. The fact that the letters reached their destination within a few steps requires short paths not only to exist, but also to be detectable for the ignorant agents in the vicinity network. This led Kleinberg to the idea of a *local algorithm*. Roughly speaking, a local algorithm should act – for example crawl a network – step by

step without knowing the whole structure. In each step only a specific, local part of the whole data should be used to reach the current decision. A definition of local algorithm for the specific problem will be given in a moment.

The real-world vicinity network is idealized by a parameterized network model that is easily found to have a high and constant clustering coefficient. It is once again a network comprised of short deterministic and long random edges, modeling the intuition that we know our neighborhood and have some occasional acquaintances. The aim is to determine the parameters under which there exists a local algorithm capable of finding a path with on average logarithmic length for a randomly chosen pair of vertices.

Model for Local Search. The network $G(V, E)$ is parameterized by n, p, q and r. The vertex set V contains the points of a 2-dimensional $n \times n$ lattice. On the one hand, E contains bi-directed arcs between each vertex and its $2p$ closest horizontal and $2p$ closest vertical neighbors. On the other hand, for each vertex, v, there are q directed arcs of the form $(v, x) \in E$, where x is chosen out of $V \setminus \{v\}$ according to the distribution $p(x) = \frac{d^{-r}(v,x)}{\sum_y d^{-r}(v,y)}$, where $d(x, y)$ denotes the minimum number of steps to go from x to y on the grid and $r > 0$ is a constant. We call such a network $G_K(n, p, q, r)$ a Kleinberg-Grid. (Note that for $p = 1$ the clustering coefficient is 0, but for $p > 1$ it is greater than 0 and essentially independent of n.)

Local Algorithm. The following notion of a local algorithm is not very general, but rather tailor-made for the above model. A local algorithm provides a rule giving the subsequent vertex at each vertex of the path to be output in the end, based only on the following types of information:

- Global Knowledge
  - The structure of the underlying grid.
  - The position of the destination vertex in the underlying grid.
- Local Knowledge
  - The positions of the current vertex in the underlying grid and of its neighbors in the whole network (i.e., including its long-range connections).
  - The positions of all vertices visited so far, including their neighbors positions.

Results. The local algorithm Kleinberg analyses is the most natural one – which gives even more explanatory power for the sociological experiment: Every recipient of the message passes it on to that vertex among its neighbors that is closest to the destination in $d(\cdot, \cdot)$. Call this the Kleinberg-Algorithm.

Theorem 13.1.2. *Let $p, q \in \mathbb{N}$ be fixed. Then the following holds for every Kleinberg-Grid $G_K(n, p, q, r)$:*

For $r = 0$
every local algorithm finds paths of average length *in $\Omega(n^{\frac{2}{3}})$.*
For $0 < r < 2$
every local algorithm finds paths of average length *in $\Omega(n^{\frac{2-r}{3}})$.*
For $r = 2$
the Kleinberg-Algorithm finds paths of average length *in $\mathcal{O}(\log^2 n)$.*
For $r > 2$
every local algorithm finds paths of average length *in $\Omega(n^{\frac{r-2}{r-1}})$.*

Sketch of Proof. Though the negative results of Theorem 13.1.2 (that no local algorithm can find a path of the desired length) are the intriguing ones, it is the proof of the positive result that will give us the required insight, and will be sketched here.

In order to estimate the average number of steps which the Kleinberg-Algorithm takes (for $r = 2$ and x the destination vertex) subdivide the vertex space in subsets U_k of vertices v with $2^{k-1} \leq d(x, v) < 2^k$. The algorithm always proceeds to a vertex that is closer to x than the current vertex. Thus, if it once reaches a subset U_i it will only advance to U_j where $j \leq i$. As the total number of subsets grows logarithmically with n, we are done if we can show that the algorithm needs at most a constant number of steps to leave a subset U_k, independent of k.

As the subset U_k can be very big, we interpret leaving a subset as finding a vertex that has a random edge into $\bigcup_{i<k} U_i$. As the algorithm visits every vertex at most once, we can apply the technique of postponed decisions, i.e., choose the random edge of a vertex v when we reach v. In order to have a constant probability at every level k, the probability for v to have a random contact at distance less than or equal to 2^{k-1} from v, must be constant for all k. This is true for a 2-dimensional lattice if and only if $r = 2$. □

At this point the result of Theorem 13.1.2 seems generalizable to other dimensions, where r should always equal the dimension. This can easily be seen for dimension 1. The details of the proof and the negative results may be more difficult to generalize.

The above proof already gives a hint why the negative results hold for dimension 2. If $r > 2$ the random arcs are on average too short to reach the next section in a constant time, when the algorithm is in a big and far away subset. On the other hand, $r < 2$ distributes too much of the probabilistic mass on long reaching arcs. The algorithm will encounter lots of arcs that bring it far beyond the target, but too rarely one that takes it to a subset closer to the target.

In general, the distribution must pay sufficient respect to the underlying grid structure to allow for a local algorithm to make use of the random arcs, but still need to be 'far-reaching' enough. It seems worthwhile to conduct such an analysis on other, more life-like, models for the vicinity network.

13.1.4 Power Law Models

As already described in Section 11.1 there is a wide interest in finding graphs where the fraction of vertices of a specified degree k follows a power law. That means that the degree distribution p is of the form

$$p(k) = ck^{-\delta} \qquad \delta > 0, c > 0.$$

This mirrors a distribution where most of the vertices have a small degree, some vertices have a medium degree, and only very few vertices have very high degree.

Power laws have not only been observed for degree distributions but also for other graph properties. The following dependencies (according to [197]) can especially be found in the Internet topology:

1. Degree of vertex as a function of the rank, i.e., the position of the vertex in a sorted list of vertex degrees in decreasing order
2. Number of vertex pairs within a neighborhood as a function of the neighborhood size (in hops)
3. Eigenvalues of the adjacency matrix as a function of the rank

A more detailed view to these power laws found in Internet topologies is given in Section 13.4. Since in the literature the most interesting fact seems to be the degree distribution, or equivalently the number of vertices that have a certain degree k, we will focus mostly on this.

In some contexts (protein networks, e-mail networks, etc.) we can observe an additional factor q^k to the power law with $0 < q < 1$ – the so called exponential cutoff (for details see [448]). Trying to fit a degree distribution to this special form, the power law $p(k) = ck^{-\delta}q^k$ obtains a lower exponent δ than would be attained otherwise. A power law with an exponential cutoff allows to normalize the distribution even in the case that the exponent δ lies in $(0, 2]$.

Since the 'strict' power law, i.e., in the form without cutoff, is more fundamental and more explicit in a mathematical way, we will in the following restrict ourselves to some models that construct networks with power laws not considering exponential cutoff. We start by describing the most well-known preferential attachment model and then give some modifications of this and other models.

Preferential Attachment Graphs. In many real life networks we can observe two important facts: *growth* and *preferential attachment*. Growth happens because networks like the WWW, friendships, etc. grow with time. Every day more web sites go online, and someone finds new friends.

An often made observation in nature is that some already highly connected vertices are likely to become even more connected than vertices with small degree. It is more likely that a new website also inserts a link to a well-known website like *google* than to some private homepage. One could argue that someone who already has a lot of friends easily gets more new friends than someone with only a few friends – the so called 'the rich get richer'-phenomenon. This is modeled by a preferential attachment rule.

One of the first models to tackle these two special characteristics is the preferential attachment model presented by Barabási and Albert in [40].

Graph Process. Formally speaking a graph process $(\mathcal{G}^t)$ is a sequence of sets $\mathcal{G}^t$ of graphs (called states of the process $(\mathcal{G}^t)$) each endowed with a probability distribution. Thereby the sets and their distributions are defined recursively by some rule of evolution. More intuitively one thinks of a graph process as the different ways in which a graph can develop over the time states.

In [40] a graph process (G_m^t) is described in this intuitive way as the history of a graph $G = (V, E)$. At every point in time one vertex v with outdegree m is added to the graph G. Each of its outgoing edges connects to some vertex $i \in V$ chosen by a probability distribution proportional to the current degree or indegree of i.

Formally, this description gives a rule how any graph of a certain state of the process is transformed into a graph of the next state. Further, this rule of evolution prescribes for any graph of a state of the graph process the probabilities with which it transforms into a certain graph of the next state. In this way, the sets and distributions of the graph process are recursively defined. Unfortunately, the above description from [40] entails some significant imprecisions which we will discuss now.

