

# 5


## Network Models

◇

**mod·el:** (*n.*) a simplified description, especially a mathematical one, of a system or process, to assist calculations and predictions.

We have seen that real networks of many different types share some common properties:

- They have short paths – it takes a few steps to go from any node to any other node.
- They have many triangles, reflected in high clustering coefficients.
- They have heterogeneous distributions of node and link variables, like degree and weights.



The next step in our investigation is to understand **where such properties come from. How do nodes choose their neighbors? How are hubs generated? How are triangles formed?** In this chapter we will answer these questions.

One way to study the **origin of network characteristics** is to **formulate a model** (i.e. a set of instructions used to assemble a network). The rules of a model incorporate intuitions, or hypotheses, about how network features emerge. By following the recipe of a model, we can build a network and compare it with real networks to see how they are similar or different. In this way, we can learn about the mechanisms that give rise to real-world networks.

cite!

Our exposition will trace the historical development of network science, presenting the classic models in order of their introduction. We discuss each model's failure to reproduce features observed in real networks, leading to the development of a new class of more realistic models. We will present simple mechanisms that allow us to generate model graphs having many basic features of real networks.

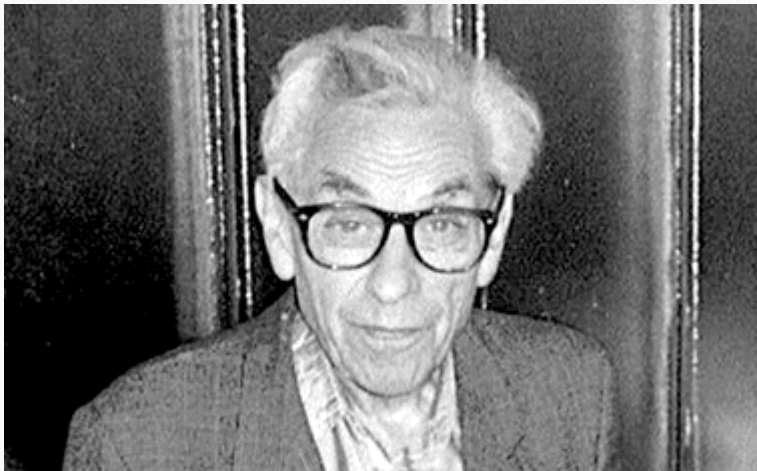
## 5.1 Random Networks

Suppose you have a bunch of disconnected nodes and wish to introduce some links. There are many ways to place the links between pairs of nodes. An “egalitarian” approach is to place them between randomly selected pairs of nodes. A network built in this fashion is called a *random* or *Erdős–Rényi* network (Box [5.1](#)). For simplicity, let us formulate the model in an equivalent version proposed by Gilbert. The **Gilbert model** has **two parameters**: the number of nodes  $N$  and a *link probability*  $p$ , describing how likely it is that a link is formed between any randomly chosen pair of nodes.<sup>[1](#)</sup>

### Box 5.1 Paul Erdős

The *Erdős–Rényi* random graph model is named after two mathematicians, Paul Erdős and Alfréd Rényi, who laid the foundations of random graph theory with several groundbreaking papers published jointly between 1959 and 1968.

Paul Erdős, shown in Figure 5.1, was an interesting character. He did not have a home, but he was not homeless. He would visit colleagues and stay at their homes while they worked on some mathematical problem together. Colleagues were happy to host him, as these visits were very productive professionally, often leading to prestigious scholarly publications. Once a theorem was proved or a paper written, Erdős would move to a new challenge, a new collaborator, and a new home.



**Fig. 5.1** The mathematician Paul Erdős in 1992. Image adapted from

[commons.wikimedia.org/wiki/File:Erdos\\_budapest\\_fall\\_1992.jpg](https://commons.wikimedia.org/wiki/File:Erdos_budapest_fall_1992.jpg) by Kmhkmh, used under CC BY 3.0 license.

Erdős worked on many different kinds of problems in addition to graph theory, and collaborated with over 500 colleagues. This makes him a hub in the math collaboration network, a social network discussed in Chapter 2.

The parameters of a random network model, as formulated by Gilbert, are the number of nodes  $N$  and the link probability  $p$ . The network can be constructed via the following procedure:

1. Select a pair of nodes, say  $i$  and  $j$ .
2. Generate a random number  $r$  between 0 and 1. If  $r < p$ , then add a link between  $i$  and  $j$ .
3. Repeat (1) and (2) for all pairs of nodes.

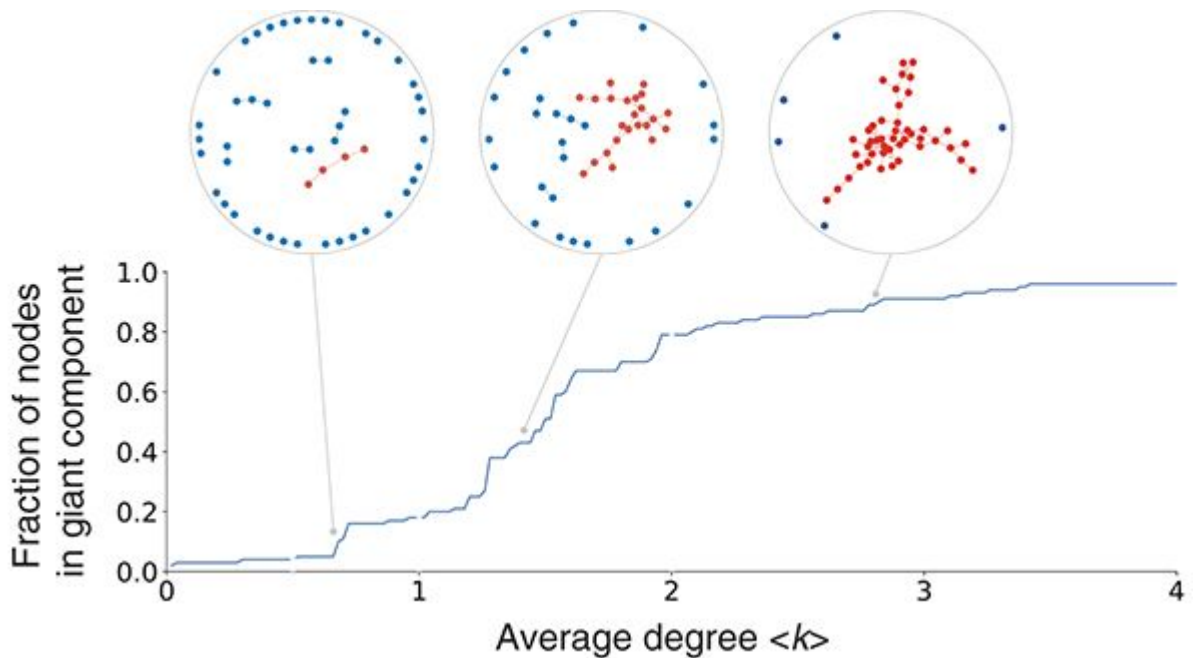
The main difference between the two formulations is that in the version by Erdős and Rényi the number of links of the network is fixed, whereas in the model by Gilbert it is variable. If we generate multiple networks following the procedure described in the above box, all using the same values for the number of nodes and the link probability, in general they will have different numbers of links, fluctuating around the average. However, when the number of

nodes is sufficiently large, the fluctuations in the number of links are small.

To see what random networks look like for different values of the link probability, imagine a very large set of nodes, without links. Naturally, the system is totally fragmented into *singletons* (i.e. isolated nodes). Now let's add links at random, one at a time. What is going to happen? Clearly, an increasing number of pairs of nodes will get connected, and through them connected subnetworks will be formed. At some point the network will become connected, so that it will be possible to go from any node to any other node by following links. Therefore there must be a transition from configurations in which all subnetworks are relatively small, to a configuration where at least one of the subnetworks contains almost all the nodes. It is natural to expect that the subnetworks will grow smoothly and that the transition will happen gradually. Instead, Erdős and Rényi discovered that this transition is abrupt, and takes place as a specific density of links is reached. A giant component forms when  $\langle k \rangle = 1$ , that is, when each node has one neighbor on average.

In Figure 5.2 we show some configurations of the Erdős–Rényi graph for different values of the average degree. The largest connected component is very small before the transition point and it grows rapidly with the average degree afterwards. The rest of the nodes are divided among small connected subnetworks. As the average degree gets even larger, the giant component “eats” all the remaining subnetworks and ends up including all nodes: the

network becomes connected. Appendix [B.2](#) presents a demonstration of the emergence of the giant component.



**Fig. 5.2** Evolution of random networks for growing values of the average degree  $\langle k \rangle$ , which corresponds to the process of adding links to the system one at a time. The largest connected component, highlighted in red, is very small when the average degree is lower than one. Around  $\langle k \rangle = 1$  a giant component grows very fast at the expense of the other, smaller components.

### 5.1.1 Density

Building a random network for a given value of the link probability is similar to the process of tossing a biased coin repeatedly and counting how many times we obtain heads or tails. The expected number of heads is proportional to the probability that the coin yields heads, and also proportional to the number of coin tosses. Analogously, the expected number of links in a random network is proportional to the link probability and to the number of node pairs.



Suppose a biased coin yields heads with probability  $p$ . For instance, if  $p = 0.1$ , we can expect that, on average, out of 10 tosses, we will obtain one heads and nine tails. If  $p = 0.5$ , we recover the familiar situation in which the coin is fair and we expect the same number of heads and tails. If  $p = 0$ , the coin never shows heads; if  $p = 1$  instead, it never shows tails. The expected number of heads out of  $t$  tosses is then  $pt$  (i.e. a fraction  $p$  of the tosses). In our random network model, the number of “tosses” corresponds to the number of possible pairs out of  $N$  nodes, which is  $\binom{N}{2} = N(N - 1)/2$ . Hence the expected number of links in a random graph is

$$\langle L \rangle = p \binom{N}{2} = \frac{pN(N - 1)}{2}. \quad (5.1)$$

Remembering Eq. (1.6) that expresses the average degree of a network as twice the number of links divided by the number of nodes, we get the expected average degree  $\langle k \rangle$  of a random network:

$$\langle k \rangle = \frac{2\langle L \rangle}{N} = p(N - 1). \quad (5.2)$$

Equation (5.2) tells us that the expected average degree of a node in an Erdős–Rényi network is the fraction  $p$  of its  $N - 1$  possible neighbors. Furthermore, plugging the expected number of links into the definition in Eq. (1.3), or plugging the expected average degree into Eq. (1.7), we find that the expected density of a random network is  $\langle d \rangle = p$ .

---

Intuitively, the link probability expresses the density of a random network: it is the expected ratio between the expected and maximum number of links. We know that real networks are usually sparse (i.e. they have very small average degree compared to the total number of nodes, and very small density). We conclude that for a random graph to be a good model of real networks, the link probability should be close to zero.

### 5.1.2 Degree Distribution

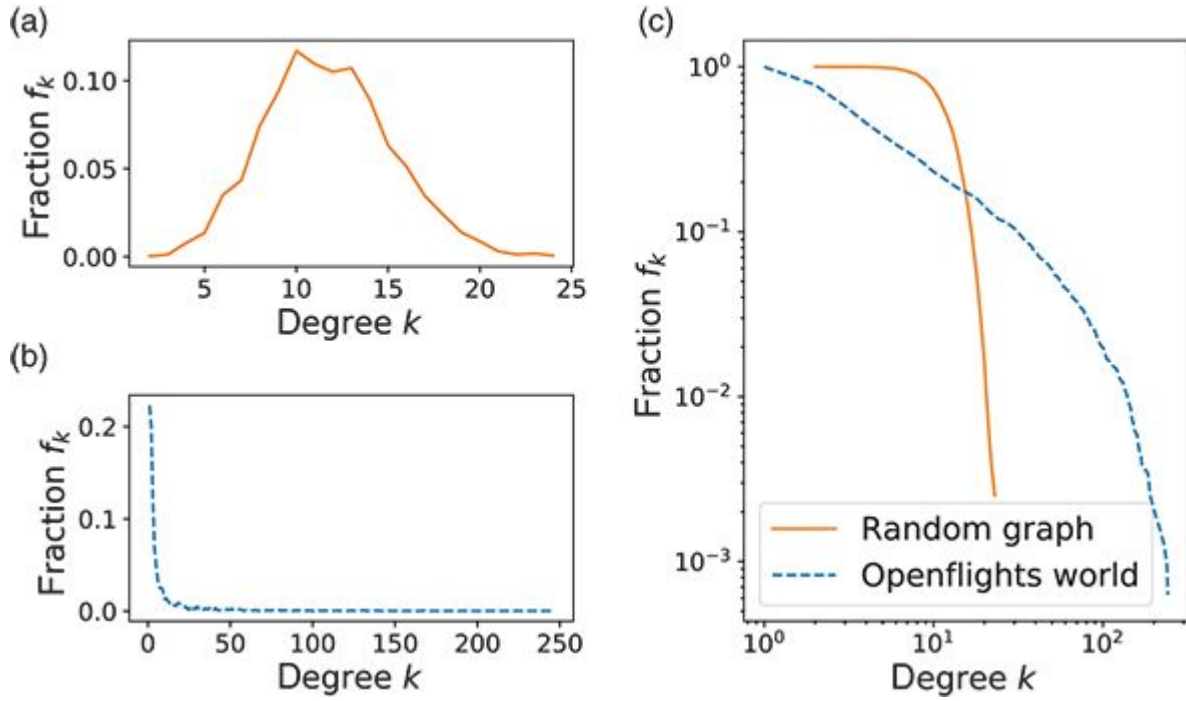
Suppose you have constructed a random network. What is its degree distribution? What we would like to know is the probability that a node has  $k$  neighbors. Since none of the nodes plays any special role in this model, we can just consider any one node, say  $i$ , and ask what is the probability that  $i$  has no neighbors, one neighbor, two neighbors, etc. Each of the remaining  $N - 1$  nodes of the network could be a neighbor of  $i$ . By design, the decision on whether or not a link is placed between  $i$  and each other node is independent of the existence (or absence) of other links elsewhere. Each pair involving  $i$  has probability  $p$  of being connected, regardless of the rest of the network.