The choice of the initial state (which usually contains exactly one graph) is a nonnegligible matter. For example, taking $m = 1$, if the graph is disconnected at the beginning of the sequence then any emerging graph is also disconnected. In contrast, any connected graph stays connected. Moreover, we need at least one vertex to which the m new edges can connect. But it is not defined how to connect to a vertex without any edge, since its probability is zero. Thus, there must be at least one loop at that vertex, or some other rule how to connect to this vertex.

Secondly, one has to spell out that the distribution shall be proportional to the degree. In particular, it has to be clear whether and how the new vertex is already part of the vertex set V. If it is excluded no loops can occur. (Note that for $m = 1$ loops are the only elementary cycles possible.) If it is an element of V it is usually counted as if it had degree m, though its edges are not yet connected to their second vertices. Moreover, if $m > 1$ one has to define a probability distribution on the set of all $\binom{|V|}{m}$ or $\binom{|V|+1}{m}$ possible ways to connect which is not sufficiently defined by requiring proportionality to the degree for each single vertex.

Note that the process (G_1^t) is equivalent to the process (G_m^t) for large t in the following sense: Starting with the process (G_1^{tm}) and always contracting the last m vertices after m states we get the same result as for the process (G_m^t).

With the graph process (G_1^t), the probability that an arbitrary vertex has degree k is $\Pr[k] = k^{-\delta}$, with $\delta = 3$. There are several possibilities to prove this degree distribution. Some of them, and a precise version of the model, are presented in Section 13.2.1.

Other Power-Law Models. There are more models that try to construct a graph that resembles some real process and for which the degree distribution follows a power law. One of them is the *model of initial attractiveness* by Buckley and Osthus (see [68] for details and references). Here the vertices are given a value $a \geq 1$ that describes their initial attractiveness. For example a search engine is already from the start more attractive to be linked to than a specialized webpage for scientists. So the probability that a new vertex is linked to vertex i is proportional to its indegree plus a constant initial attractiveness am.

A different approach to imitate the growing of the world wide web is the *copying model* introduced by Kleinberg and others [375] where we are given a fixed outdegree and no attractiveness. We choose a prototype vertex $v \in V$ uniformly at random out of the vertex set V of the current graph. Let v' be a new vertex inserted into the network. For all edges (v, w) for $w \in V$ edges (v', w) are inserted into the network. In a next step each edge (v', w) is retained unchanged with probability p, or becomes rewired with probability $1 - p$. This simulates the process of copying a web page almost identical to the one the user is geared to and modifying it by rewiring some links. The authors also showed that this model obtains a degree distribution following a power law. One advantage of this model is that it constructs a lot of induced bipartite subgraphs that are often observed in real web graphs (for some further details see Section 3.9.3 about *Hubs & Authorities*). But it does not account for the high clustering coefficient that is also characteristic of the webgraph.

Another model that tries to combine most of the observations made in nature, and therefore does not restrict to only one way of choosing the possibilities for a connection, is the model of Cooper and Frieze [131]. Here we first have the choice between a method NEW and a method OLD, which we choose by a probability distribution α. Method OLD inserts a number of edges starting at a vertex already in the network whereas method NEW inserts first a new vertex and then connects a number of edges to this new vertex. The number of inserted edges is chosen according to probability distribution β. The end vertices to which to connect the edges are chosen either uniformly at random, or depending on the current vertex degree, or by a mixture of both.

13.2 Global Structure Analysis

13.2.1 Finding Power Laws of the Degree Distribution

We would like to have some general approaches to find the exact degree distribution of a given graph model. Since there are no known general methods we will present four different ways of showing the degree distribution of the preferential attachment model. One will be a static representation of one state of the graph process called Linearized Chord Diagrams, introduced by Bollobás [68]. Furthermore we will give three heuristic approaches that yield the same result.

Linearized Chord Diagrams. A Linearized Chord Diagram (LCD) consists of $2n$ distinct points on the x-axis paired off by chords in the upper half-plane.

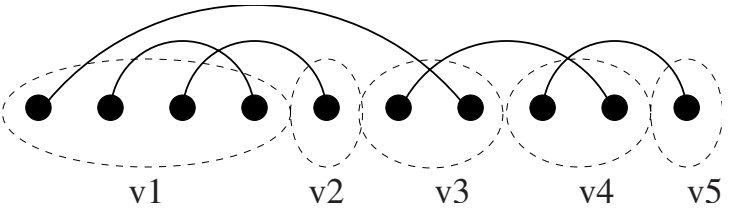

Fig. 13.2. An LCD representing a graph

The goal is now to construct a graph out of this LCD that represents a static state of a graph process.

Reconsider the preferential attachment model by Barabási and Albert (Section 13.1.4). There a graph process (G_m^t) is used. Let us consider the case $m = 1$. Let $\Pr[v]$ be the probability that the vertex v_t inserted at time t is connected to vertex v. We define

$$\Pr[v] = \begin{cases} 1/(2t-1) & \text{if } v = v_t, \\ k_v/(2t-1) & \text{otherwise} \end{cases} \tag{13.1}$$

where k_v denotes the current degree of vertex v before the connection. The normalizing term is $(2t-1)$ because the new edge is understood to be incident to v_t only, until its second endpoint is chosen.

The LCD Model. To construct a Linearized Chord Diagram as defined in the beginning of this section we can use *n-pairings*. An n-pairing L is a partition of the set $S = \{1, 2, \ldots, 2n\}$ into pairs. So there are $\frac{(2n)!}{n!2^n}$ n-pairings. Figure the elements of S in their natural order on the x-axis and represent each pair by connecting its two elements by a chord (compare Figure 13.2). On such a Linearized Chord Diagram the construction of the graph for the pairing L becomes understandable. Construct the graph $\Phi(L)$ by the following rules: starting from the left of the x-axis we identify all endpoints up to and including the first right endpoint of a chord to form vertex v_1. Then we identify all further endpoints until the second right endpoint as vertex v_2 and so on. To form the edges we replace all chords by an edge connecting the vertices associated with the endpoints. Figure 13.2 gives an example of such a Linearized Chord Diagram and the associated graph.

The same can be achieved by choosing $2n$ points at random in the $[0, 1]$ interval and associating the points $2i - 1$ and $2i$, $i \in \{1, 2, \ldots, n\}$ as a chord.

LCD's as Static Representation of (G_m^n). For a special point in time $t = n$ we can construct a Linearized Chord Diagram with n chords and build the graph $\Phi(L)$. The obtained graph model is exactly the nth state of the graph process (G_1^t), i.e., G_1^n. To see this observe how the evolution rule of (G_1^t) can be imitated for LCD's. Add one pair to an arbitrary LCD by placing the right point of the

pair at the end of the point set and inserting the left point of the pair uniformly at random before any of the $2n+1$ points. Then the new edge is connected by the same distribution as in the (G_1^t) process.

It can easily be shown that the degree distribution of this 'static' graph follows a power law with exponent $\gamma = -3$ (for details see [69]).

Now we will give three heuristic approaches that work with the preferential attachment model.

Continuum Theory. Let k_i again denote the degree of vertex i. The value k_i increases if a new vertex v enters the network and connects an edge to vertex i. The probability that this happens will be $\frac{k_i}{\sum_{j\in V\setminus\{v\}} k_j}$. Note that this does not yet determine the full probability distribution, but it is sufficient for our argument. In addition we have to specify a start sequence. We want to start with a graph of $m_0(\geq m)$ vertices and zero edges. As in this case the probability distribution is not defined, we stipulate it to be the uniform distribution for the first step. Obviously after the first step we have a star plus vertices of degree zero which are irrelevant for the further process. Unfortunately, the exact shape of the initial sequence is not given in [15].

We now want to consider k_i as a continuous real variable. Therefore the rate with which k_i changes is proportional to the probability that an edge connects to i. So we can state the following dynamic equation:

$$\frac{\partial k_i}{\partial t} = m\frac{k_i}{\sum_{jj\in V\setminus\{v\}} k_j} \tag{13.2}$$

So we get for the total number of degrees in the network, except for that of the new vertex, $\sum_{j=1}^{N-1} k_j = 2mt - m$.

Thus the above equation changes to $\frac{\partial k_i}{\partial t} = \frac{k_i}{2t-1}$ and, since we consider very large times t, we can approximate it as

$$\frac{\partial k_i}{\partial t} = \frac{k_i}{2t}. \tag{13.3}$$

By construction of the preferential attachment model we know that the initial condition $k_i(t_i) = m$ holds where t_i is the time when vertex i was inserted into the network. Using this initial condition we obtain as a solution of the differential equation (13.3) the following result:

$$k_i(t) = m\left(\frac{t}{t_i}\right)^{\beta}, \qquad \beta = \frac{1}{2}. \tag{13.4}$$

Our goal is now to determine $p(k)$, the probability that an arbitrary vertex has degree exactly k. Since $p(k) = \frac{\partial \Pr[k_i(t)<k]}{\partial k}$ we firstly have to determine the probability that the degree of vertex i at time t is strictly smaller than k.

By using the solution of the differential equation given above, the following equations arise:

$$\begin{aligned}
\Pr[k_i(t) < k] &= \Pr[m\left(\frac{t}{t_i}\right)^{\beta} < k] \\
&= \Pr\left[t_i > \frac{m^{\frac{1}{\beta}} t}{k^{\frac{1}{\beta}}}\right] \\
&= 1 - \Pr\left[t_i \leq \frac{m^{\frac{1}{\beta}} t}{k^{\frac{1}{\beta}}}\right] \\
&= 1 - \int_0^{m^{\frac{1}{\beta}} t k^{-\frac{1}{\beta}}} \Pr[t_i = t]\, dt \\
&= 1 - \frac{m^{\frac{1}{\beta}} t}{k^{\frac{1}{\beta}}(t + m_0)}
\end{aligned}$$

The last equation follows from the fact that the probability space over t_i has to sum up to one and the probabilities are assumed to be constant and uniformly distributed, thus $1 = \sum_{i=1}^{t} \Pr[t_i] \quad \Longrightarrow \quad \Pr[t_i] = \frac{1}{m_0+t}$.

Differentiating the above equations with respect to k we obtain for $p(k)$:

$$p(k) = \frac{\partial \Pr[k_i(t) < k]}{\partial k} = \frac{2m^{\frac{1}{\beta}} t}{m_0 + t} \cdot \frac{1}{k^{\frac{1}{\beta}+1}}. \tag{13.5}$$

For $t \to \infty$ asymptotically we get $p(k) \sim 2m^{\frac{1}{\beta}} k^{-\gamma}$ with $\gamma = \frac{1}{\beta} + 1 = 3$. Note that the exponent is independent of m. So we get a degree distribution that follows a power law where the coefficient is proportional to m^2.

Master Equation Approach. With the master equation approach we want to use recursion to find the shape of the degree distribution. So we are looking for equations that use information from the time steps before, in form of the degree distribution of older time steps. Since we know the initial distribution it is easy to solve this recursion.

This approach to determining the power law of the preferential attachment model was introduced by Dorogovtsev, Mendes, and Samukhin [166].

We study the probability $p(k, t_i, t)$ that a vertex i that entered the system at time t_i has degree k at time t. During the graph process the degree of a vertex i increases by one with probability $\frac{k}{2t}$.

For simplicity of formulas we use the dot notation for the derivative with respect to t.

A master equation for this probability $p(k, t_i, t)$ is of the form:

$$\dot{p}(k, t_i, t) = \sum_{k'} [W_{k' \to k}\ p(k', t_i, t) - W_{k \to k'}\ p(k, t_i, t)] \tag{13.6}$$

Here $W_{k' \to k}$ denotes the probability of changing from state k' to state k. In our model this probability is obviously

$$W_{k' \to k} = \frac{k'}{2t}\delta_{k',k-1} \qquad \text{, where} \quad \delta_{i,j} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{13.7}$$

is the Kronecker symbol.

By summing up over all vertices inserted up to time t, we define the probability $P(k,t) := \frac{\sum_{t_i}^{t} p(k,t_i,t)}{t}$ that some arbitrary vertex has degree k.

As we are interested in a stationary distribution, we are looking for the point where the derivative with respect to time is zero.

$$\begin{aligned}
0 = \dot{P}(k,t) &= \frac{t\sum_{t_i}\dot{p}(k,t_i,t) - \sum_{t_i} p(k,t_i,t)}{t^2} \\
&= \left(\frac{1}{t}\sum_{t_i}\dot{p}(k,t_i,t)\right) - \frac{1}{t}P(k,t) \\
&= \left(\sum_{k'}\frac{1}{t}\left[W_{k' \to k}\; p(k',t_i,t) - W_{k \to k'}\; p(k,t_i,t)\right]\right) - \frac{1}{t}P(k,t) \\
&= \left(\sum_{k'}\left[W_{k' \to k}\; P(k',t) - W_{k \to k'}\; P(k,t)\right]\right) - \frac{1}{t}P(k,t) \\
&= \left(\sum_{k'}\left[\frac{k'}{2t}\delta_{k',k-1}P(k',t) - \frac{k}{2t}\delta_{k,k'-1}P(k,t)\right]\right) - \frac{2}{2t}P(k,t) \\
&= \frac{k-1}{2t}P(k-1,t) - \frac{k+2}{2t}P(k,t)
\end{aligned}$$

There is now a t' so that for every time t greater than t' we get the stationary distribution, $\tilde{P}(k)$. This results in the recursive equation $\tilde{P}(k) = \frac{k-1}{k+2}\tilde{P}(k-1)$ for $k \geq m+1$. For the case $k = m$ the probability directly results from the scaling condition of the probability measure: $\tilde{P}(m) = \frac{2}{m+2}$.

This directly yields the power law of the form $\Pr[k] = \frac{2m(m+1)}{k(k+1)(k+2)}$ which converges to the value of the power law found using the continuum theory, $2m^2\gamma^{-3}$.

This approach can also be used to determine the degree distribution of a more general case of preferential linking. In this model, one new vertex is inserted at every point in time. At the same time we insert m edges with one endpoint at unspecified vertices or from the outside. This can be done since here we only take into consideration the indegree of a vertex. The other endpoints are distributed to existing vertices proportional to $q(s) + A$ where $q(s)$ is the indegree of vertex s, and A an additional attractiveness associated with all vertices.

Rate Equation Approach. In this approach we want to analyze the change over time of the numbers of vertices with degree k – we are looking for the rate at which this number changes.

This approach for the preferential attachment model is due to Krapivsky, Redner, and Leyvraz [369].

We are considering the average number (over all graphs of the state of the process) $N_k(t)$ of vertices that have exactly degree k at time t. Asymptotically we have, by the strong law of large numbers, the following for large t: $N_k(t)/t \sim \Pr[k]$ and $\sum_k kN_k(t)/t \sim 2m$.

If a new vertex enters the network, $N_k(t)$ changes as follows:

$$\Pr[k] = \quad \frac{\partial N_k}{\partial t} = m\frac{(k-1)N_{k-1}(t) - kN_k(t)}{\sum_k kN_k(t)} + \delta_{k,m}. \tag{13.8}$$

Here the first term of the numerator denotes the total number of edges leaving vertices with degree exactly $k-1$ where new edges connect to those vertices and therefore increase the degree to k. The second term determines the number of edges leaving vertices with degree exactly k where new edges connect to those vertices and therefore increase the degree to a value higher than k. If the newly entered vertex has exactly degree k, i.e., $m = k$, then we have to add a 1 to our rate equation.

Applying the above limits we obtain exactly the same recursive equation as found with the master equation approach, and therefore we have the same power law.

Flexibility of the Approaches. All the approaches mentioned before are very helpful and easy to understand for the case of analyzing the preferential attachment model. Some of them are also applicable for more general versions of the preferential attachment model, as for $m \neq 1$ and others. But it is not clear whether there is a useful application of these approaches to totally different models. For the rate equation approach an adaption to more general evolving networks, as well as for networks with site deletion and link-arrangement, is possible. There is a huge need for approaches that can deal with other models. It would be even more desirable to find a way to treat numerous types of evolving network models with a single approach.

13.2.2 Generating Functions

The power law exemplifies an often faced problem in network-analysis: In many cases all that is known of the network is its degree sequence, or at least the distribution of the degrees. It seems as if one could infer certain other structural features of the network, for example second order neighbors from its degree-sequence. Combinatorics provides a powerful tool to retrieve such insights from sequences: Generating functions. Our goal is to present some basics of generating functions and then develop the method for the special purposes of network analysis.