We are back to a coin-tossing problem, with a biased coin and a total number  $N - 1$  of tosses. Our question turns into asking what the probability is that we obtain  $k$  heads out of  $N - 1$  tosses if the probability of obtaining heads in each toss is  $p$ . This is given by the *binomial distribution*:

$$P(k) = \binom{N-1}{k} p^k (1-p)^{N-1-k}. \quad (5.3)$$

In the limit of large  $N$  and constant (not too small)  $pN \approx \langle k \rangle$ , as in many real-world sparse networks, the binomial distribution is well approximated by a bell-shaped distribution with mean  $\langle k \rangle$  and variance  $\langle k \rangle$ : the average degree is a good statistical descriptor of the distribution.

The resulting probability distribution for the degree in a random network is a bell-shaped curve with a prominent peak concentrated around the average degree  $\langle k \rangle$ , and rapidly decaying on both sides of the peak [Figure 5.3(a)]. The degree of most nodes is close to the average degree, and large deviations from it are very unlikely.



**Fig. 5.3** Probability distribution of degree in a random network. (a) Degree distribution of an Erdős–Rényi random graph with the same number of nodes and links as the world flight network in our data collection:  $N = 3179$ ,  $L = 18,617$ . (b) Degree distribution for the world flight network. (c) Comparison between the two distributions in (a) and (b) on a double-logarithmic scale.

In Chapter 3 we have seen that the degree distribution of many real networks is rather different from such a distribution (Section 3.2), due to the presence of hubs (i.e. nodes with much larger degree than the average). In Figure 5.3(b) we plot the degree distribution of our world flight network. The heavy tail of the distribution covers over two orders of magnitude in degree; while many nodes have but a handful of neighbors, some hubs have hundreds. In Figure 5.3(c) we plot the distribution in double-logarithmic scale, and we compare it with that of Figure 5.3(a),

which corresponds to a random network with the same number of nodes and links. Clearly, the random network model does not provide a good description of the distribution: the nodes have approximately the same degree, so there are no hubs. Such discrepancy is one of the reasons why we need more sophisticated network models.

### 5.1.3 Short Paths

Let us check whether random networks have short paths. We can use a simple argument to explore this question. In the previous section we have seen that the nodes have approximately the same degree. Let us suppose that they all have degree 10. If we start from any node, there are 10 nodes attached to it. Each of them will also have 10 neighbors, and so on. So the number of reachable nodes grows exponentially with the number of steps: in two steps we can reach 100 nodes, in three steps 1000, and in a few steps we can reach every node in the network.

Assume the network is connected and all nodes have degree  $k$ . Within  $\ell = 1$  step we reach  $k$  nodes. Each of these has  $k - 1$  new neighbors if we exclude the root node we started from. So within  $\ell = 2$  steps we can reach as many as  $k(k - 1)$  nodes. Each of the new neighbors has in turn up to  $k - 1$  new neighbors, so within distance  $\ell = 3$  from the root we find  $k(k - 1)^2$  nodes, and so on. We conclude that, at distance  $\ell$  from the root, we find up to  $k(k - 1)^{\ell-1}$  nodes. If  $k$  is not too small, we can approximate  $k - 1 \approx k$  and the total number of nodes reachable within at most  $\ell$  steps from any node is approximately  $k^\ell$ . (This is actually an overestimate, because in reality neighbors of different nodes will occasionally coincide, whereas we are assuming that this never happens.) How far from a node do we have to go to reach all other nodes? The diameter  $\ell_{\max}$  such that the number of nodes reachable within at most  $\ell_{\max}$  steps from any node matches the total number of nodes  $N$  is given by

$$k^{\ell_{\max}} = N, \quad (5.4)$$

from which we obtain

$$\ell_{\max} = \log_k N = \frac{\log N}{\log k}. \quad (5.5)$$

It turns out that this is a good approximation of the diameter of the network, even when we consider neighborhood overlap and fluctuations in degree around  $\langle k \rangle$ . The slow



logarithmic growth of  $\ell_{max}$  with  $N$  indicates that distances within the network are small, even when the network size is very large.

The fact that the maximum distance to go from any node to any other node (the diameter) in a random network is small compared to the size of the network means that Erdős–Rényi networks indeed have short paths. To give a sense of how fast the number of reachable nodes grows with the distance from any node, let us consider the world’s network of social contacts and imagine that it is a random network. If we take  $k = 150$ , which is the average number of regular contacts that humans can maintain (*Dunbar’s number*), at distance five the number of reachable people is  $150^5 \approx 75$  billion, a factor of 10 larger than the world population. So, in principle we could reach any individual in five steps or less, which is compatible with the result of Milgram’s small-world experiment (Chapter [2](#)).

### 5.1.4 Clustering Coefficient

As you will recall from Chapter [2](#), the clustering coefficient of a node measures the fraction of the node's pairs of neighbors that are connected to each other. The presence of a link between two neighbors closes a triangle with the focal node, so the clustering coefficient can also be interpreted as the fraction of triangles centered at the focal node, or the probability of closing a triangle.

On a random network, the probability that a pair of neighbors of a node is connected is  $p$ , as the link probability is the same for every pair of nodes, regardless of their having common neighbors or not. Naturally, the clustering coefficients of individual nodes may deviate a bit from  $p$ , but the average value across all nodes is well approximated by  $p$ . We have observed in Section [5.1.1](#) that  $p$  is a very small number if we are to describe real, sparse networks via the Erdős–Rényi model. It follows that the average clustering coefficient of these networks is very small — the model creates triangles with extremely small probability. In contrast, we know that real social networks have high clustering (Section [2.8](#)). Therefore random networks are either unrealistically dense, or have unrealistically few triangles. We conclude that, if we wish to account for the remarkably high proportion of triangles observed in many real networks, we need a model with some specific rule to create triangles. Such models will be presented in Sections [5.2](#) and [5.5.3](#).

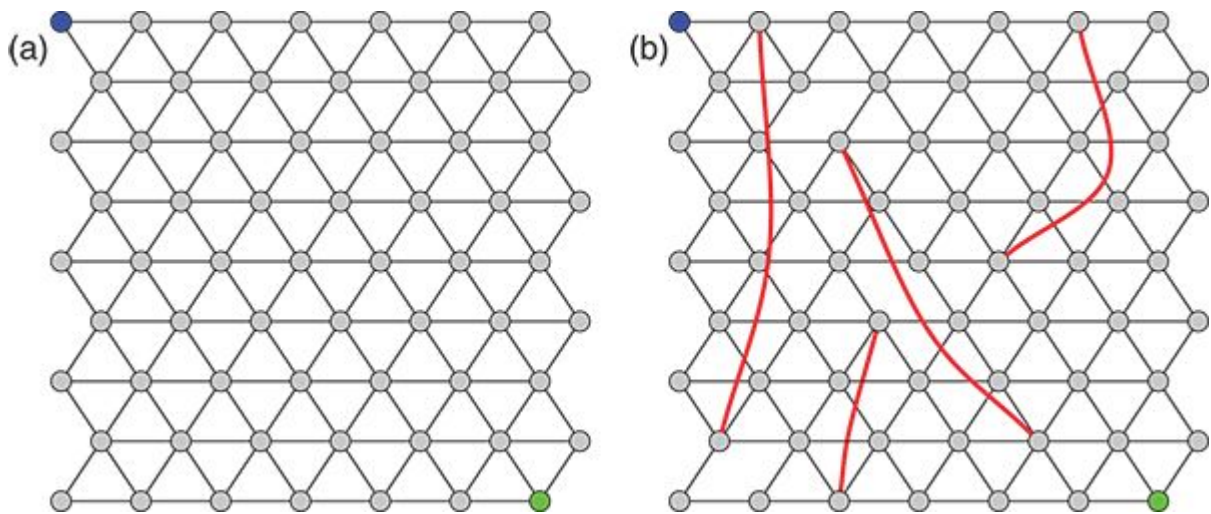
NetworkX has functions to generate random graphs according to the Erdős–Rényi and Gilbert models:

```
G = nx.gnm_random_graph(N,L) # Erdos-Renyi random graph  
G = nx.gnp_random_graph(N,p) # Gilbert random graph
```

## 5.2 Small Worlds

As we have seen in the previous section, real networks are different from random ones. Erdős–Rényi networks do have short paths, but triangles are rare, resulting in average clustering coefficient values that may be orders of magnitude smaller than those measured in real networks.

In the late 1990s, Duncan J. Watts and Steven H. Strogatz introduced the *small-world model*, also known as the *Watts–Strogatz model*, which generates networks with both features — short paths and high clustering. Their idea is to start from a grid-like network where all nodes have the same number of neighbors, like the hexagonal lattice in Figure 5.4(a). Such a network has a high average clustering coefficient, as any pair of consecutive neighbors of each node are connected, forming a triangle with the node.



**Fig. 5.4** Small-world networks. (a) A hexagonal lattice, a graph where each node has six neighbors (except on the boundary). There

are many triangles, so nodes have a high clustering coefficient. Paths from one corner to another have to traverse many links, therefore the average shortest-path length is large. (b) Four links have been rewired to randomly selected nodes, which are typically further away from the original endpoints. These links (in red) are shortcuts, and allow us to reach remote parts of the network via a low number of hops. For instance, the shortest path from the blue node to the green node goes from 10 steps on the lattice down to six steps through one of the shortcuts. Since only a few triangles are disrupted by the rewiring procedure, the clustering coefficient remains high.

The internal nodes have degree  $k = 6$  and clustering coefficient  $C = 6/\binom{6}{2} = 6/15 = 2/5$ . Border nodes have smaller degree  $k = 4, 3, 2$  and even higher clustering coefficients, respectively  $C = 3/\binom{4}{2} = 1/2$ ,  $C = 2/\binom{3}{2} = 2/3$ , and  $C = 1/\binom{2}{1} = 1$ . Therefore the average clustering coefficient is at least  $2/5$  and converges to  $2/5$  in the limit of infinite lattice ( $N \rightarrow \infty$ ).

In contrast, the network has a high average shortest-path length. For example, nodes on opposite sides of the lattice can only be reached via paths traversing many links. The distances between nodes, however, could be reduced considerably by creating a few *shortcuts* — links joining portions of the network originally far from each other, like the red links in Figure 5.4(b). This can be done by selecting some of the initial links at random, preserving one of their endpoints and replacing the other endpoint with a node selected at

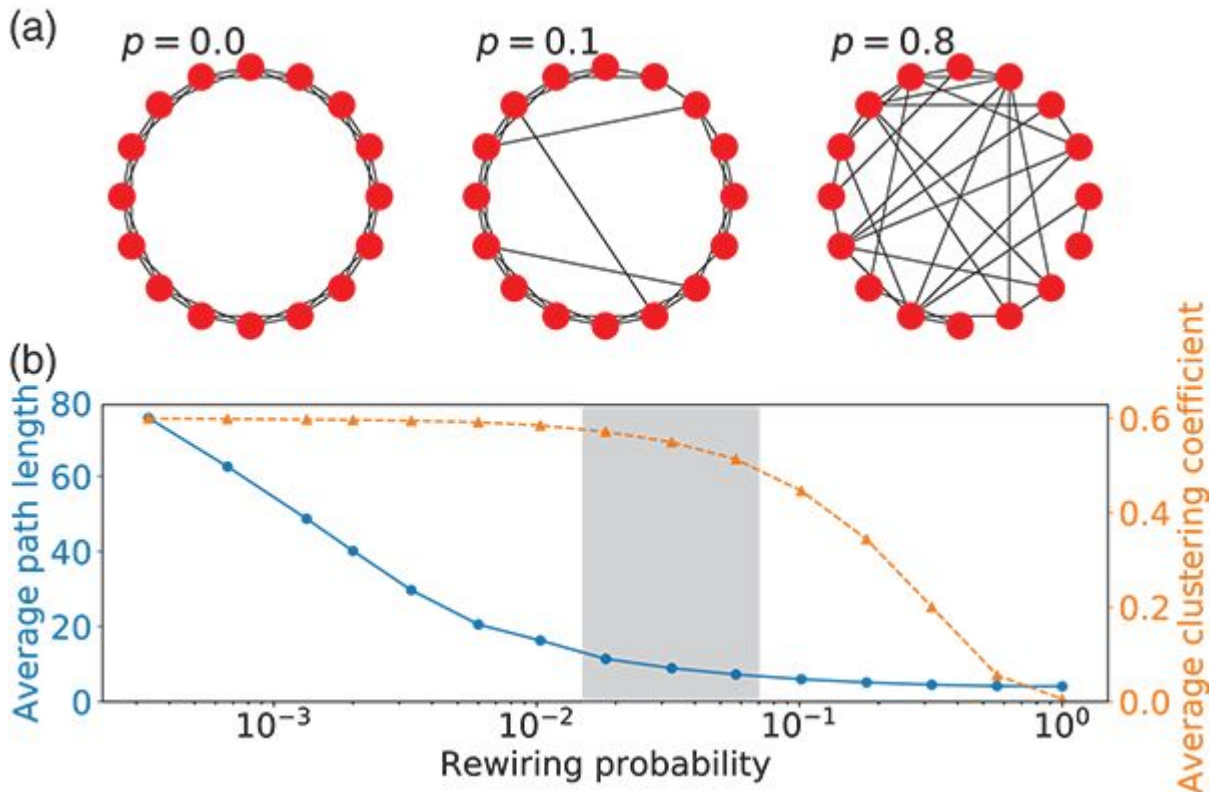
random among all other nodes. Formally, the rewiring procedure applies to each link of the network with a *rewiring probability*  $p$ .<sup>2</sup> The number of rewired links is proportional to the rewiring probability.

The expected number of rewired links is  $pL$ , where  $p$  is the rewiring probability and  $L$  is the total number of links in the network. The special case  $p = 0$  corresponds to the initial lattice, and the special case  $p = 1$  yields a random network.

If the rewiring rate is very small (close to zero), little happens. If it is very large (close to one), the network becomes a random network á la Erdős and Rényi, as basically all links are rewired to random nodes, which is equivalent to placing links between randomly chosen pairs of nodes. In this scenario most triangles are destroyed and the clustering coefficient becomes very small. But if  $p$  is chosen to be neither too small nor too large, it is possible to achieve a trade-off, where there are enough shortcuts to make the distances considerably smaller on average, but not so many as to disrupt most of the triangles. In this regime, the paths are as short as those in random networks, while the average clustering coefficient decreases only slightly with respect to the initial lattice configuration, so that it can be comparable with that of real social networks.

In Figure [5.4](#) we start from a hexagonal lattice, but any network with high clustering coefficient can be used as an initial configuration. In their pioneering paper, [Watts and Strogatz \(1998\)](#)

imagined a ring in which each node is connected to its  $k$  nearest neighbors. They considered the case  $k = 4$ , as shown in Figure 5.5(a). In this case the initial clustering coefficient is  $C = 1/2$ , because each node's neighbors form three triangles out of six possible. This is very high. One can rewire a link so that one of its attached nodes keeps the link while the other end of the link is attached to a randomly chosen node; this is the formulation in the original model. Alternatively, one can replace a link by connecting two random nodes, irrespective of their degree. Yet another variation is that random links can simply be added to the network, instead of rewiring the existing ones.



**Fig. 5.5** Small-world model. (a) In the standard configurations of Watts–Strogatz model networks, we start from a ring lattice (left),



where each node is connected to its four nearest neighbors, and progressively add shortcuts by rewiring links. (b) Decrease of the average shortest-path length and clustering coefficient as a function of the rewiring probability  $p$ . The extreme  $p = 0$  is a lattice network as in the leftmost graph above, but with  $N = 1000$  nodes. The extreme  $p = 1$  is a random network with the same number of nodes and links. The shaded area highlights the values of  $p$  such that the average path length is almost as short as that of a random network, while the clustering coefficient is still almost as large as that of the lattice.

Figure 5.5(b) plots the average shortest-path length  $\langle \ell \rangle_p$  and clustering coefficient  $C_p$  as a function of the rewiring probability  $p$ . There is a range of rewiring probability values between  $p \approx 0.01$  and  $p \approx 0.1$  (highlighted in Figure 5.5(b)) in which  $\langle \ell \rangle_p \approx \langle \ell \rangle_1$  and  $C_p \approx C_0$ . In other words, the average shortest-path length from the model is close to that of an equivalent random network, and much lower than that of the lattice. At the same time, the clustering coefficient from the model is still close to that of the lattice, and much larger than that of the random network. Therefore, the Watts–Strogatz model is indeed capable of generating — with a suitable amount of randomness — networks endowed with two desired features: short paths and high clustering. A demonstration of this is presented in Appendix B.3.

The model cannot generate hubs, however. The degree distribution transitions from that of the initial lattice, where all nodes have identical degree, to that of a random network with the same number of nodes and links, where the degrees are concentrated



around a characteristic value [Figure [5.3\(a\)](#)]. Hence, for any value of the rewiring probability  $p$ , all nodes have similar degree; none accumulates a disproportional fraction of links. We need some other model ingredient to account for the emergence of hubs.

NetworkX has a function to generate graphs according to the small-world model of Watts and Strogatz:

```
G = nx.watts_strogatz_graph(N,k,p) # small-world model network
```

### 5.3 Configuration Model

Let us focus on networks with realistic degree distributions. In Section [5.4](#) we will explore the mechanisms responsible for the existence of hubs. But first, let us answer the following question: given some degree distribution, can we construct a network whose nodes have exactly that degree distribution?

A simple solution is provided by the *configuration model*. This model actually pursues a more ambitious goal: generating a network whose nodes have an arbitrary *degree sequence*, with node 1 having degree  $k_1$ , node 2 degree  $k_2$ , and so on (Box [5.2](#)). A degree sequence could be produced from a particular distribution we are interested in reproducing, or it could be taken from the nodes of a real network. Once we reproduce the sequence of all node degrees, we must have reproduced the corresponding degree distribution as well. In contrast, many degree sequences correspond to the same distribution. For example, two networks with distinct degree sequences (1,2,1) and (1,1,2) have the same degree distribution.

### Box 5.2 Degree Sequences

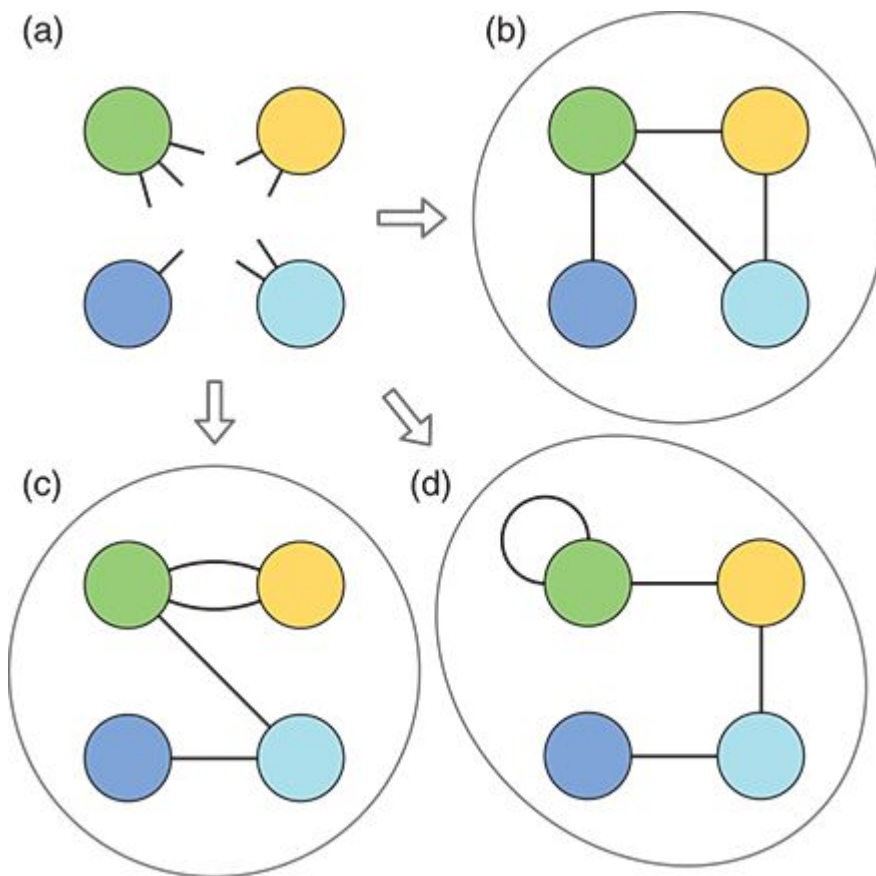
The degree sequence of a network is the list of degrees of its nodes, in the order of their labels. The degree sequence is a list of  $N$  numbers  $(k_0, k_2, k_3, \dots, k_{N-1})$ , where  $k_i$  is the degree of node  $i$ . Note that the degree sequence determines the degree distribution, but the reverse is not true. Every permutation of a degree sequence leads to the exact same distribution, as for the distribution it does not matter which node has which degree, only how many nodes have a given degree.

Suppose we have a set of nodes and their degree sequence. The first step is to assign to each node a number of *stubs* corresponding to the degree of the node, as in Figure 5.6(a). A stub is just a dangling link having the node as one of its endpoints, but not yet connected to a neighbor. The network is then constructed by the following iterative steps:

1. A pair of stubs is selected at random.
2. The chosen stubs are joined to each other, forming a link between the nodes attached to the stubs.

This routine is repeated until all stubs are joined in pairs. Naturally, for this to happen, there needs to be an even number of stubs (i.e. the sum of the degrees in the target sequence must be even). We see why this procedure achieves our aim: if a node has  $k$  stubs attached to it, it will eventually have  $k$  neighbors. Since the number of stubs

attached to each node equals its degree, each node ends up having the desired degree. As illustrated in Figure 5.6(b–d), multiple networks can be created this way, depending on the sequence of pairs of stubs that are combined. Some outcomes may not be desirable if they violate constraints. For example, one may wish to exclude networks with multiple links between two nodes [Figure 5.6(c)] or with self-loops [Figure 5.6(d)].



**Fig. 5.6** Configuration model. (a) We start with nodes and stubs corresponding to a given degree sequence. (b–d) We can connect the stubs in different ways, leading to different networks with the given degree sequence.

The way links are formed is random by construction. So, the configuration model generates random networks with a prescribed degree sequence. This turns out to be very useful in the analysis of networks. In Chapter 3 we have seen that broad degree distributions are responsible for a number of peculiar properties and effects. However, there are network features that do not depend on the degree distribution alone.

Given a network, we might question whether a specific property is explained by the degree distribution alone. Through the configuration model we can generate randomized, or *shuffled*, versions of the network having the same degree sequence. Each of these configurations is a *degree-preserving randomization* of the original network, in that it preserves the degree sequence, but everything else is otherwise totally random. Now we can check whether the feature of interest is present in the shuffled network configurations. If it is there, then the feature must result from the degree distribution alone; otherwise there must be other factors behind it.

For instance, suppose that the feature we would like to investigate is the average clustering coefficient: can the clustered structure of a real social network be explained by its degree distribution? What we need to do is compute the clustering coefficient of a sufficient number of random configurations, derive the average and standard error, and check whether the value of the measure in the original graph is compatible with the estimate from the shuffled networks, within the error. If it is, we deduce that the triangles present in the network exist simply because of degree

constraints. If it is much larger than the random estimate, as usually happens, the linking patterns of the network cannot be random; they must result from some mechanism that favors the formation of triangles.

NetworkX has a function for generating a network with a prescribed degree sequence via the configuration model:

```
G = nx.configuration_model(D) # network with degree sequence D
```

The configuration model generates all possible networks having a given degree sequence, but we might as well impose other constraints. For instance, we could be interested in exploring all networks having a given number of triangles. The idea of generating networks with specific characteristics has led to the development of a broad class of network models called *exponential random graphs* (Box [5.3](#)).

### Box 5.3 Exponential Random Graphs

It is of interest to study randomly generated networks sharing some common quantitative features, while differing in their detailed structure. On the one hand, they represent potential alternatives to the specific network configurations we encounter in the real world. On the other hand, they allow us to investigate the interplay between different structural properties. For instance, we might ask which values of the average clustering coefficient are compatible with a specific value of the density.

*Exponential random graphs* are classes of random networks subject to constraints. We define a class of networks based on a set of  $M$  network measures,  $x_m$ ,  $m = 1, \dots, M$ . We impose a constraint for each measure  $x_m$ : the average over all networks of the class must equal a specific value, say  $\langle x_m \rangle = x_m^*$ . Exponential random graphs are networks that satisfy these constraints while maximizing randomness. As it turns out, this allows us to define the probability  $P(G)$  of selecting a network  $G$  of the class having measure values  $x_1(G), x_2(G), \dots, x_M(G)$ :

$$P(G) = \frac{e^{H(G)}}{Z}, \quad (5.6)$$

with

$$H(G) = \sum_{m=1}^M \beta_m x_m(G), \quad (5.7)$$

where  $\beta_m$  is a parameter associated with measure  $x_m$ . The function  $Z$  ensures that  $P(G)$  is a probability, so that  $\sum_G P(G) = 1$ .

Through the probabilities in Eq. (5.6) one can calculate the average of any network measure. In particular, we can express every constraint by setting the average of  $x_m$  to its desired value:

$$\langle x_m \rangle = \sum_G P(G) x_m(G) = x_m^*, \quad (5.8)$$

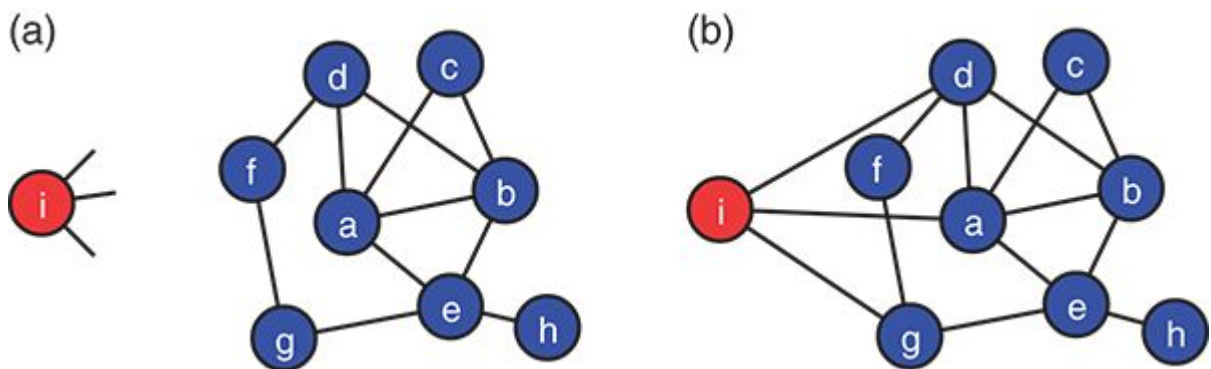
where the sum runs over all networks of the class. This yields a set of  $M$  equations with  $M$  variables, the parameters  $\beta_m$ . Solving these equations yields the values of the parameters. This specifies the model, which can then be used to calculate the average of any variable of interest. Although the constraints are imposed on averages, it turns out that for most exponential random graphs in a class, the value of any measure is close to its average.

The random network model, in the formulation by Gilbert (Section 5.1), is a special version of an exponential random graph with the single constraint that the networks must have a given average number of links.



## 5.4 Preferential Attachment

The models explored so far are **static**. By that we mean that all the nodes of the network are there from the beginning; all we do is add (or rewire) links between them. Real networks, instead, are usually **dynamic**. Nodes and links appear and disappear. If we consider popular networks such as the Internet, the Web, Facebook, and Twitter, we notice that their size has been **growing**. Nodes may also disappear (e.g. old routers of the Internet or old Web pages). But the introduction of new nodes is more likely. This is the reason why realistic dynamic models typically incorporate some form of *network growth*. The dynamic procedure starts from an initial configuration, usually a very small clique of nodes. Then nodes are added one by one. Each new node is attached to a number of old nodes based on some rule, which is characteristic of the model (Figure 5.7).



**Fig. 5.7** Network growth. The construction of a network is usually dynamic, with new nodes being added and connected to older nodes. (a) A new node **i** with three stubs is added to the system. (b)

Each stub is attached to an older node, according to some rule, and the new node is incorporated in the network.