Ordinary Generating Functions. We are given the distribution of the degree sequence, to be precise a function $p(k)$ mapping each vertex degree k to the probability for a randomly chosen vertex – in a network chosen according to that degree sequence – to be adjacent to k other vertices. (For simplicity we confine ourselves to undirected graphs.) Calculating the expectation of that distribution immediately gives the (expected) average degree z_1, i.e., the average number of neighbors of a random vertex. Can we as easily calculate the probability for a vertex to have k second order neighbors, i.e., vertices whose shortest path distance to it equals exactly 2, from the distribution of the first order neighbors? Trying a direct approach, one might want to average over all degrees of a vertex the average of the degrees of the adjacent vertices. In some sense, one would like to perform calculations that relate the whole distribution to itself. But how to get hold of the entire distribution in a way useful for calculation? A generating function solves exactly this problem: On the one hand, it is an encoding of the complete information contained in the distribution, but on the other hand it is a mathematical object that can be calculated with. We define:

Definition 13.2.1. *For a probability distribution* $p : \mathbb{N} \mapsto [0, 1]$

$$G_p(x) = \sum_k p(k)x^k \tag{13.9}$$

is called the generating function *of* p.

This definition is by no means in its most general form. This particular way of encapsulating p is sometimes called the ordinary generating function.

The formal definition is justified in the light of the following proposition:

Proposition 13.2.2. *Let* p *be a probability distribution and* G_p *its generating function:*

1. $$G_p(1) = 1$$
2. $$G_p(x) \text{ converges for } x \text{ in } [-1, 1].$$
3. $$p(k) = \frac{1}{k!} \left. \frac{\partial^k G_p}{\partial x^k} \right|_{(x=0)}$$
4. $$E(p) := \sum_k kp(k) = G_p'(1)$$
5. $$\mathrm{Var}(p) := \sum_k k(k-1)p(k) = G_p''(1)$$

The convergence is shown by standard analytic criteria. The other parts of the proposition are immediate from the definition, keeping in mind for the first that $\sum_k p(k) = 1$ for a probability distribution.

Part 3 of the proposition shows that a generating function encodes the information of the underlying distribution. From another perspective a generating function, G_p, is a formal power series that actually converges on a certain interval. Taking powers $(G_p(x))^m$ of it will again result in such a power series. Interpreting this in turn as a generating function amounts to interpreting the coefficient of some x^k in $(G_p(x))^m$. For $m = 2$ this is $\sum_{j+l=k} p(j)p(l)$, in other words, this is the probability that the values of two independent realizations of the random variable with distribution p sum up to k. In general $(G_p(x))^m$ is the generating function for the distribution of the sum of the values of m independent realizations of the random variable distributed according to p.

Generating Functions for Degree Sequences. For $k \in \mathbb{N}$ let D_k be a random variable equal to the number of vertices of degree k. Further $p(k)$ shall be the probability that a randomly chosen vertex has degree equal to k. It holds that $np(k) = E(D_k)$, the expectation of the random variable. To construct a random graph according to the distribution p may mean two slightly different things. We may take D_k to be a constant function for every k, thus, there is a fixed number of vertices with degree k. Alternatively, we only require the *expectation* of D_k to equal that fixed number. The latter model will make the graphs to which the first model is confined only the most probable. Moreover, as the first fixes the degree sequence, only those sequences of fixed values of D_k that are realizable generate a non-empty model. For example the sum of all degrees must not be odd. (The next section will discuss which degree sequences are realizable.) Despite these differences, for a realizable sequence the statistical results we are interested in here are not affected by changing between these two models. We confine our explicit considerations to the second and more general interpretation, where $p(k)$ only prescribes the expectation of D_k.

To justify the technicality of generating functions, some structural features of the network should be easily derived from its degree sequence's distribution. So far the average vertex degree z_1 has been shown to be $G'_p(1)$, which is not a real simplification for computation. Next we ask for the degree distribution of a vertex, chosen by the following experiment: Choose an edge of the graph uniformly at random and then one of its endpoints. The probability f to thereby reach a vertex of degree k is proportional to $kp(k)$. That means the corresponding generating function is $G_f(x) = \sum_k \frac{kp(k)}{\sum_k kp(k)} x^k = x\frac{G'_p(x)}{G'_p(1)}$. Removing the factor x in the right hand term amounts to reducing the exponent of x in the middle term, thus obtaining a generating function, where the coefficient of x^k in $G_f(x)$ becomes the coefficient of x^{k-1} in the new generating function. Hence the new function is the generating function of $f(k-1)$. Interpreting this combinatorially, we look at the distribution of the degrees minus one. In other words, we want to know the probability distribution p^* for the number of edges that are incident to the vertex, not counting the one edge we came from in the above choosing

procedure. Its generating function can thus be written nicely, as

$$G_{p^*}(x) = \sum_k \frac{kp(k)}{\sum_k kp(k)} x^{k-1} = \frac{G'_p(x)}{G'_p(1)} \quad . \tag{13.10}$$

This distribution p^* is useful to determine the distribution of rth neighbors of a random vertex.

Distribution of rth Neighbors. What is the probability, for a randomly chosen vertex v, that exactly k vertices are at a shortest path distance of exactly r? For $r = 2$, assume that the number of vertices of distance exactly 2 from v is $(\sum_{w \in N(v)} d(w)) - d(v)$, (where $d(v)$ denotes the degree of v), and for general r that the network contains no cycles. This assumption seems to be a good approximation for big, sparse, random graphs, as the number of occasional cycles seems negligible. But its exact implications are left to be studied. For the sake of explanation, assume the network to be directed in the following way: Choose a shortest-path tree from a fixed vertex v and direct each of its edges in the orientation in which it is used by that tree. Give a third orientation, zero, to the remaining edges. In this way the definition is consistent even for non tree-like networks. But assume again a tree structure. Any vertex except for v has exactly one in-edge and p^* is the distribution of the number of out-edges of that in-edge. Now a second assumption is required: For an out-edge $\{x, y\}$ of a vertex x the degree of y shall be distributed independently of that of x's other out-edges' endvertices, and independently of the degree of x. Of course, there are examples of pathological distributions for the degree-sequence where this assumption fails. Again, the assumption seems reasonable in most cases. Again, precise propositions on the matter are left to be desired.

Given these two assumptions, tree structure and independence, the generating function of the distribution of second neighbors is seen to be

$$\sum_k p(k)(G_{p^*}(x))^k = G_p(G_{p^*}(x)), \tag{13.11}$$

recalling that k independent realizations of a random variable amount to taking the kth power of its generating function. Correspondingly, the generating function of the distribution of the rth neighbors $G_{(r)}$ is:

$$G_{(r)} := \underbrace{G_p(G_{p^*}(G_{p^*} \ldots G_{p^*}(x)))}_{r \text{ functions altogether}}. \tag{13.12}$$

Taking expectations for second neighbors, i.e., calculating z_2, simplifies nicely:

$$z_2 = [G_p(G_{p^*}(x))]'|_{(x=1)} = G'_p(\underbrace{G_{p^*}(1)}_{=1})G'_{p*}(1) = G''_p(1) \tag{13.13}$$

Recall that the expectation of the first neighbors z_1 is $G'_p(1)$. Note that in general the formula for r-neighbors does not boil down to the rth derivative.

Component Size. For the analysis of the component size, first consider the case without a giant component. A giant component is a component of size in $\Theta(n)$. Thus we assume that all components have finite size even in the limit. Assume again the network to be virtually tree-like. Again the results are subject to further assumptions on the independency of certain stochastic events. And again these assumptions are false for certain distributions and, though likely for large networks, it is unclear where they are applicable. To point out these presuppositions we take a closer look at the stochastic events involved.