Another limitation of the network models considered until this point is that they cannot explain the existence of **hubs**. To be more precise, the configuration model can generate hubs, but only by deciding *a priori* the degree of the nodes — that is not helpful in explaining how hubs emerge in the real world. The **random network and small-world models do not give rise to hubs**. The main reason is that in both cases the linking rules are basically egalitarian — the nodes choose their neighbors totally at random. This way, it is extremely unlikely for any one node to have an advantage over the other nodes and end up with many more neighbors than the rest. If we want to recover the hubs, it is thus necessary to **introduce a mechanism that favors some nodes over others**. Such a mechanism is called preferential attachment: the higher the degree of a node, the more links it will receive.

The underlying idea is simple. Suppose that you create a new Web page and wish to include some links to other pages. Our knowledge is necessarily limited to a tiny fraction of the trillions of pages on the Web. Most pages of which we are aware are likely to be popular, and as such they are linked from many other pages. Incidentally, this could be the reason why we discovered them: if a page has incoming links from many documents, it is more likely to be found via Web surfing or a search engine. Therefore, our choice of documents to link from our new Web page will favor popular and highly linked pages. Likewise, when we write a scientific article and

we compile its bibliography, it is common to refer to papers that are frequently cited by other authors, as we are likely to bump into them when we read other articles and scroll through their references.

In the language of networks, a popular node is one with high degree, indicating that it has **many neighbors**. Preferential attachment means that high-degree nodes have a high probability of receiving new links. Such a criterion has been known in various contexts and under multiple names (Box [5.4](#)). The best-known network growth model incorporating preferential attachment was proposed by Barabási and Albert in 1999, and is known as the *Barabási–Albert model*, or *BA model*, or *preferential attachment model*. It is a simple combination of growth and preferential attachment. At each step, a new node is added and connected to some existing nodes. The probability that a new node is attached to an old node is *proportional* to the degree of the old node. In this way, a node with degree 100 is 100 times more likely to receive a new link than a node with degree one. In Figure [5.7](#), for example, the chances that node **a** receives a link from node **i** would be double the chances of node **c** under preferential attachment.

### Box 5.4 Preferential Attachment

The principle behind preferential attachment is simple: **the more you have, the more you will receive**! It is ancient, too. The first known reference can be found in the Gospel of Matthew (25:29): “*For to every one who has will more be given, and he will have abundance; but from him who has not, even what he has will be taken away.*” In this quote, the principle is summarized in the first sentence, the other one is its symmetric counterpart, stating that the less one has, the less one will have in the future. So *the rich get richer* and *the poor get poorer*. Preferential attachment is therefore also called the **Matthew effect**. Another common name is *cumulative advantage*.

The first scientific implementation of this principle was Pólya's *urn model*, which works as follows. An urn contains  $X$  white and  $Y$  black balls; one ball is drawn randomly from the urn and put back in, along with another ball of the same color. If  $X$  is much larger than  $Y$ , it is more likely that we extract a white ball than a black ball. If indeed we pick a white ball, at the end of the round there will be  $X + 1$  white balls and  $Y$  black ones, which will give an extra advantage to white in the next round. Hence, the number of white balls will increase faster than the number of black balls.

**Preferential attachment** has been used to explain heavy-tailed degree distributions of many different quantities, like the number of species per genus of flowering plants, the

number of (distinct) words in a text, the populations of cities, individual wealth, scientific production, citation statistics, and firm size, among others. Preferential attachment models were introduced by George U. Yule, Herbert A. Simon, Robert K. Merton, Derek de Solla Price, Albert-László Barabási, and Réka Albert.

We start from a complete graph with  $m_0$  nodes. Each iteration of the algorithm consists of two steps:

1. A new node  $i$  is added to the network, with  $m \leq m_0$  new links attached to it. The parameter  $m$  is thus the average degree of the network.
2. Each new link is wired to an old node  $j$  with probability


$$\Pi(i \leftrightarrow j) = \frac{k_j}{\sum_l k_l}. \quad (5.9)$$

The denominator in Eq. (5.9) is the sum of the degrees of all nodes (except  $i$ ), and guarantees that the sum of all probabilities equals one, as it must be.

The procedure is repeated until the network reaches the desired number of nodes  $N$ . Box 5.5 shows how to select a node with the desired probability in Python.

### Box 5.5 Random Selection with a Probability Distribution

It is often necessary to randomly choose nodes with probability proportional to some quantity. For example, in the case of a random network, we select a node to attach a link with uniform probability, meaning that each node has the same chance of being selected. In Python we can do this using the `random` module:

```
nodes = [1, 2, 3, 4]
selected_node = random.choice(nodes)
```

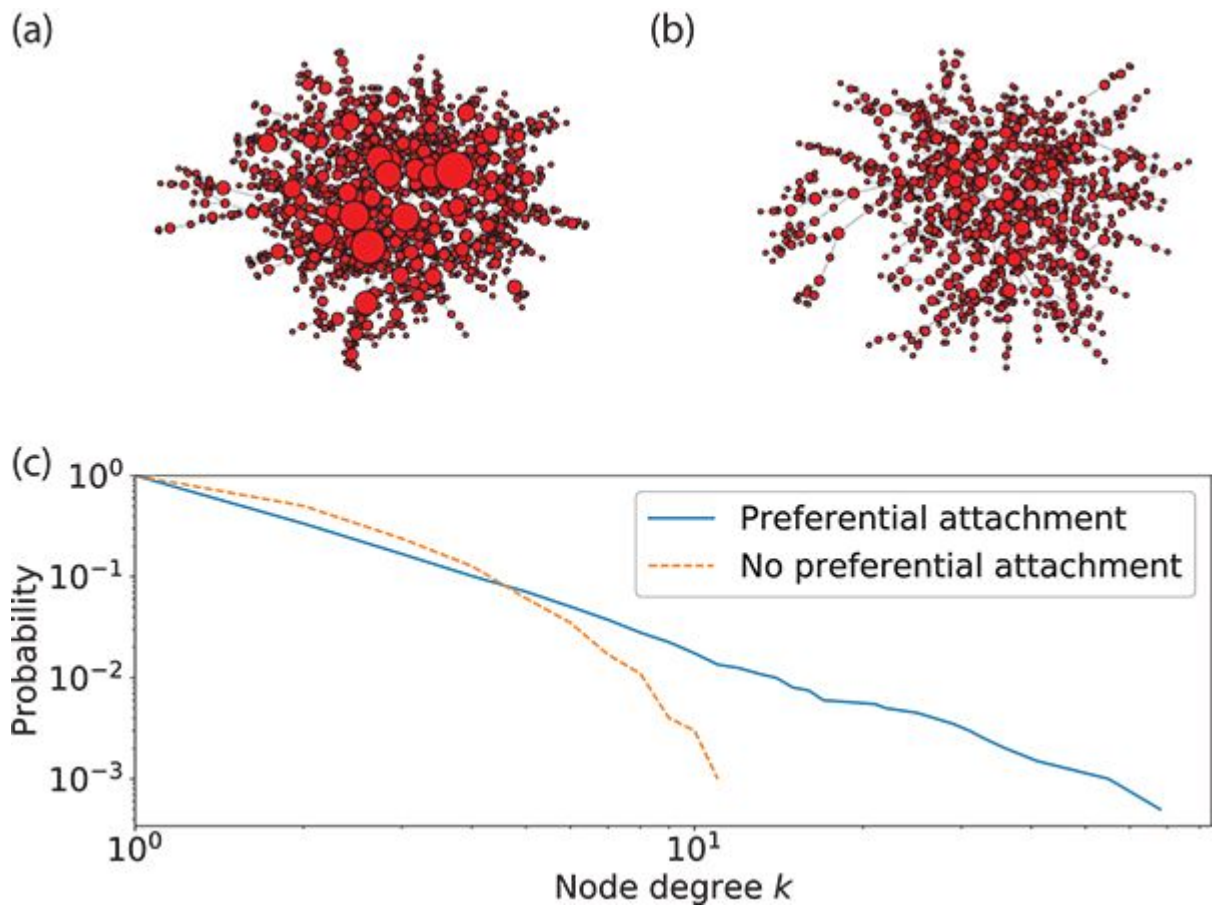
In other cases we need to select nodes with different probabilities. For example, in the case of preferential attachment (Section [5.4](#)), at each step we need to choose a node with probability proportional to its degree. Or in the case of the fitness model (Section [5.5.2](#)), selection is weighted by some more complicated function of degree and fitness. Fortunately, dealing with these cases is also easy as of Python 3.6. All we have to do is provide a second argument with a list of weights associated with the nodes. Suppose we wish to select a node according to its degree, as in preferential attachment. We can use the degrees as weights:

```
nodes = [1, 2, 3, 4]
degrees = [3, 1, 2, 2]
selected_node = random.choices(nodes, degrees)
```

Node 1 ( $k = 3$ ) is three times more likely to be selected than node 2 ( $k = 1$ ). The `random.choices()` function allows one to randomly select from a population based on any given set of weights. The weights can be probabilities from a distribution, but it is not necessary — they do not need to sum to one. And they do not have to be integers. Do take care to ensure that the population and the weight sequences are *aligned*: the  $i$ th element in the population must correspond to the  $i$ th element in the sequence of weights.

At the beginning, all nodes have equal degree by construction. While new nodes and links are added to the system, the degrees of the nodes grow. However, the oldest nodes are there from the beginning, so they can receive links at any time, in contrast to nodes that enter the game much later. So the degrees of the oldest nodes exceed those of newer nodes, and this makes the former even more likely to attract new links in the future, at the expense of the latter, because of preferential attachment. Such *rich-gets-richer* dynamics generate the desired heterogeneity in the degree distribution, with the oldest nodes becoming network hubs. In Figure [5.8\(a,c\)](#) we show a network constructed with the BA model, along with its degree distribution. We observe a heavy-tailed distribution, which confirms the existence of hubs. Appendix [B.4](#) presents a demonstration of the model.





**Fig. 5.8** Preferential attachment. (a) Network generated with the BA model. It has  $N = 2000$  nodes and average degree  $\langle k \rangle = 2$ . The size of a node is proportional to its degree; the large nodes are the hubs. (b) Network generated via a similar growth model, but with random rather than preferential attachment. There are no hubs. (c) Cumulative degree distribution of the networks in (a) and (b). The BA model generates a broad distribution, while the absence of preferential attachment leads to a narrower distribution, without hubs.

At this point you might wonder whether, for hubs to emerge, it is sufficient to have growth, without preferential attachment. After all, the initial nodes will have more time to collect links, whatever



the linking criterion. For instance, suppose that each new node can pick as neighbor any randomly chosen node, regardless of its degree. Like before, we expect that the older the nodes, the larger their degrees. This is true, but we can see in Figure 5.8(b,c) that in this case the values of the degrees are not very different from each other, and the corresponding distribution does not have a heavy tail.

We conclude that the combination of growth and random node selection does not do the job; preferential attachment is needed.

Indeed, empirical studies have confirmed that preferential attachment is at work in the growth of many real networks.

NetworkX has a function to generate graphs according to the BA model:

```
G = nx.barabasi_albert_graph(N,m) # BA model network
```

## 5.5 Other Preferential Models

The BA model uses *linear preferential attachment*, in that the link probability is strictly proportional to the degree of the target node. Suppose that we relax this rule, and let the probability vary as some power of the degree. We call this *non-linear preferential attachment*.

The extension of the BA model using non-linear preferential attachment is identical to the original BA model, except that Eq. (5.9) expressing the probability that an old node  $j$  receives a link from a new node  $i$  is given by

$$\Pi_{\alpha}(i \leftrightarrow j) = \frac{k_j^{\alpha}}{\sum_l k_l^{\alpha}}, \quad (5.10)$$

where the exponent  $\alpha$  is a parameter. For  $\alpha = 1$  we recover the standard BA model. What happens when  $\alpha \neq 1$ ? There are two different scenarios:

1. If  $\alpha < 1$ , the link probability does not grow with degree as fast as in the BA model, so the advantage of high-degree nodes over the others is not as big. As a result, the degree distribution does not have a heavy tail — the hubs disappear!
2. If  $\alpha > 1$ , high-degree nodes accumulate new links much faster than low-degree nodes. As a consequence, one of the nodes will end up being connected to a fraction of all other nodes. The effect is even more extreme when  $\alpha > 2$ , in which case we observe a *winner-takes-all* effect: a single node may be connected to all other nodes, which have approximately the same, low degree.

Depending on the value of the exponent that expresses the power of the degree, we either end up without hubs (sub-linear

preferential attachment) or with one super-hub (super-linear preferential attachment). Either way, non-linear preferential attachment fails to generate hubs as observed in real-world networks; linear preferential attachment is the only way to go. This exposes a fundamental fragility of the BA model, as the necessity of strict proportionality between link probability and degree appears unrealistic. Luckily, as we will see in Section [5.5.4](#), there are natural linking mechanisms that implicitly induce linear preferential attachment.

In addition to the dependency on linear preferential attachment, the BA model has other limitations:

- It yields a fixed pattern for the degree distribution. The slope of the preferential attachment curve in Figure [5.8\(c\)](#) is the same for any choice of the model parameters. Real degree distributions could decay faster or more slowly.
- The hubs are the oldest nodes; new nodes cannot overcome them in degree.
- It does not create many triangles. The average clustering coefficient is much lower than in many real networks. ✓
- Nodes and links are only added; in real networks they can also be deleted.
- Since each node is attached to older nodes, the network consists of a single connected component. Many real networks have multiple components.

Next we present more sophisticated models of network growth that address some of these limitations.

### 5.5.1 Attractiveness Model

Preferential attachment has a subtle pitfall: what happens if a node has no neighbors? Its degree is zero, so the probability that it will get links is also zero: the node will never have neighbors! So, if we started from an initial core of nodes with no neighbors, the BA model would collapse, as the new nodes could not be attached to any of the old nodes. The standard initial configuration of the BA model consists of a complete graph, so every node has neighbors and this problem does not occur, but ideally a model should work with different choices of the initial condition. If we consider directed networks, and assume that the link probability depends only on the in-degree, the problem occurs irrespective of the initial configuration. Each new node has initially in-degree zero, as it comes with outgoing links and can only receive incoming links from newer nodes. Therefore no new nodes can receive incoming links.

Fortunately, there is a simple way out of this. Instead of having a link probability that is strictly proportional to the degree, we can modify the rule slightly. The idea, originally proposed by Derek de Solla Price in the context of citation networks, is that a node receives links because of its degree, but also because it has an intrinsic attractiveness. In the *attractiveness model*, the link probability is *proportional to the sum of degree and a constant attractiveness*.