Construct a random graph G for a given probability distribution of the vertex degree, p, as always in this section. Next, choose an edge e uniformly at random among the edges of G. Flip a coin to select v, one of e's vertices. The random variable we are interested in is the size of the component of v in $G \setminus e$. Let p° be its distribution, and p^* as above the distribution of the degree of v in $G \setminus e$ found by this random experiment. Then for example $p^\circ(1) = p^*(0)$. In general, for k the degree of v, let $n_1, \ldots, n_k$ be the neighbors of v in $G \setminus e$. Further, we need a laborious definition: $P_k(s-1) :=$ Pr[The sizes of the components of the k vertices $n_1 \ldots n_k$ in $G \setminus \{e, (v,n_1), \ldots (v,n_k)\}$ sum up to $s-1$.] Then when may write: $p^\circ(s) = \sum_k p^*(k) P_k(s-1)$. How to compute P_k? It does not in general equal the distribution of the component size of a randomly chosen vertex when removing one of its edges. Take into account that in the above experiment a vertex is more likely to be chosen the higher its degree. On the other hand, supposing a tree-like structure, the component size of n_j is the same in $G \setminus \{e, (v,n_1), \ldots (v,n_k)\}$ as in $G \setminus (v,n_j)$. Now, assume that our experiment chooses the edges (v,n_i) independently and uniformly at random among all edges in G, then P_k is distributed as the sum of k random variables distributed according to p°. These assumptions are not true in general. Yet, granted their applicability for a special case under consideration, we can conclude along the following lines for the generating function of p°:

$$\begin{aligned} G_{p^\circ}(x) = \sum_{s=0}^{n} p^\circ x^s &= \sum_{s=0}^{n} x^s \sum_k p^*(k) P_k(s-1) \\ &= x \sum_k p^*(k) \underbrace{\sum_{s=0}^{n} x^{s-1} P_k(s-1)}_{G_{P_k}(x)} \end{aligned}$$

Since we presume P_k as the distribution of the sum of k independent realizations of p°, we have $G_{P_k}(x) = G^k_{p^\circ}(x)$, and $G_{p^\circ}(x) = x \sum_k p^*(k) (G_{p^\circ}(x))^k$. This can be restated as

$$G_{p^\circ}(x) = x G_{p^*}(G_{p^\circ}(x)). \tag{13.14}$$

In a similar way we arrive at a consistency requirement for $p^\bullet$, the distribution of the component size of a randomly chosen vertex:

$$G_{p^\bullet}(x) = x G_p(G_{p^\circ}(x)) \tag{13.15}$$

The assumptions on stochastic independence made here are not true in general. Granted they are verified for a specific degree distribution, the functional

Equations (13.14) and (13.15) still withstand their general solution. Numerical solutions have been carried out for special cases (for details see [448]).

But the expected component size of a random vertex can be computed directly from those equations. The expectation of a distribution is the derivative of its generating function at point 1. Therefore $E(p^\bullet) = G'_{p^\bullet}(1) = 1 + G'_p(1)G'_{p^\circ}(1)$. But, as $G'_{p^\circ}(1) = 1 + G'_{p^*}(1)G'_{p^\circ}(1)$, this becomes:

$$E(p^\bullet) = G'_{p^\bullet}(1) = 1 + \frac{G'_p(1)}{1 - G'_{p^*}(1)} = 1 + \frac{z_1^2}{z_1 - z_2} \tag{13.16}$$

Giant Component. So far we have excluded graphs with a giant component, i.e., a component that grows linearly with the graph. For a distribution that would generate such a component, the probability for a cycle would of course be no longer negligible. If we, however, still infer a tree-like structure as a good approximation, Formula 13.16 for the expected component size should no longer be independent of n, the number of vertices.

Indeed for $G'_{p^*}(1) \to 1$ equation (13.16) diverges, meaning that the expected component size is not bounded for unbounded n. What can be derived from $G'_{p^*}(1) = 1$?

$$\begin{aligned} 1 = G'_{p^*}(1) \quad &\iff \\ \sum_k k(k-1)p(k)x^{k-2}\Big|_{(x=1)} = \sum_k kp(k) \quad &\iff \\ \sum_k k(k-2)p(k) = 0 \end{aligned}$$

This equation marks the phase transition to the occurrence of a giant component, as the sum on the left increases monotonically with the relative number of vertices of degree greater than 2.

How much of the graph is occupied by the giant component? In [448] it is claimed that the above considerations on the component size still apply to the 'non-giant' part of the graph. But $G_p\bullet(1)$ becomes smaller than 1 in these cases. Following the lines of [448], this should in turn give the fraction of the vertex set that is covered by non-giant components. In other words, $n(1 - G_p\bullet(1))$ equals the (expected) number of vertices in the giant component. This is an audacious claim, as we calculate information about the non-giant part insinuating that it shows the same degree distribution as the whole graph. For example high-degree vertices could be more likely to be in the giant-component. Maybe those calculations actually lead to reasonable results, at least in many cases, but we cannot give any mathematical reason to be sure.

Generating Functions for Bipartite Graphs. So far this section has collected several ideas based on generating functions in order to squeeze as much information as possible from the mere knowledge of the degree distribution. Some

of them depend on further assumptions, some are less appropriate than others. Some conclusions drawn in the literature are left out due to their questionable validity.

Finally, we become a little more applied. Many real-world networks show a bipartite structure. For example, in [184] we find a graph model used to analyze how infections can spread in a community. The model consists of two sets of vertices, persons and places, and edges from any person to any place the person regularly visits. As in other examples, like the co-author or the co-starring network, we are given bipartite data, but the interest is often mainly in the projection onto one of the vertex sets, mostly the persons'. Suppose we are given two probability distributions of degrees a and b, for the persons and the places, and the fraction ρ between the numbers of persons and places. Make a partition of n vertices according to ρ, realize a and b each in one part of the partition, and choose $\bar{H}$ uniformly at random among all bipartite graphs of n vertices with the same partition and degree sequences on the partition sets. Let H be the projection of $\bar{H}$ on the persons' vertices, and p the corresponding distribution of its degree sequence. Then $G_p = G_b(G_{a^*})$ and $G_{p^*} = G_{b^*}(G_{a^*})$. Now the whole machinery of generating functions can be applied again. In this way generating functions can help to bridge the gap between the bipartite data we are given and the projected behavior we are interested in.

13.2.3 Graphs with Given Degree Sequences

Given a degree sequence, generating functions allow to derive some deeper graph properties. Now we wish to construct a graph of a given degree sequence. At best, the generating algorithm would construct such a graph with uniform probability over all graphs that have a proposed degree sequence $d_1, d_2, \ldots, d_n$.

For reasons of simplicity we assume that $d_1 \geq d_2 \geq \cdots \geq d_n$ are the degrees of vertices $v_1, v_2, \ldots, v_n$.

Definition 13.2.3. *A degree sequence $d_1, d_2, \ldots, d_n$ is called* realizable *if there is a graph G with vertices $v_1, v_2, \ldots, v_n \in V$ with exactly the given degree sequence.*

Erdős and Gallai [180] gave sufficient and necessary conditions for realizability of a simple, undirected graph.

Necessary and Sufficient Conditions. In order to construct a graph with a given degree sequence we should at first verify whether that sequence is realizable at all. Secondly, we are only interested in connected graphs. Thus, we also want to know whether the degree sequence can be realized by a connected graph.

Starting with the first property we can observe the following. A degree sequence $d = (d_1, d_2, \ldots, d_n)$ is realizable if and only if $\sum_{i=1}^{n} d_i$ is even (since the sum of the degrees is twice the number of edges), and for all subsets $\{v_1, v_2, \ldots, v_\ell\}$ of the ℓ highest vertex degrees, the degrees of those vertices can be absorbed within those vertices and with the outside degrees. This means that there are

enough edges within the vertex set and to the outside to bind to all the degrees. More formally we can state the following theorem.

Theorem 13.2.4. *A degree sequence $d = (d_1, d_2, \ldots, d_n)$ is realizable if and only if $\sum_{i=1}^{n} d_i$ is even and*

$$\sum_{i=1}^{\ell} d_i \leq \ell(\ell-1) + \sum_{i=\ell+1}^{n} \min\{\ell, d_i\} \qquad 1 \leq \ell \leq n. \tag{13.17}$$

This inequality is intuitively obvious, and therefore one direction of the theorem is trivial to prove. All degrees in the first ℓ degrees of highest order have to be connected first of all to the $(\ell-1)$ other vertices in this set of vertices. The rest of the open degrees have to be at least as many as there are open degrees in the outside of the chosen set. How many can there be? For each vertex there is the minimum of either ℓ (since no more are needed for the ℓ vertices in the chosen set) or the degree of a vertex i where only vertices $\ell+1, \ldots, n$ are taken into account.

A more precise theorem about realizability of an undirected, simple graph is given below.

Theorem 13.2.5. *A sequence $d = (d_1, d_2, \ldots, d_n)$ is realizable if and only if the sequence $H(d) = (d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_n)$ is realizable.*

Furthermore we are interested not only in a graph with this degree sequence, but in a connected graph. The necessary and sufficient conditions on connectedness are well known, but should be repeated here for completeness.

Theorem 13.2.6. *A graph G is connected if and only if it contains a spanning tree as a subgraph.*

As we neither have a graph nor a spanning tree, we are interested in a property that can give us the information whether a graph with certain degree sequence is constructible. As a spanning trees comprises $(n-1)$ edges, the sum of degrees must be at least $2(n-1)$. This necessary condition is already sufficient, as it will become clear from the constructing algorithms given below.

If we can fulfill Theorem 13.2.4, and $\sum_{v_i \in V} d_i \geq 2(n-1)$ holds, there exists a connected graph with the given degree sequence.

Algorithms. There are several easy-to-implement algorithms with linear running time that construct a graph with a given degree sequence. In the following we present two slightly different algorithms; one constructs a graph with a sparse core, the other constructs a graph with a dense core. The reader has to be aware that all of these easy algorithms do not construct a random graph out of all graphs with the desired degree sequence with the same probability. But starting from the graph constructed by one of these algorithms we give a method to generate a random instance that is indeed equiprobable among all graphs with the desired degree sequence. We assume that the sum of all degrees is at least $2(n-1)$.

For both algorithms we need a subroutine called `connectivity`. This subroutine first of all checks whether the constructed graph is connected. If the graph G is not connected, it finds a connected component that contains a cycle. Such a connected component must exist because of the assumption on the degrees made above. Let uv be an edge in the cycle, and st be an edge in another connected component. We now delete edges uv and st, and insert edges us and vt to the network.

Sparse Core. In this section we want to describe an algorithm that constructs a graph with the given degree sequence that additionally is sparse. We are given a degree sequence $d_1 \geq d_2 \geq \cdots \geq d_n$, and we assign the vertices $v_1, v_2, \ldots, v_n$ to those degrees. As long as there exists a vertex v_i with $d_i > 0$, we choose the vertex v_ℓ with the currently lowest degree d_ℓ. Then we insert d_ℓ edges from v_ℓ to the first d_ℓ vertices with highest degree. After that we update the residual vertex degrees $d_i = d_i - 1$ for $i = 1, \ldots, d_\ell$ and $d_\ell = 0$. Last, but not least, we have to check connectivity and, if necessary, establish it using the above mentioned method `connectivity`.

Dense Core. To construct a graph with a dense core for a certain degree sequence, we only have to change the above algorithm for sparse cores slightly. As long as there exists a vertex v_i with $d_i > 0$ we now choose such a vertex arbitrarily and insert edges from v_i to the d_i vertices with the highest residual degrees. After that we only have to update the residual degrees and establish connectivity if it is not given.

Markov-Process. To generate a random instance from the space of all graphs with the desired degree sequence, we start using an easy to find graph G with the desired realization. In a next step, 2 edges (u, v) and (s, t) with $u \neq v, s \neq t$ such that $(u, s), (v, t) \notin G$ are chosen uniformly at random. The second step is to delete the edges (u, v) and (s, t) and replace them with (u, s) and (v, t).

This process is a standard Markov-chain process often used for randomized algorithms. We can observe that the degree distribution is unchanged by this algorithm. If rewiring two edges would induce a disconnected graph, the algorithm simply does not do this step, and repeats the random choice. The following theorem states that this algorithm constructs a random instance out of the space of all graphs with the desired degree sequence.

Theorem 13.2.7. *Independent of the starting point, in the limit, the above Markov-chain process will reach every possible connected realization with equal probability.*

For practical reasons one has to find a stopping rule so that we can bound the number of steps of the algorithm. Mihail et al. [420] observed that the process levels off in terms of the difference of two sorted lists (at different points in time) of all neighbors (by degree) of nodes with unique degrees. Using this measure they heuristically claim a number of at most 3 times the level-off number of steps to get a good random graph for instances like today's AS-level topology

(about 12,000 vertices). They observed the number of steps to level-off to be less than 10,000 for graphs of 3,000 vertices, less than 75,000 for graphs with 7,500 vertices, and less than 180,000 for graphs with 11,000 vertices.

***d*-Regular Graphs.** A special variant of graphs with given degree sequences are d-regular graphs where each vertex has exactly degree d.

There are several algorithms known that can construct an equiprobable d-regular graph. McKay and Wormald [416] gave an algorithm that is also applicable for arbitrary degree sequences. For a given $d \in \mathcal{O}(n^{\frac{1}{3}})$ its expected running time is in $\mathcal{O}(n^2d^4)$, and furthermore it is very difficult to implement. A modification of this algorithm for only d-regular graphs improves the running time to $\mathcal{O}(nd^3)$, but does not remove the disadvantages. Tinhofer [550] gave a simpler algorithm that does not generate the graphs uniformly at random and, moreover, the resulting probability distribution can be virtually unknown. Jerrum and Sinclair [329] introduced an approximation algorithm where all graphs have only a probability varying by a factor of $(1 + \varepsilon)$, but the d-regular graph can be constructed in polynomial time (in n and ε), and the algorithm works for all possible degrees d.

A very simple model is the pairing model, introduced in the following. The running time is exponential ($\mathcal{O}(nd \exp(\frac{d^2-1}{4}))$), and the graph can only be constructed in this running time for $d \leq n^{\frac{1}{3}}$.

Pairing Model. A simple model to construct a d-regular graph is the so-called pairing model. There, nd points are partitioned into n groups – clearly every group should include exactly d points. In a first step a random pairing of all points has to be chosen. Out of this pairing we now construct a graph G. Let the n groups be associated with n vertices of the graph. There is an edge (i, j) between vertices i and j in the graph if and only if there is a pair in the pairing containing points in the ith and jth group. This so constructed graph is a d-regular graph if there are no duplicated edges. Furthermore, we have to check a posteriori whether the graph is connected.

13.3 Further Models of Network Evolution

In this section we want to present some further models for evolving networks. Since there is a huge variety of them, we want to consider only some of those network models that include significantly new ideas or concepts.

13.3.1 Game Theory of Evolution

The literature for games *on* a (fixed) network is considerable. But game theoretical mechanisms can also be used to *form* a network, and this falls in our scope of interest. The following example is designed to model economic cooperation.

Vertices correspond to agents, who establish or destroy an edge between each other trying to selfishly maximize their value of a cost revenue function.

The objective function of an agent sums revenues that arise from each other agent directly or indirectly connected to him minus the costs that occur for each edge incident to him: Let c be the fixed costs for an incident edge and $\delta \in (0,1)$. The cost revenue of a vertex v is $u_v(G) = (\sum_{w \in V(G)} \delta^{d(v,w)}) - \deg(v)c$, where G is the current network and $d(v,w)$ is the edge-minimal path distance from v to w in G. (To avoid confusion we denote the degree of a vertex v by $\deg(v)$ here.) Set that distance to infinity for vertices in different components, or confine the index set of the sum to the component of v.

An edge is built when it increases the objective function of at least one of the agents becoming incident and does not diminish the objective function of the other. To delete an edge it suffices that one incident agent benefits from the deletion. In fact the model analyzed is a little more involved. Agents may simultaneously delete any subset of their incident edges, while participating in the creation of a new edge, and consider their cost revenue function after both actions.

To put it formally:

Definition 13.3.1. *A network G is* stable *if for all $v \in V(G)$*

$$\forall e \in E(G) : v \in e \implies u_v(G) \geq u_v(G \setminus e)$$

and

$$\forall w \in V(G), \forall S \subseteq \{e \in E(G) \mid v \in e \vee w \in e\} :$$
$$u_v((G \cup \{v,w\}) \setminus S) > u_v(G) \implies u_w((G \cup \{v,w\}) \setminus S) < u_w(G)$$

This quasi-pareto notion of stability does not guarantee that a stable network is in some sense 'good', namely that at least in total the benefit is maximal. Therefore we define:

Definition 13.3.2. *A network G is* efficient *if*

$$\forall G' : V(G) = V(G') \implies \sum_v u_v(G') \leq \sum_v u_v(G).$$

Theorem 13.3.3. *In the above setting we have:*
For $c < \delta, (\delta - c) > \delta^2$ the complete graph is stable.
For $c < \delta, (\delta - c) \leq \delta^2$ the star is stable.
For $c \geq \delta$ the empty graph is stable.

Theorem 13.3.4. *In the above setting we have:*
For $(\delta - c) > \delta^2$ only the complete graph is efficient.
For $(\delta - c) < \delta^2, c < \delta + (n-2)\delta^2/2$ only a star is efficient.
For $(\delta - c) < \delta^2, c > \delta + (n-2)\delta^2/2$ only the empty graph is efficient.