The attractiveness model is a slightly modified version of the original BA model, in which Eq. (5.9) expressing the probability that an old node  $j$  receives a link from a new node  $i$  is replaced by

$$\Pi(i \leftrightarrow j) = \frac{A + k_j}{\sum_l (A + k_l)}, \quad (5.11)$$

where  $A$  is the attractiveness parameter and can take any positive value. The case  $A = 0$  yields the BA model.

For any value of the attractiveness parameter  $A$ , the model builds networks with heavy-tailed degree distributions. The slope of the distribution depends on  $A$ . This way, the model is able to match degree distributions of multiple real networks, unlike the BA model.

### 5.5.2 Fitness Model

As we have seen in Section [5.4](#), in the BA model the hubs are also the oldest nodes. This feature is not realistic. In the Web, for example, there can be pages that are created long after others, but that end up being more popular and attracting more hyperlinks. Take Google, for example. It was created in 1998, when there were already millions of sites, but it ended up being the most popular Web hub. Likewise, in the scientific literature the most cited papers are not the oldest ones: occasionally new groundbreaking papers surpass many earlier publications.

This happens because nodes (websites, papers, social media users, and so on) have their own individual appeal that may boost the rate at which they accrue links, giving them an edge over much older nodes. Such appeal is only partly, and indirectly, reflected by their degree. The attractiveness parameter of the model described in the previous section is the same for all nodes, so it does not allow us to discriminate among nodes and to introduce discrepancies in their rate of degree growth. Therefore, in the attractiveness model, as in the BA model, hubs are still the oldest nodes.

To allow for the possibility that new nodes become hubs, Bianconi and Barabási proposed a *fitness model* in which each node has its own individual appeal, called *fitness*. The fitness values are intrinsic features of the nodes; they do not change over time. The link probability is proportional to the *product* of the degree and the fitness of the target node.



The fitness model is similar to the BA model, but each node  $i$  is assigned a fitness value  $\eta_i > 0$  generated from some distribution  $\rho(\eta)$ . Then at every step, each new link from a new node  $i$  is wired to an old node  $j$  with probability

$$\Pi(i \leftrightarrow j) = \frac{\eta_j k_j}{\sum_l \eta_l k_l}. \quad (5.12)$$

If all nodes have identical fitness, the model reduces to the BA model, as the constant  $\eta$  is a factor that cancels out between numerator and denominator of Eq. (5.12), returning the standard prescription of preferential attachment.

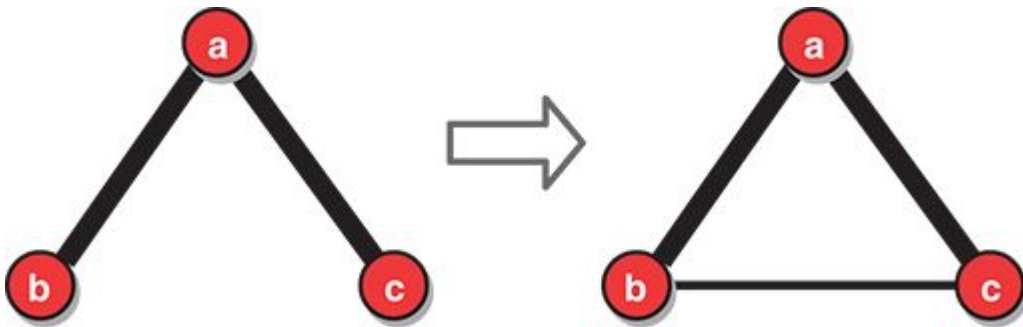
If the fitness distribution  $\rho(\eta)$  has *infinite support* (i.e.  $\eta$  can take arbitrarily large values), then there is a winner-takes-all effect with the highest-fitness node being linked to most nodes. But if the fitness distribution  $\rho(\eta)$  has *finite support* (i.e.  $\eta$  has a finite maximum value), then the degree distribution of the model has a heavy tail. An example of this case is the uniform distribution in the unit interval. In Python we can draw fitness values with uniform distribution using the `random()` function.

The fitness model generates networks with two desired properties. First, as long as the fitness values are bounded, the network has multiple hubs. Second, high fitness allows a node to compete with all its peers, regardless of their age and status. This is because the nodes increase their degree at a rate determined by their

individual fitness. Therefore the nodes with the largest fitness values eventually achieve the largest degrees, regardless of when they are introduced in the system.

### 5.5.3 Random Walk Model

Networks built with the BA model have very low clustering coefficients. To see why, recall that in order to have many triangles, it is necessary that links join pairs of nodes with at least one common neighbor. For instance, if nodes **b** and **c** are both attached to **a**, a link between **b** and **c** will close the triangle **abc** (Figure 5.9). In the BA model, however, the probability of a node receiving a link is proportional to its degree, regardless of whether the new pair of neighbors have a common neighbor or not. This is why triangles are formed rarely. To increase the production of triangles it is necessary to introduce a mechanism that favors the creation of links between nodes with common neighbors.



**Fig. 5.9** Strong triadic closure. An individual **a** has strong connections, indicated by the thick links, with **b** and **c**. According to Granovetter's principle of strong triadic closure (Box 5.6), there must be or will eventually be at least a weak connection between **b** and **c**.

The formation of a triangle via the addition of a link is called *triadic closure* and is perhaps the main mechanism explaining the formation of links in social networks (Box 5.6). That should not be

surprising: many people we know have been introduced to us by common acquaintances. The mechanism has several model implementations. Here we discuss a very intuitive one called the *random walk model*. The idea is that in addition to creating random connections, we also connect to a new neighbor's neighbors — in a social network, to a new friend's friends.

The random walk model can start from any small network. Each iteration of the algorithm consists of the following steps:

1. A new node  $i$  is added to the network, with  $m > 1$  new links attached to it.
2. The first link is wired to an old node  $j$ , chosen at random.
3. Each other link is attached to a randomly selected neighbor of  $j$ , with probability  $p$ , or to another randomly selected node, with probability  $1 - p$ .

The parameter  $p$  is the probability of triadic closure, because by setting a link between  $i$  and a neighbor of  $j$ , say  $l$ , we close the triangle  $(i, j, l)$ . If  $p = 0$ , there is no triadic closure and new nodes choose their neighbors entirely at random. When  $p = 1$  all links except the first one are wired to neighbors of the initially selected old node, thus closing triangles.

### Box 5.6 Triadic Closure and the Strength of Weak Ties

In 1973, sociologist Mark S. Granovetter, after a long struggle with editors, published a paper entitled “The strength of weak ties” that would become the most cited article in sociology. The paper set forth a tight relationship between three fundamental features of social networks: triangles, link weight, and communities.

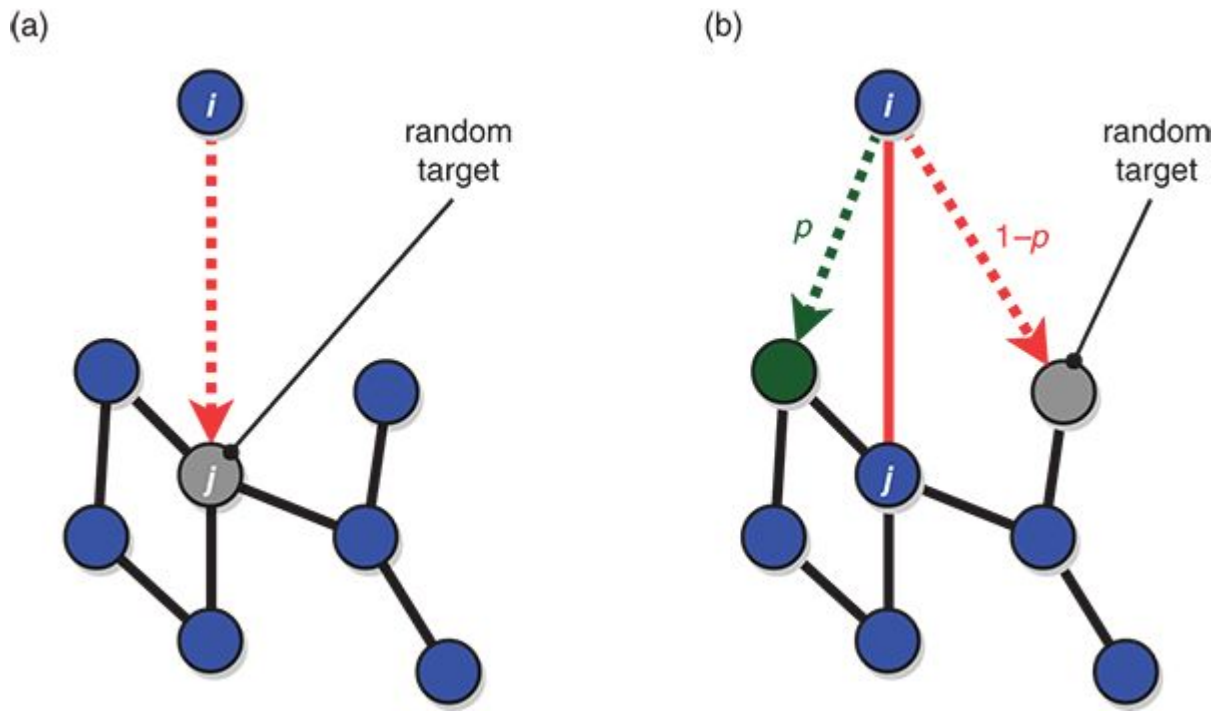
Granovetter introduced the principle of *strong triadic closure* for how links are formed in social networks. If person **a** has strong (high-weight) connections with two individuals **b** and **c**, it is very likely that **b** and **c** are friends or that they will eventually become friends. There may be various reasons. If **b** and **c** spend a lot of time with **a**, it is likely that they will eventually meet through **a**. Also, since **a** is a good friend of both, **b** and **c** will be inclined to trust each other. Finally, if **b** and **c** keep ignoring each other it might be a source of stress for the group. Strong triadic closure prescribes that there must be a link between **b** and **c**, so **a**, **b**, and **c** would form a triangle (Figure 5.9). This formalizes the relationship between triangles and link weights.

A social community is a circle of people having a lot of interactions with each other, because of family ties, work relationships, and so on. (We discuss communities in Chapter 6.) Granovetter argued that links with large weights, signaling *strong ties* between individuals, are most likely to be

found within the same community, while low-weight links (*weak ties*) tend to lie between communities. The intuition is that people in different circles have limited contact. Granovetter provided an argument to support his theory. Suppose that there is a strong link between two individuals **a** and **b** belonging to different communities. Each of them is likely to have strong connections with other members of their own community. Let us suppose that **a** is a close friend of **c**. Due to strong triadic closure, there must be a link between **b** and **c** as well, because **ab** and **ac** are strong ties. But a link connecting two communities can hardly be a side of triangles, because otherwise the communities are not well separated, so **ab** must be weak. On the contrary, since there are many strong ties between members of the same community, there will be many triangles within communities. This argument suggests an interplay between communities and link weights, as well as between communities and triangles. Despite their low weights, weak ties are critical for the structure of social networks, because they connect communities to each other, enabling the spread of information across the network.

The model is illustrated in Figure [5.10](#). It creates networks with a number of triangles that can be varied by tuning a parameter  $p$ .<sup>3</sup> The largest density of triangles is obtained when  $p = 1$ . In addition, if  $p$  is not too small, the model generates heavy-tailed degree distributions. This is due to the process of triadic closure. By

choosing a neighbor of an old node, we are just selecting a link of the network. As you may recall from Section 3.3, if we select a link at random, the probability that an endpoint of that link has a given degree is proportional to the degree. So, old nodes will receive links with a probability that is proportional to their degrees, exactly as in preferential attachment.



**Fig. 5.10** Random walk model. (a) A new node  $i$  is attached to a randomly chosen node  $j$ . (b) Each additional link brought by  $i$  is attached to a neighbor of  $j$  with probability  $p$ , which leads to the formation of triangles. Otherwise the link is attached to a randomly chosen node.

The mechanism used in the random walk model is more intuitive than the one in the preferential attachment model, as it does not assume that new nodes have knowledge of the degrees of old

nodes. The model simply explores the network in a random fashion, and the nodes are “discovered” with a frequency proportional to their degrees. In other words, the process of *triadic closure implicitly induces preferential attachment*. From this point of view it is basically equivalent to *link selection* (i.e. choosing a link at random and attaching the new node to one of the endpoints of the link). The difference is that in triadic closure the new node is attached to both endpoints of the selected link, but the resulting degree distributions are similar in both cases. Therefore, the strict proportionality between link probability and degree, necessary to generate broad degree distributions, can be enforced by simple mechanisms based on random choices.

Finally, when  $p$  is large enough that there is a sufficiently high density of triangles, the random walk model produces networks with community structure, which we will discuss in Chapter [6](#). The relationship between triangles and communities is well known in the network science literature (Box [5.6](#)).



### 5.5.4 Copy Model

In social networks, triadic closure implies that an individual *copies* the contacts of somebody else. This kind of copying mechanism can take place in other contexts as well. Some examples:

- Gene duplication is a process through which new genetic material is generated during molecular evolution. Consider the protein–protein interaction network, where each node represents the protein expressed by a gene. When a gene is duplicated, the new gene/protein node will be interacting with the same proteins as the original node in the protein interaction network. So the links to those nodes are copied.
- Scholars often discover new articles in the lists of references of publications they read, and cite them in their own papers. In doing so, they copy (some of) the citations of other publications.
- While browsing online, Web content creators may discover relevant Web pages such as authoritative sources or hub pages that provide lists of resources. By linking to these pages from newly created pages, authors copy the hyperlinks leading to them.

These scenarios are captured by the *copy model*. It is similar to the random walk model discussed earlier: a new node gets wired either to a randomly selected old node, with some probability, or else to its neighbors. However, there is no triadic closure in the copy model;

the new node does not get attached simultaneously to a node and (some of) its neighbors. Therefore we obtain networks with hubs, but with few triangles.

### 5.5.5 Rank Model

Preferential attachment implies that the nodes have a perception of how important other nodes are, as it requires knowledge of their degrees during network growth, to estimate the link probability and properly distribute the links. We may wonder whether we can operate without knowing the degrees of the nodes. In Sections [5.5.3](#) and [5.5.4](#) we have seen that triadic closure and link selection/copy are viable strategies. Here we consider a different approach.