Through the work of A. Watts [572], this approach received a push towards evolution. Given the parameters c and δ, the question is, which networks will emerge? This remains unclear until the order in which agents may alter their

incident part of the edge set (and the initial network) is given. Up to now only the case for an empty network as the initial configuration has been scrutinized. The order, in which the agents can make their decisions, is given in the following random way: At each step t of a discretized time model, one edge e of the complete graph of all agents (whether e is part of the current network or not) is chosen uniformly at random. Then the two incident agents may decide whether to keep or drop or, respectively, establish or leave out the edge e for the updated network. This means for an existing edge e that it is deleted if and only if one of its endvertices benefits from the deletion, and for a non-existing edge e that it is inserted if and only if at least one of the potentially incident vertices will benefit and the other will at least not be worse off. Note that in this model more sophisticated actions, that are comprised of the creation of one and the possible deletion of several other edges, are not allowed. All decisions are taken selfishly by only considering the cost revenue of the network immediately after the decision of time t. In particular no vertex has any kind of long time strategy. The process terminates if a stable network is reached. For this model the following holds:

Theorem 13.3.5. *In the above setting we have:*

For $(\delta - c) > \delta^2 > 0$	*the process terminates in a complete graph in finite time.*
For $(\delta - c) < 0$	*the empty set is stable.*
For $\delta^2 > (\delta - c) > 0$	$P_{star} :=$ Pr[*Process terminates in finite time in a star*] > 0, *but* $P_{star} \to 0$ *for* $n \to \infty$.

The first result is obvious as any new edge pays. The second just reformulates the stability Theorem 13.3.3. For the third part, note that a star can no longer emerge as soon as two disjoint pairs of vertices form their edges.

The model and the results, though remarkable, still leave lots of room for further refinement, generalization, and variation. For example, if a star has positive probability that tends to zero, then this could mean that one can expect networks in which some vertices will have high degree, but most vertices will show very low degree. This is a first indication of the much discussed structure of a power law.

13.3.2 Deterministic Impact of the Euclidean Distance

For the following geometric model we want to denote the Euclidean distance between a vertex i and a vertex j by $d(i, j)$. The idea of this model by Fabrikant, Koutsoupias, and Papadimitriou [196] is to iteratively construct a tree. In a first step a sequence $p_0, p_1, \ldots, p_n$ of vertices is distributed within a unit square or unit sphere. In the next step we insert edges successively. Here we want to distinguish between two opposite goals. On the one hand, we are interested in connecting vertices to their geometrically nearest neighbor. On the other hand, we are interested in a high degree of centrality for each vertex. In order to deal with this trade-off between the 'last mile' costs and the operation costs due to

communication delays, we connect the vertices with edges in the following way. Vertex i becomes connected to the vertex j that fulfills $\min_{j<i} \alpha \cdot d(i,j) + h_j$, where h_j denotes the centrality measure and α the relative importance of both goals. Here centrality measures can be the average number of hops to other vertices, the maximum number of hops to another vertex, or the number of hops to a given center - a fixed vertex $v \in V$ (for more details on centrality measures, see Chapter 3).

The behavior of this model is of course highly dependent on the value α and, to a lesser extent, on the shape used to place the vertices. Let T denote the constructed tree in a unit square. And let us define h_j to be the number of hops from p_i to p_0 in the tree T. Then we can state the following properties of T for different values of α.

Theorem 13.3.6. *(Theorem 2.1. in [196])*
If T is generated as above then:

1. *If $\alpha < 1/\sqrt{2}$, then T is a star with vertex p_0 as its center.*
2. *If $\alpha = \Omega(\sqrt{n})$, then the degree distribution of T is exponential, that is, the expected number of vertices that have degree at least k is at most $n^2 \exp(-ck)$ for some constant c:*
   $E[|\{i : \text{ degree of } i \geq k\}|] < n^2 \exp(-ck)$.
3. *If $\alpha \geq 4$ and $\alpha = o(\sqrt{n})$, then the degree distribution of T is a power law; specifically, the expected number of vertices with degree at least k is greater than $c \cdot (k/n)^{-\beta}$ for some constants c and β (that may depend on α though): $E[|\{i : \text{ degree of } i \geq k\}|] > c(k/n)^{-\beta}$. Specifically, for $\alpha = o(\sqrt{3}n)$ the constants are: $\beta \geq 1/6$ and $c = \mathcal{O}(\alpha^{-1/2})$.*

This theorem gives the impression that networks constructed by this algorithm have a degree sequence following a power law. But there are some points to add. The power law given in this theorem does not resemble the definition of power law given in this chapter. Here the authors analyzed the behavior of the degree distribution where only vertices with degree at least k come into consideration. Therefore, one has to take care when comparing results. A second point is that the results only hold for a very small number of vertices in the network. For a majority of vertices (all but $\mathcal{O}(n^{1/6})$) there is no statement made in the work by Fabrikant et al. Subsequently, Berger et al. [58] prove the real behavior of the degree distribution obtained by this model and show that "there is a very large number of vertices with almost maximum degree".

13.3.3 Probabilistic Impact of the Euclidean Distance

The model of Waxman [574] uses the Euclidean distance, henceforth denoted by $d(\cdot,\cdot)$, to determine the probability distribution used to generate a graph of the model. In a first step n points on a finite 2-dimensional lattice are chosen equiprobably to form the vertex set $V(G)$ of the graph G. Then each edge $\{i,j\},\ i,j \in V$, of the complete graph on these vertices is chosen to be part of the edge set $E(G)$ with probability $\Pr(\{i,j\}) = \beta \exp \frac{-d(i,j)}{L\alpha}$. Thereby L denotes

the maximum Euclidean distance of two lattice points, i.e., the diagonal of the lattice.

Increasing $\alpha \in (0, 1]$ will decrease the expected length of an edge, whereas increasing $\beta \in (0, 1]$ will result in a higher number of edges in expectation.

In a variant of the model, the function $d(\cdot, \cdot)$ is defined by random for each pair of chosen vertices. Thus, it will in general not even fulfill the triangle inequality.

13.4 Internet Topology

The Internet consists of two main levels, the router level and the Autonomous System level. Both are systems with certain properties, like a power law with specified exponent, a certain connectivity, and so on. These properties are analyzed by Faloutsos et al. [197] in detail. One goal is now to construct synthetic networks that resemble the Internet very much and, further, that can generate a prediction of the future Internet topology. There are two types of generators: The first type are model-oriented generators that implement only a specified set of models, as for example given in the previous sections. A universal topology generator, on the other hand, should further have the property of being extensible to new models that can be added in an easy way.

To have such a universal generator is interesting for researchers who need good synthetic topologies to simulate their Internet protocols and algorithms. Therefore very good generation tools are needed. Those tools should have at least the following characteristics to be usable for a wide range of researchers and their different applications, not only for Internet topologies (see also [417]).

1. *Representativeness:* The tool should generate accurate synthetic topologies where as many aspects of the target network as possible are reflected.
2. *Inclusiveness:* A single tool should combine the strengths of as many models as possible.
3. *Flexibility:* The tool should be able to generate networks of arbitrary size.
4. *Efficiency:* Even large topologies should be generated in reasonable CPU time and memory.
5. *Extensibility:* The generator should be easily extendible by new models by the user.
6. *User-friendliness:* There should be an easy to learn interface and mechanics of use.
7. *Interoperability:* There should be interfaces to the main simulation and visualization applications.
8. *Robustness* The tool should be robust in the sense of resilience to random failures and, moreover, have the capability to detect errors easily.

These are the desired characteristics of a generator tool. In order to reach the characteristics made above, there are some challenges that have not been solved yet in an acceptable way. Two main challenges in the field of topology generation are (quoted from [417]):

1. How do we develop an adapting and evolving generation tool that constitutes an interface between general Internet research and pure topology generation research? Through this interface, representative topologies, developed by the topology generation research community, can be made readily available to the Internet research community at large.
2. How do we design a tool that also achieves the goal of facilitating pure topology generation research? A researcher that devises a generation model should be able to test it readily without having to develop a topology generator from scratch.

The topology generators available today or, better, their underlying models can be classified as follows (after [417]). On the one hand, there are ad-hoc models that are based on educated guesses, like the model of Waxman [574] and further models [109, 157]. On the other hand there are measurement based models where measures can be, for example, a power law. We can divide this class into causality-oblivious and causality-aware models. By causality we think of some possible fundamental or physical causes, whereas causality-oblivious models orient themselves towards such abstract features as power laws. The INET model and generator, described in Section 13.4.2, and the PLRG model by Aiello et al. [9] belong to the first of these subclasses. The preferential attachment model and the topology generator BRITE belong to the causality-aware models.