In realistic settings, it is more common to have a perception of the *relative value* of things, rather than of their absolute value. We can state with great confidence that Bill Gates is much richer than any of the authors of this book, even though we ignore the exact amount of his wealth. The claim that we are able to rank the nodes of a network based on a specific variable (e.g. degree or age) is thus more believable than the claim that we can accurately estimate the values of the variable. This is the idea at the basis of the *rank model*.

We keep the nodes ranked by one of their properties, say the degree. Then we select nodes to receive new links with probability proportional to some inverse power of their rank. The top node will have the highest chance of receiving a link, followed by the node ranked second, third, and so on. How the link probability decays with the rank is determined by an exponent parameter.

The rank model can start from any small graph with  $m_0$  nodes. A node property, such as the degree, age, or some measure of fitness is selected to rank the nodes. Each iteration of the algorithm consists of the following steps:

1. All nodes are ranked based on the property of interest. Nodes are assigned ranks  $R = 1, 2$ , etc. Node  $l$  receives rank  $R_l$ .
2. A new node  $i$  is added to the network, with  $m \leq m_0$  new links attached to it.
3. Each new link from  $i$  is wired to an old node  $j$  with probability

$$\Pi(i \leftrightarrow j) = \frac{R_j^{-\alpha}}{\sum_l R_l^{-\alpha}}, \quad (5.13)$$

where the exponent  $\alpha > 0$  is a parameter.

Nodes may have to be re-ranked at each iteration if the ranking property depends on the links from new nodes joining the network, as happens for example when nodes are ranked by degree.

A top-ranked node (small rank) is more likely to receive a new link than a poorly ranked one (large rank). If the variable used for the ranking is degree, this means that high-degree nodes have higher chances of attracting new links than low-degree nodes, as in the BA

model. However, the actual link probability values are different because they depend on the ranks, not on the degrees.

As it turns out, the rank model generates networks with heavy-tailed degree distributions, for any property used to rank the nodes and any value of the exponent parameter. By tuning the exponent, it is possible to vary the shape of the distribution, and to reproduce the empirical distributions observed in many real-world networks.

Hubs are created even if nodes have partial information on the system, in that they are aware of the existence of only a fraction of the nodes. This reflects a familiar scenario. Imagine you are writing a Wikipedia article and wish to link to relevant news articles. You might use a search engine to identify pages to link. The search engine presents pages ranked by relevance to your query. You link to the top result with high probability; to the second result with half that probability; to the third with one-third probability; and so on. You might not even bother looking past the first page of results. Your article will be a new Wikipedia node, linking to old nodes according to a procedure closely resembling the rank model. This can help explain the emergence of popular hubs on the Web.

## 5.6 Summary

Network models help us understand the basic mechanisms that are responsible for the characteristic structural features observed in a real network. The basic ingredients of network models are the rules that determine how nodes get attached to each other. Below we list the lessons we have learned from the models reviewed in this chapter.

1. In random networks generated by the Erdős–Rényi model, every node has the same probability of becoming a neighbor of any other node. These networks have short paths, but there are very few triangles and no hubs.
2. The small-world model modifies an initial lattice structure with high average clustering coefficient by creating some random shortcuts between the nodes. A few shortcuts suffice to drastically reduce the distances between the nodes, inducing the small-world property, while the clustering coefficient remains high. The model is unable to create hubs.
3. The configuration model generates networks with any predefined degree sequence. The structure is therefore imposed “by hand,” it is not explained by the model. The configuration model is often used as a baseline, to check whether any property of a network is due to its degree distribution alone or to other factors. This can be done by comparing the property of interest in

the original system and in randomized networks with the same degree sequence, created by the model.

4. Realistic network models include network growth, in that nodes and links are added to the graph over time. This matches the evolution of many real-world networks, like the Internet, the Web, etc.
5. Preferential attachment is the key mechanism to explain the emergence of hubs: the higher the degree of a node, the higher the probability that it will be connected to other nodes.
6. The Barabási–Albert model, with its combination of network growth and preferential attachment, yields networks with heavy-tailed degree distributions, therefore explaining the emergence of hubs.
7. Preferential attachment can be induced implicitly by simple processes involving random choices, like triadic closure and link selection.
8. Several models have been proposed to overcome the limitations of the Barabási–Albert model, by introducing ingredients such as attractiveness, fitness, triadic closure, and ranking.

## 5.7 Further Reading

The random graph model was introduced in the same year by [Erdős and Rényi \(1959\)](#) and [Gilbert \(1959\)](#), though the idea was put forward in an earlier paper by [Solomonoff and Rapoport \(1951\)](#). The Gilbert model is often erroneously attributed to Erdős and Rényi in the literature. We refer to the average number of regular contacts that humans can maintain as *Dunbar's number* ([Dunbar, 1992](#)).

The small-world model was developed by [Watts and Strogatz \(1998\)](#). [Molloy and Reed \(1995\)](#) proposed the configuration model. Exponential random graphs were introduced by [Holland and Leinhardt \(1981\)](#).

[Barabási and Albert](#) are usually credited with the preferential attachment model, often called the Barabási–Albert or BA model ([Barabási and Albert, 1999](#)). Other scholars had proposed the model before; the closest ancestor was a paper by [Price \(1976\)](#). Non-linear preferential attachment was investigated by [Krapivsky et al. \(2000\)](#) and [Krapivsky and Redner \(2001\)](#). Attractiveness was added to preferential attachment by [Dorogovtsev et al. \(2000\)](#). The fitness model was proposed by [Bianconi and Barabási \(2001\)](#).

[Granovetter \(1973\)](#) authored the pioneering paper “The strength of weak ties.” The random walk model was introduced by [Vázquez \(2003\)](#). The copy model was an idea of [Kleinberg et al. \(1999\)](#) that emerged during early studies of the Web graph. Several authors proposed gene duplication models ([Wagner, 1994](#); [Bhan et al., 2002](#); [Solé et al., 2002](#); [Vázquez](#)



[\*et al.\*, 2003a](#)). The rank model was developed by [\*Fortunato et al.\* \(2006\)](#).

## Exercises

- 5.1 Go through the Chapter 5 Tutorial on the book's GitHub repository.<sup>4</sup>
- 5.2 What is the difference between the random graph by Erdős and Rényi and that by Gilbert?
- 5.3 Suppose we want to construct a random graph with 1000 nodes and about 3000 links. Give a value of the link probability  $p$  that could lead to this outcome.
- 5.4 Suppose you are making a random network with 50 nodes, and you want the average node degree  $\langle k \rangle$  to be 10. What approximate value for  $p$  would you use in this case?
- 5.5 Given a random network of 50 nodes and average node degree  $\langle k \rangle = 10$ , which of the following is likely to be closest to the average path length of that network?
- a. 5
  - b. 0
  - c. 25
  - d. 5
- 5.6 Consider the family of random networks with average degree  $\langle k \rangle = 10$ . How many nodes are we likely to need in order to generate such a network with average path length  $\langle \ell \rangle = 3.0$ ? (*Hint: If you use a guess and check strategy, make sure that  $\langle k \rangle$*

stays the same for the various values of  $N$ . Each will involve a different value of  $p$ .)

- a. 60
- b. 100
- c. 250
- d. 500

5.7 Build a random network with 1000 nodes and  $p = 0.002$ . Plot its degree distribution. (*Hint*: We show how to plot a distribution in the Chapter 3 Tutorial.) Answer the following questions:

- 1. What is the largest degree of the network?
- 2. What is the *mode* of the distribution (i.e. the most common value of the degree)?
- 3. Is the network connected? If not, how many nodes are in the giant component?
- 4. What is the average clustering coefficient? Compare it with the link probability  $p$
- 5. What is the network diameter?

5.8 Consider the following process. First, start with a collection of  $N$  nodes and no links. This is clearly a disconnected network. Then, one by one, add a link between two nodes not already connected to each other. Continue until you have a complete network. How many steps are in this process?

5.9 Consider the step-by-step process from the previous question. At each step in the process, suppose you record the size of the largest component. Which of the following tends to be true

about the sequence of largest component sizes as you add edges?

- a. Increases slowly at the beginning of the sequence, increasing very fast at the end of the sequence
- b. Increases slowly until some threshold, increases very quickly for a short time, and then increases slowly afterwards
- c. Increases at a constant rate from the beginning to the end of the sequence
- d. Increases very fast at the beginning of the sequence, then tapers off and increases slowly until the end
- e. Increases and decreases randomly

**5.10** Reproduce the plot of Figure [5.2](#) for networks with 1000 nodes. (*Hint:* Use the NetworkX function to generate random networks.) Use 25 equally spaced values of the link probability, in the interval  $[0, 0.005]$ . For each value generate 20 different networks, compute the relative size of the giant component and report the average and the standard deviation in the plot.

**5.11** What is a reason why random networks are not good models of social networks?

- a. Random networks are typically not connected
- b. Random networks have small average shortest-path lengths
- c. Nodes in random networks have very different degrees
- d. Random networks have low clustering coefficients

- 5.12** Consider a ring-like lattice like the one in Figure 5.5(a), with 100 nodes each connected to its four nearest neighbors (two on either side of it). What is the average clustering coefficient? Does the network size matter? (*Hint*: Given the symmetry, it is sufficient to calculate the clustering coefficient of any node.)
- 5.13** The Watts–Strogatz model is useful in capturing which property of real-world social networks not found in Erdős–Rényi random graphs?
- Short average path lengths
  - Long average path lengths
  - Low clustering coefficient
  - High clustering coefficient
- 5.14** Reproduce the plot of Figure 5.5(b), by calculating the average shortest path ( $\langle \ell \rangle$ ) and the average clustering coefficient ( $C$ ) for Watts–Strogatz networks constructed for different values of the rewiring probability  $p$ . Take 20 equally spaced values of  $p$  between 0 and 1. For each value of  $p$ , build 20 different networks and compute the average of  $\langle \ell \rangle$  and  $C$ . To plot the two curves on a common  $y$ -axis, you can normalize the values by dividing them by the corresponding values for  $p = 0$ .
- 5.15** Build Watts–Strogatz networks with 1000 nodes,  $k = 4$ , and these values for the rewiring probability:  $p = 0.0001, 0.001, 0.01, 0.1, 1$ . Compute and compare their degree distributions, by plotting them in the same diagram.

- 5.16** Consider the US airport network (USAN). Create a randomized version of it (RUSAN) using the configuration model. To do that, take the degree sequence of the network and apply the `configuration_model()` function of NetworkX. Carry out the following tasks:
1. Verify that the degree distribution is identical to that of the USAN
  2. Compare the average shortest paths of USAN and RUSAN. How do you interpret the difference in the values?
  3. Compare the average clustering coefficients of USAN and RUSAN. How do you interpret the difference in the values?
- 5.17** Which of the following features would you expect to find in a BA but not in a random network with the same number of nodes and edges?
- a. Nodes with degree greater than one
  - b. Hub nodes with degree many times larger than that of a typical node
  - c. Short average path lengths
  - d. Long average path lengths
- 5.18** Build a BA network with 1000 nodes and  $m = 3$ . Carry out the following tasks:
1. Plot the degree distribution of the network, in double-logarithmic scale
  2. Derive the average degree, see how it compares with  $m$ , and interpret the result
  3. Calculate the average clustering coefficient

4. Verify that the graph is connected
  5. Calculate the average shortest path
- 5.19** Build an Erdős– Rényi random graph with the same number of nodes and links as the BA network in the previous exercise.
1. Derive the degree distribution and compare it with that of the BA network, by drawing them in the same plot, in double-logarithmic scale
  2. Calculate the average clustering coefficient and average shortest path length and compare them with the corresponding values for the BA network. Interpret the results
- 5.20** The attractiveness and fitness models are based on the same idea that nodes have an intrinsic appeal, unrelated to their degree. What are the differences between the two models?
- 5.21** In the fitness model, suppose that the fitness of a node coincides with its degree. Could you guess what kind of degree distribution the resulting network will have? (*Hint: The discussion on non-linear preferential attachment in Section [5.5](#) might help.*)
- 5.22** Give a reason why networks generated by the BA model do not have many triangles.
- 5.23** If you use an online social network such as Facebook, Instagram, or LinkedIn, consider your links in this network: how many are strong ties and how many are weak ties?

- 5.24** Link selection consists of choosing a link and wiring a new node to one of its endpoints. Suppose that we wire the new node to both endpoints. What would be the difference from the random walk model? And how would networks generated by the two models differ?
- 5.25** Suppose that we wish to build a network such that there is a relevant amount of squares (cycles of length four). Based on what you learned about triadic closure, could you suggest a mechanism that incentivizes the formation of squares?
- 5.26** Consider two versions of the rank model, with different ranking criteria. In the first version nodes are ranked by age (the time elapsed since they were added to the network). In the second version nodes are ranked by their degree. Is there a difference between networks generated by the two models, and if so, what is it?
- 5.27** The `socfb-Northwestern25` network in the book's GitHub repository is a snapshot of Northwestern University's Facebook network. The nodes are anonymous users and the links are friend relationships. Load this network into a NetworkX graph; be sure to use the proper graph class for an undirected, unweighted network. Once you measure the number of nodes and links, use `nx.gnm_random_graph()` to create a separate random network with the same number of nodes and links as the Facebook graph. Use this random network to answer the following questions:



1. What is the 95th percentile for degree in the random network (i.e. the value such that 95% of nodes have this degree or less)?
2. We are dealing with a random network, so some properties are going to differ somewhat each time one is generated. True or false: Given fixed parameters  $N$  and  $L$ , all random networks created with `gnm_random_graph()` will have the same mean degree.
3. Which of the following shapes best describes the degree distribution in this random network?
  - a. Uniform: node degrees are evenly distributed between the minimum and maximum
  - b. Normal: most node degrees are near the mean, dropping off rapidly in both directions
  - c. Right-tailed: most node degrees are relatively small compared to the range of degrees
  - d. Left-tailed: most node degrees are relatively large compared to the range of degrees
4. Estimate the average shortest-path length in this random network using a random sample of 1000 pairs of nodes.
5. What is the average clustering coefficient of this random network? Answer to at least two decimal places.