13.4.1 Properties of the Internet's Topology

Faloutsos, Faloutsos and Faloutsos [197] analyzed the structure of the Internet topology at three different points in time, and especially analyzed the growing of special metrics. Some of the very obvious metrics are, for example, the rank of a vertex, i.e., the position of the vertex in a sorted list of vertex degrees in decreasing order, and the frequency of a vertex degree, i.e., how often a degree k occurs among all the vertices. Using the minimal distance between two vertices, i.e., the minimal number of edges on a path between the two vertices, one can determine the number of pairs of vertices $P(h)$ that are separated by a distance of no more than h. By taking this definition we obviously have the property that self-pairs are included in $P(h)$ and all other pairs are counted twice. A resultant metric is the average number of vertices $N(h)$ that lie in a distance of at most h hops.

In the Internet there are two levels worth evaluating (for details see [259]). In [197] the data collected by the Border Gateway Protocols (BGP) that stores all inter-domain connections is evaluated. There the authors looked at three special points in time. The first graph is from November 1997, the second from April 1998, and the third from December 1998.

By evaluating this information, and looking for power laws for the above mentioned metrics, they draw the following conclusions. The first conclusion is that the degree $d(v)$ of a vertex v is proportional to the rank r_v of the vertex, raised to the power of a constant, R. This yields the power law of the form

$d(v) \propto r_v^R$. R is defined as the slope in the graph of the function that maps $d(v)$ on the rank of v (denoted as $(d(v);\text{rank of v})$) in log-log plot. A second observation is about the frequency f_k of a vertex degree k: $f_k \propto k^O$ with O a constant. The constant O can be determined by determining the slope of the $(f_k;k)$ plot with a log-log scale. For the total number of pairs of vertices $P(h)$ they can only approximate the power law to the form $P(h) \propto h^H$, where H is the so called hop-plot exponent and is constant. In this case the constant H is defined by the slope of a plot in log-log scale of the $(P(h);h)$ graph.

13.4.2 INET – The InterNEt Topology Generator

This model-oriented topology generator (more details in [332]) tries to implement more than just the analyzed power law in the degree distribution. Several of the analyses of Faloutsos et al. result in exponential laws. The first exponential law they observed and determined an exact form for is the frequency of certain degrees.

$$f_k = \exp(at + b)k^O, \tag{13.18}$$

where f_k is the frequency of a degree k. a, b, O are known constants and t is the time in months since November 1997. Having this equation, we can also predict the frequency of a degree k for t a month in the future.

A second exponential law they found was the degree growth.

$$k = \exp(pt + q)r^R \tag{13.19}$$

The degree k at a given rank r also grows exponentially over time. Here p, q, R are known constants and t is again the number of months since November 1997. This law tells us that the value of the ith largest degree of the Internet grows exponentially. This does not necessarily mean that every AS's degree grows exponentially with time because the rank of a particular AS can change as the number of AS's increases.

Two further exponential laws are the pair size growth and the resultant neighborhood size growth.

$$P_t(h) = \exp(s_h t)P_0(h) \tag{13.20}$$

The pair size within h hops, $P(h)$, grows exponentially with the factor $P_0(h)$, that is the pair size within h hops at time 0 (=November 1997). The neighborhood size within h hops, $A(h)$, grows exponentially as follows.

$$\begin{aligned} A_t(h) = \frac{P_t(h)}{P_0(h)} &= \exp((\log P_0(h) - \log P_0(0)) + (s_h - s_0)t) \\ &= A_0(h)\exp((s_h - s_0)t) \end{aligned} \tag{13.21}$$

Here $A_0(h)$ is the neighborhood size at time 0 (= November 1997). The value t is, as always, the number of months since time 0.

The INET topology generator now uses the observed and analyzed exponential laws to construct a network that strongly resembles the real Internet network

evaluated at time t. In a first step, the user has to input the number of vertices and the fraction p of the number of vertices that have degree one. By assuming exponential growth of the number of AS's in the Internet, the generator computes the value of t – the number of months since November 1997. Then it is easy to also compute the distributions of degree frequency and rank. Since the second power law only holds for 98% of the vertices we have to assign degrees to the top 2% of the vertices using the rank distribution (13.19). p percent of the vertices are assigned degree one. The remaining vertices are assigned degrees following the frequency degree distribution. The edges are inserted into the initial graph G to be generated according to the following rules. First, a spanning tree is built among vertices with degree strictly larger than one. This is done by successively choosing uniformly at random a vertex with degree strictly larger than one that is not in the current tree G, and connecting it to a vertex in G with probability proportional to $\frac{k}{K}$. Here k is the degree of the vertex in G, and K is the sum of degrees of all vertices already in G that still have at least one unfilled degree. In a next step, $p|V|$ vertices with degree one are connected to vertices in G with proportional probability as above. In a final step, the remaining degrees in G, starting with the vertex with largest assigned degree, are connected to vertices with free degrees randomly picked, again using proportional probability. The connectivity of the graph is first tested by a feasibility test before actually inserting the edges.

Other Generators. There are several more topology generators available. The GT-ITM generator [109] is able to construct different topologies. One of them is a transit-stub network that has a well-defined hierarchical structure. The Tiers generator [157] is designed to provide a three level hierarchy that represents Wide Area Networks (WAN), Metropolitan Area Networks (MAN), and Local Area Networks (LAN). The generator BRITE [417] also contains several mechanisms to construct topologies. It not only includes the well-known basic model of Barabási and Albert on router and on AS level but also the Waxman model for both types. It can also illustrate and evaluate the networks made by the INET and GT-ITM generators, and the data obtained from the National Laboratory for Applied Network Research routing data [452].

By using the mentioned power laws observed in the Internet it is now easy to determine the representativeness of such a generator. Medina et al. [418] used the above mentioned topology generators to generate different kinds of topologies, and then evaluated them according to the existence of power laws. As a result they can say that the degree versus rank and the number of vertices versus degree power laws were not observed in all of the topologies. In this way the existence can be used to validate the accuracy of a generator. Power laws concerning the neighborhood size and the eigenvalues were found in all the generated topologies, but with different values of the exponent.

13.5 Chapter Notes

In 1959, Gilbert [244] introduced the model $(G_{n,p})$ in the sense of our second definition. In the same year Erdős and Rényi [181] presented a model parameterized by the number of vertices and the number of edges, n and m, which corresponds to the first definition we give, except that we fix the average vertex degree, $z = \frac{2m}{n}$. For a good introduction to this research area we refer to Bollobás [67, Chapter 7]. There Theorem 9 corresponds to our Theorem 13.1.1.

Our discussion about Local Search in Small Worlds is based on Kleinbergs pertinent work [360].

Further details on the exponential cutoff, and an evolutionary model that regards the exponential cutoff, are given in the very recent paper by Fenner, Levene, and Loizou [205].

In Bollobás et al. [68] further interesting behavior caused by certain initial choices of vertices and edges for the preferential attachment model by Barabási and Albert is given, and some more imprecisions are pointed out. For more details on the equivalence of (G_m^t) and (G_1^{tm}) see [68], too. Also, more explanations of other power law models and some mathematical background are given there.

Simple and efficient generators for standard random graphs, small worlds, and preferential attachment graphs are described in [43].

Generating functions are a concept for dealing with counting problems that is far more general than we present it here. Most books on combinatorics include a thorough discussion of generating functions (see for example [10]). A particular reference for generating functions only is [586], which can be downloaded at `www.cis.upenn.edu/~wilf/`. We mainly mathematically clarify the assertions found in [448]. There also further details on generating functions for bipartite graphs can be found.

The results in Section 13.2.3 are basically taken from Mihail et al. [420]. A proof for Theorem 13.2.4 is contained in [57, Chapter 6] (the original paper of Erdős and Gallai [180] is in Hungarian). The more precise Theorem 13.2.5 was firstly given and proven in Havel and Hakimi [288, 270], and it is stated as found in Aigner and Triesch [11]. For references on Markov-chain-processes see [420]. An overview of the algorithms for d-regular graphs can be found in Steger and Wormald [532]. They also construct a polynomial algorithm that works for all d and give an idea of how to implement that algorithm to obtain an expected running time in $\mathcal{O}(nd^2 + d^4)$.

We present the results given in Section 13.3.1 following A. Watts [572], though earlier work by Jackson and Wolinsky [323] prepared the ground.

The p^*-model is introduced in Section 10.2.5, and therefore it is omitted in this chapter.