---

<sup>1</sup> The link probability is not to be confused with the rewiring probability that will be introduced in Section [5.2](#), although we use the letter  $p$  for both.

<sup>2</sup> Note that the rewiring probability in the small-world model is not the same as the link probability in the random network model, despite the fact that we use the letter  $p$  for both. While this might be a bit confusing, it is a convention of the network science

community that we follow; please be careful to interpret the parameter  $p$  based on the context of the model that is being discussed.

<sup>3</sup> Again, do not confuse this parameter with those in the random network and small-world models that use the same letter  $p$ .

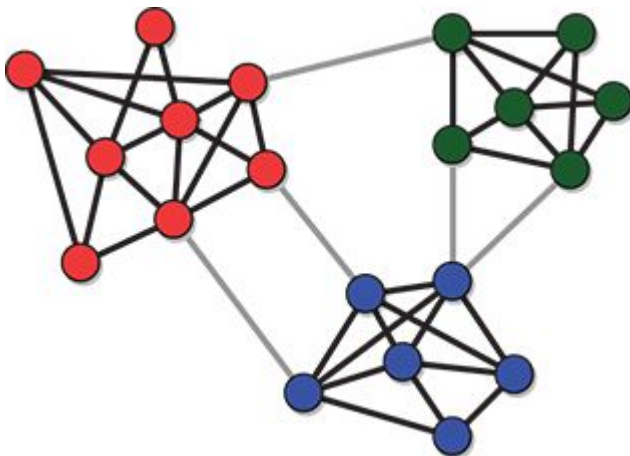
<sup>4</sup> [github.com/CambridgeUniversityPress/FirstCourseNetworkScience](https://github.com/CambridgeUniversityPress/FirstCourseNetworkScience)

## 6 Communities



**clus·ter:** (*n.*) a group of similar things or people positioned or occurring closely together.

When you look at the layout of a network, one of the first things you may notice is that **nodes are grouped in communities**, also called **clusters** or **modules** — sets of nodes with a relatively higher density of connections within than between them (Figure [6.1](#)).

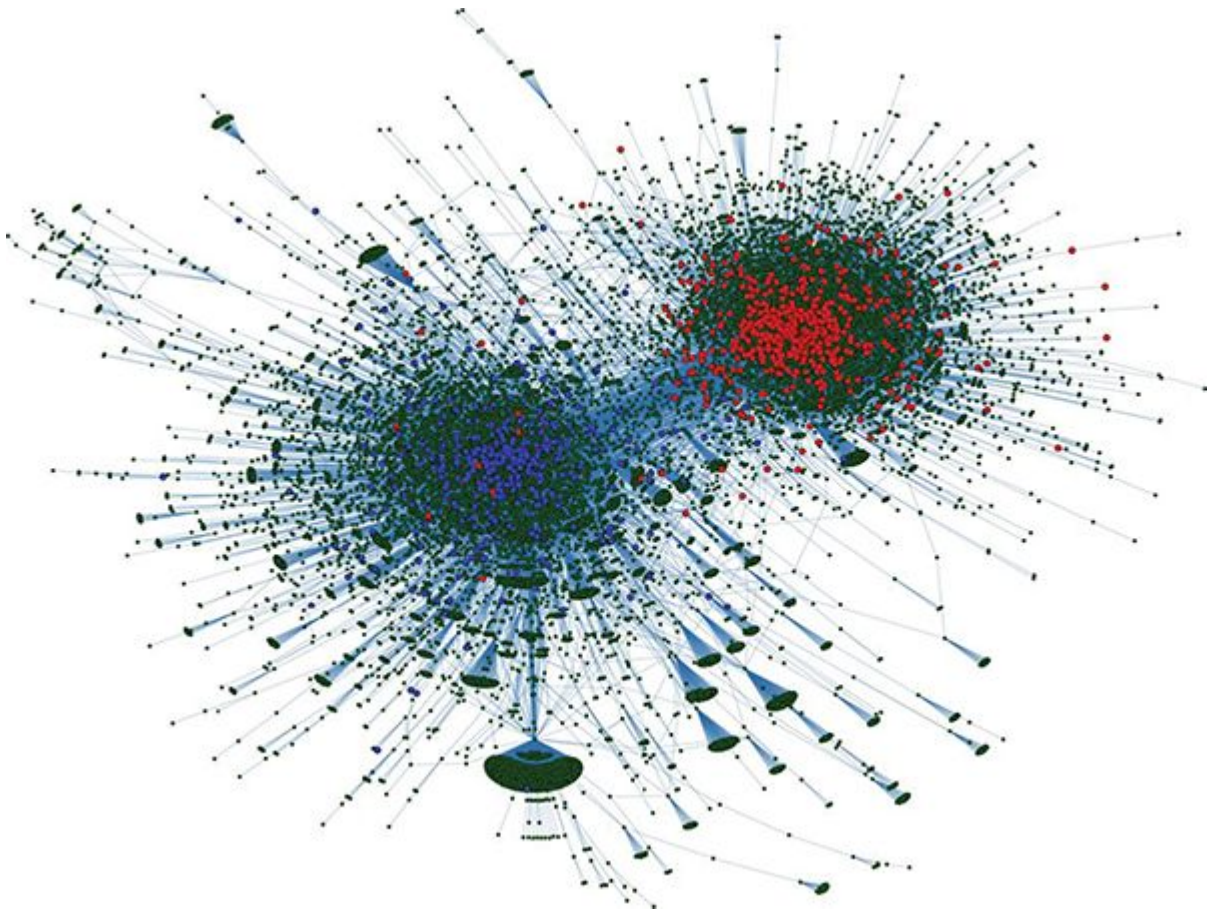


**Fig. 6.1** A network with three communities, indicated by node colors.

Communities often tell us how a network is organized and what functions it serves. For example, dense clusters of connected neurons in the brain are often synchronized in their firing patterns. In a protein–protein interaction network, groups of connected (interacting) proteins are typically associated with a particular biological function within the organism. Sometimes we can infer the

role of an unknown gene with respect to a complex disease by looking at the cluster it belongs to in a gene regulatory network. On the Web, as we discussed in Section [4.2.5](#), clusters of pages with many hyperlinks pointing to each other usually identify a topic. And in social networks, as discussed in Section [2.1](#), communities of friends share important features, such as political beliefs. Social communities can have significant influence on public opinion. For example, when looking at the diffusion network for politically charged memes on Twitter (Figure [0.3](#)), we immediately note that people are divided across two distinct communities that do not interact much with each other.

A similar picture is offered in Figure [6.2](#), showing the retweet network of political memes in the United States. Again we observe two mostly segregated communities. When we inspect some of the users, it becomes clear that the two clusters are aligned with political right and left. As discussed in Section [4.5](#), this scenario is sometimes called an *echo chamber* or *filter bubble* to indicate that a person is only exposed to people with similar ideas and beliefs. While it is normal for us to have friends like us in real life, with online social networks and social media it is easier to filter out different views, as we are encouraged to connect with people who are similar to us or with whom we already share common friends. We also have tools to easily mute or ignore people with whom we disagree, which is a little harder to do in real life. It has been theorized that when opinions are unchallenged, biases are reinforced and polarization may ensue.



**Fig. 6.2** Retweet network of political hashtags on Twitter prior to the 2010 US election. Colored nodes are a sample of users classified as liberal (blue) or conservative (red). A link between two nodes indicates a retweet of one of the two corresponding users by the other.

Knowing the community structure of a network allows us to classify the nodes based on the position they have in their own clusters. Nodes that are fully embedded in a cluster, in that their neighbors belong to the same cluster, represent the core of the group, as they do not mix with members of other groups. Nodes lying at the boundary of a community have neighbors both inside and outside their group and serve as gatekeepers between different

parts of the network. As such, they play an important role for diffusion processes taking place in the network. If we want to stop a rumor or fake news spreading in a social network, halt the spreading of an epidemic in a contact network, or make sure critical information reaches all communities, those are the nodes we need to control.

Given the importance of communities in understanding the functions of a network and of individual nodes, it is critical to be able to detect communities in a network. Sometimes communities are very evident and all we need to do to see them is lay out the nodes on a plane so that connected nodes are positioned close to each other. That is the idea behind the popular *force-directed network layout algorithms* described in Chapter 1. For example, in Figure 6.2, because people in a community (liberal or conservative) are densely connected to each other, they end up clustered together in the layout. Many networks of interest, however, are much larger than those for which it is possible to produce meaningful visualizations. And even in many small systems, visualization does not help to identify the clusters. So it is necessary to develop algorithms that can automatically discover the communities starting from the knowledge of the network structure and possibly other inputs (e.g. the desired number of clusters).

The problem of identifying communities in networks is a truly interdisciplinary topic, and as such it takes different names: *community detection*, *community discovery*, and *clustering*,<sup>1</sup> among others. Grouping nodes into communities is considered an *unsupervised* classification task, in that we do not have precise prior

knowledge or examples of what the resulting partitions should look like. Indeed, there is no unique definition of community. The natural intuition is that there should be more links among nodes in the same community than among nodes in different communities. In other words, the link density within (across) communities should be higher (lower) than the overall network density [Eq. (1.3)]. This criterion can be mathematically formalized in many different ways. This is why we find many clustering techniques in the scientific literature.

This chapter offers a brief introduction to the problem and its most popular solutions. We start by introducing some basic elements: the main variables, classic definitions of communities, and high-level properties of partitions. Then we discuss two related problems, network partitioning and data clustering, which have contributed many tools and techniques to the topic. Finally we present some of the widely adopted algorithms, along with standard procedures to test clustering techniques.

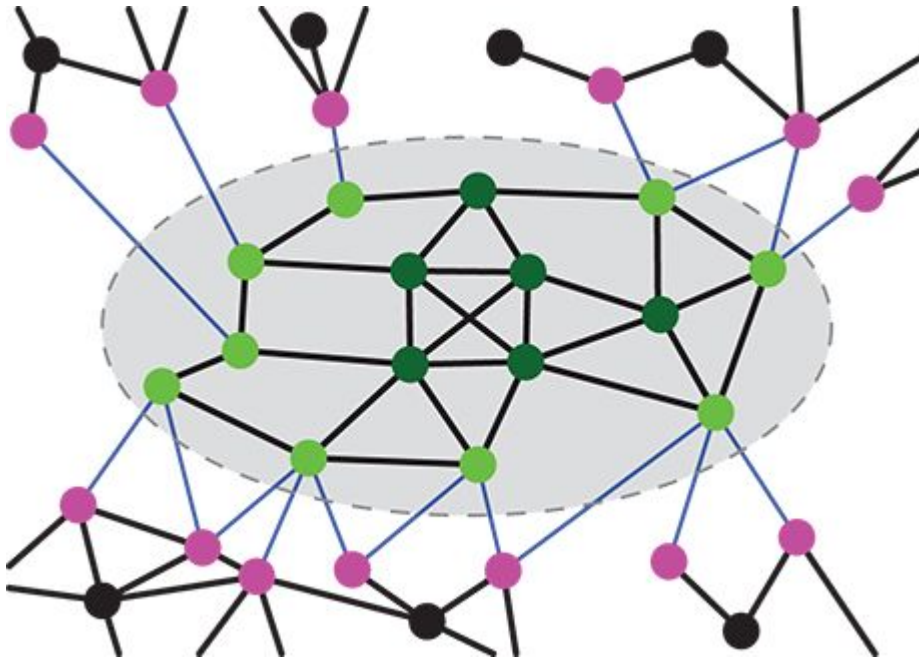
## **6.1 Basic Definitions**



### 6.1.1 Community Variables

A community is typically a connected subnetwork. Consider the community illustrated by the green nodes in Figure 6.3. The magenta nodes are external but connected to the community, while some of the remaining nodes of the network are shown in black. The blue links connect the community to the rest of the network. The key community variables we will be using throughout this chapter are:

- The *internal* and *external degree* of a node in a community — the number of neighbors inside and outside the community, respectively. In Figure 6.3, the internal degree of a green node is the number of black links attached to it and the external degree is the number of blue links.
- The number of *internal links* to the community — the number of links connecting two nodes in the community (black links in the oval of Figure 6.3).
- The *community degree* — the sum of the degrees of the nodes in the community; the sum of the number of neighbors of each green node in Figure 6.3.
- The *internal link density* — the ratio between the number of internal links and the maximum number of links that could exist between any two community nodes. This is the same as the density defined in Chapter 1, but for the community subnetwork.



**Fig. 6.3** Schematic picture of a community (inside the gray oval) and of its immediate neighbors. Reprinted from [Fortunato and Hric \(2016\)](#) with permission from Elsevier.

Let us formulate the definitions and notation for our community variables a bit more formally. Suppose you have a community  $C$ .

- The numbers of nodes and internal links in  $C$  are  $N_C$  and  $L_C$ , respectively.
- The *internal degree*  $k_i^{int}$  and the *external degree*  $k_i^{ext}$  of a node  $i$  with respect to community  $C$  are the numbers of links connecting  $i$  to nodes in  $C$  and to the rest of the network, respectively. Since every neighbor of  $i$  must be either inside or outside  $C$ , the degree of  $i$  is  $k_i = k_i^{int} + k_i^{ext}$ . If  $k_i^{ext} = 0$  and  $k_i^{int} > 0$ , then  $i$  has neighbors only within  $C$  and is an *internal node* of  $C$  (dark green nodes in the figure). If  $k_i^{ext} > 0$  and  $k_i^{int} > 0$  for a node  $i \in C$ , then  $i$  has neighbors both inside and outside  $C$  and is a *boundary node* of  $C$  (bright green nodes in the figure). If  $k_i^{int} = 0$ , then the node is disjoint from  $C$ , as it has no neighbors inside the community (black nodes in the figure).
- The *internal link density* is given by

$$\delta_C^{int} = \frac{L_C}{\binom{N_C}{2}} = \frac{2L_C}{N_C(N_C - 1)}. \quad (6.1)$$

Note that this is equivalent to Eq. (1.3) for nodes and links internal to  $C$ , because we are assuming that the network is undirected; the maximum number of internal links that a community with  $N_C$  nodes may have is  $\binom{N_C}{2}$ .

- The *community degree*, or *volume*, is the sum of the degrees of the nodes in  $C$ :

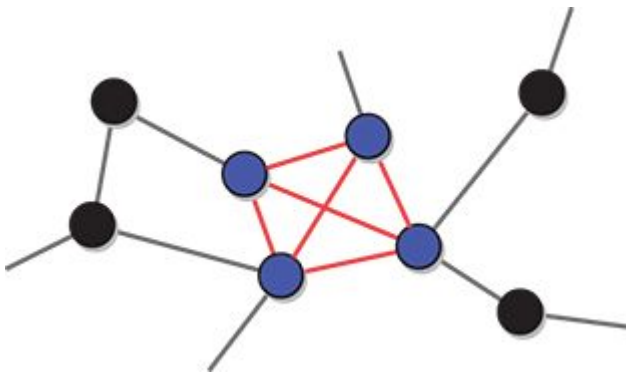
$$k_C = \sum_{i \in C} k_i. \quad (6.2)$$

All definitions hold for undirected and unweighted networks. The extensions to weighted networks are straightforward, we just need to replace degrees with strengths. For instance, the internal degree of a node becomes the *internal strength*, which is the sum of the weights of the links connecting it to nodes in the community. For directed networks we need to distinguish between incoming and outgoing links. Extensions of the measures are fairly simple to implement, but their usefulness is unclear.

### 6.1.2 Community Definitions

Figure [6.1](#) is the traditional picture of network community structure. It emphasizes two things: (1) communities have high *cohesion* (i.e. they have many internal links, so the nodes stick together) and (2) communities have high *separation* (i.e. they are connected to each other by few links). Classic definitions of community-like structures are based on cohesion and on the interplay of cohesion and separation.

Definitions based on cohesion alone treat the community as a system of its own, disregarding the rest of the network. The most popular notion of community of this type is that of a *clique*, defined in Chapter [1](#) as a complete subnetwork, whose nodes are all connected to each other (Figure [6.4](#)). However, in general, communities are not as dense as cliques. Moreover, all nodes have identical internal degree within a clique, whereas in real network communities some nodes are more important than others, as reflected in their heterogeneous linking patterns.



**Fig. 6.4** Part of a network including a 4-node clique, identified by the blue nodes and the red links.

For a more useful definition of community, we should take into account both the internal cohesion of a candidate subnetwork and its separation from the rest of the network. A popular idea is that a community is a subnetwork such that *the number of internal links is larger than the number of external links*. This idea has inspired the following definitions:

- A *strong community* is a subnetwork such that each node has more neighbors in the subnetwork than in the rest of the network. In other words, the internal degree of each node in a strong community exceeds its external degree.
- A *weak community* is a subnetwork such that the sum of the internal degrees of all nodes exceeds the sum of their external degrees.

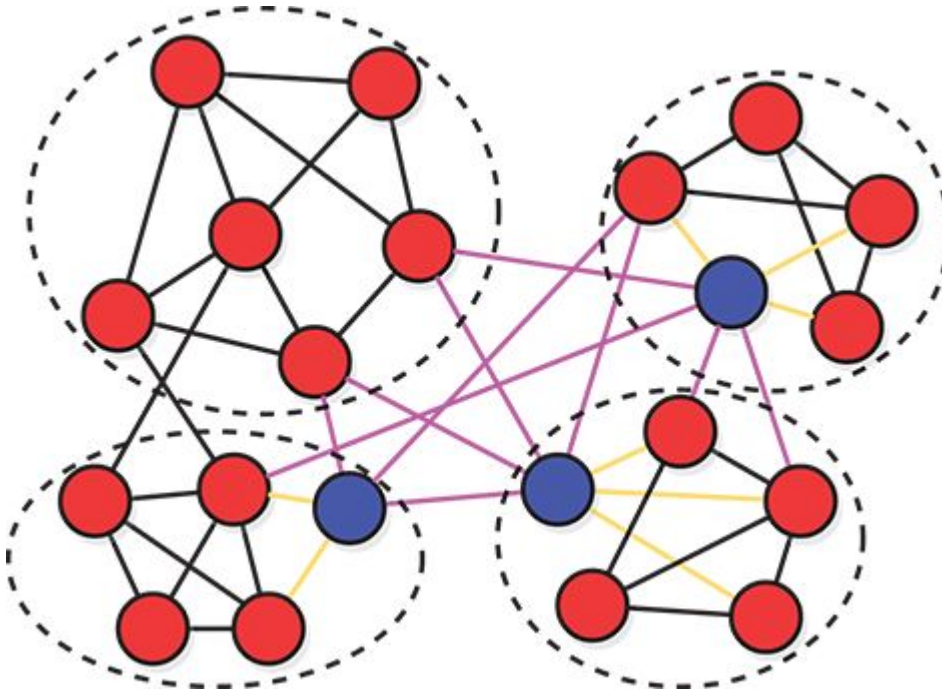
A strong community is necessarily also a weak community: if the inequality between internal and external degree holds for each node, then it must hold for the sum over all nodes. The converse is not generally true: if the inequality between internal and external degree holds for the sum, it may be violated for one or more nodes.

A drawback of these definitions is that they separate the community under consideration from the rest of the network, which is taken as a single object. But the rest of the network can in turn be partitioned into communities. If a subnetwork  $C$  is a proper community, one would expect each of its nodes to be more strongly

attached to the other nodes in  $C$  than to the nodes in any other subnetwork of the partition. Such a concept has inspired less stringent definitions of strong and weak community:

- A *strong community* is a community such that each node has more neighbors inside it than in any other community.
- A *weak community* is a community such that the sum of the internal degrees of the nodes inside it exceeds the sum of their external degrees in each of the other communities. A node's external degree in a community other than its own is the number of neighbors in that community.

A strong (weak) community according to the earlier definition is necessarily also a strong (weak) community according to the less stringent definition. The converse is not true in general, as illustrated by the example in Figure [6.5](#). In particular, a subnetwork can be a strong community in the less stringent sense even though all of its nodes have internal degree smaller than their respective external degree.



**Fig. 6.5** Strong and weak communities. The four subnetworks enclosed in the dashed contours are weak communities according to both definitions we have given. They are also strong communities according to the less stringent definition, as the internal degree of each node exceeds the number of links joining the node with those of every other community. However, three of the subnetworks are not strong communities in the more stringent sense, because some nodes (in blue) have external degree larger than their internal degree (the internal and external links of these nodes are colored in yellow and magenta, respectively). Adapted from [Fortunato and Hric \(2016\)](#).

As we have seen, traditional definitions of communities rely on counting links (internal and external) in various ways. But the number of links usually increases with the community size. Therefore, the comparisons between internal and external degrees of different communities are biased by their sizes. Ideally, we would



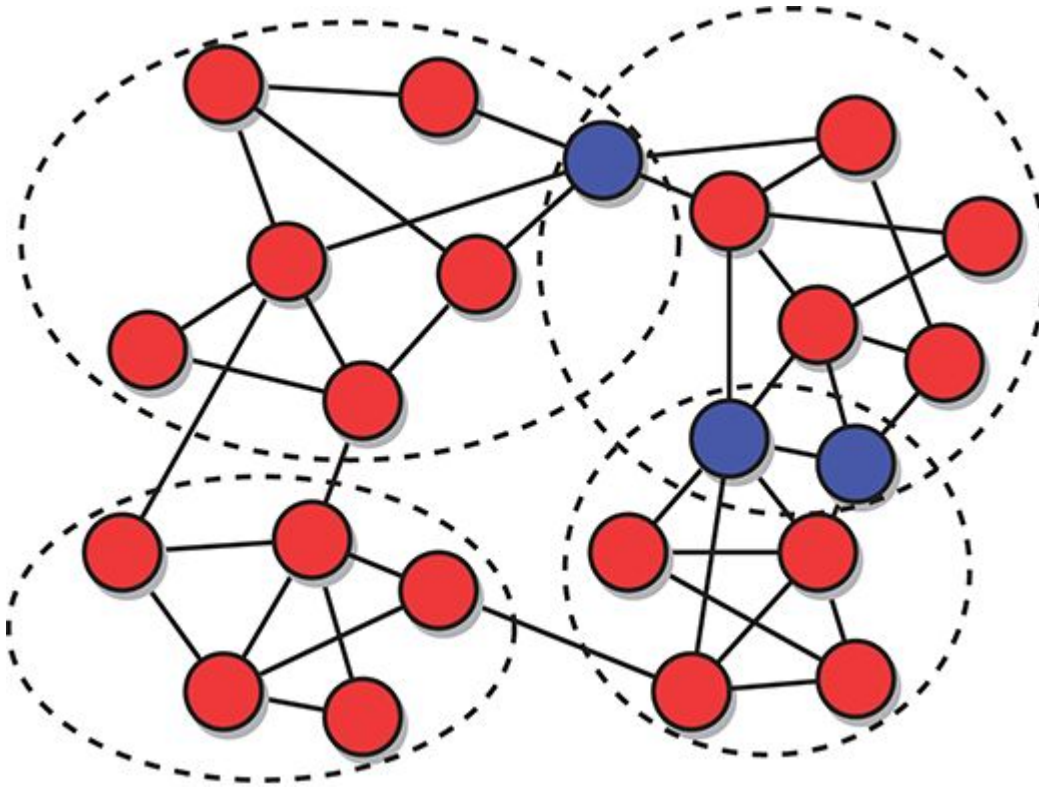
like to compare *probabilities*: if nodes within a subnetwork are more likely to be connected than nodes in different subnetworks, we would call the subnetwork a community. Probabilities eliminate troublesome dependencies on community size. But how to determine link probabilities? For that, we need a model stating how links are formed in networks with community structure. Sections [6.3.4](#) and [6.4.1](#) present probabilistic models to define and detect communities.

Is a definition of community really necessary? Actually, most network clustering methods do not require a precise definition of community, as we will see in Section [6.3](#). However, defining criteria for communities beforehand can be useful when checking the reliability of the final results.

### 6.1.3 Partitions

A *partition* is a division, or grouping of a network into communities, such that each node belongs to only one community. The number of all possible partitions is called the *Bell number* and increases faster than exponentially with the number of nodes of the network. For instance, a network with 15 nodes has 1,382,958,545 possible partitions! Therefore, for networks with more than a handful of nodes, it is hopeless to choose the best partition of a network by going through all of them. Indeed, clustering algorithms typically explore only a tiny portion of the space of all partitions, where interesting solutions are most likely to be found.

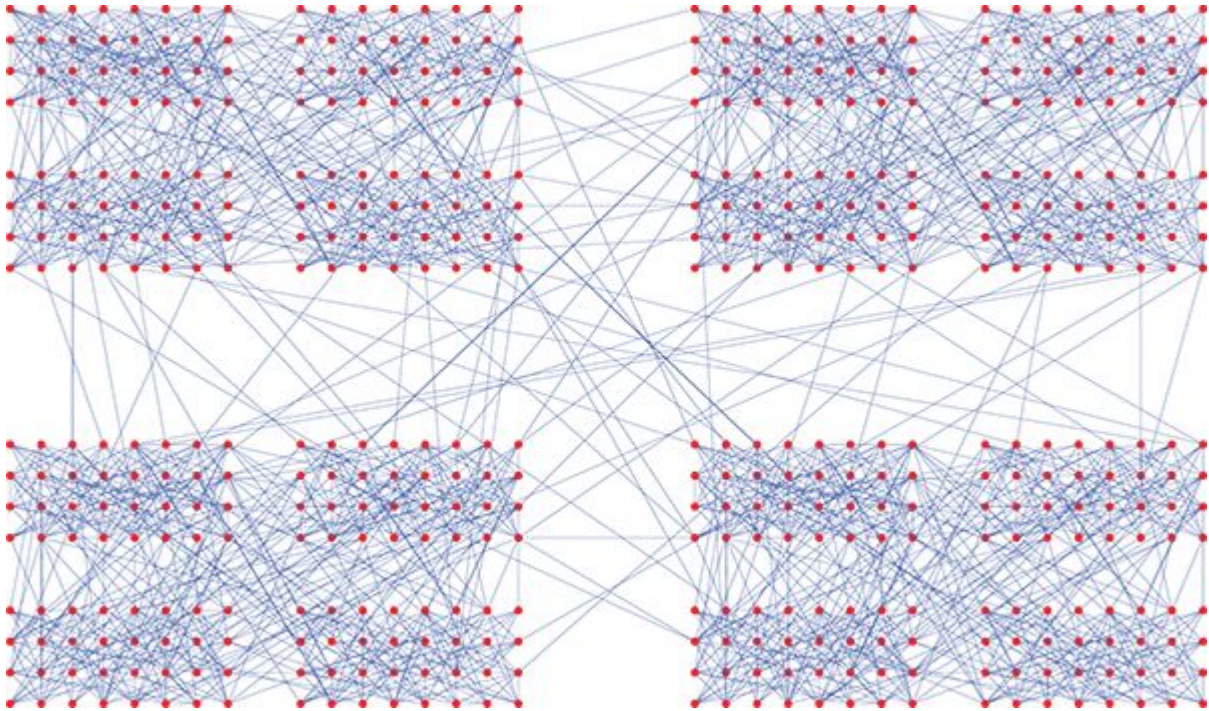
The communities in many real networks *overlap* (i.e. they share some of their nodes). For instance, in social networks individuals can belong to different circles at the same time, like family, friends, and work colleagues. Figure [6.6](#) shows an example of a network with overlapping communities. A division of a network into overlapping communities is called a *cover*. The number of possible covers of a network is far higher than the already huge number of partitions, due to the many ways clusters can overlap.



**Fig. 6.6** Overlapping communities. We show a division of a network into four communities, enclosed by the dashed contours. Three of them share nodes, indicated in blue. Adapted from [Fortunato and Hric \(2016\)](#).

Partitions can be *hierarchical* when the network has multiple levels of organization, at different scales. In this case, clusters display in turn community structure, with smaller communities inside, which may again contain smaller communities, and so on (Figure [6.7](#)). For instance, in a collaboration network of employees of a multinational corporation, we would expect to distinguish clusters of employees working in the same branch, but within each branch we might see a further subdivision into departments. In such

situations every level of the hierarchy has a meaning, and a good clustering method should be able to uncover all of them.



**Fig. 6.7** A network with hierarchical communities. We can observe two hierarchical community structures: four large clusters with 128 nodes each and 16 small clusters with 32 nodes each. The smaller clusters are fully included within the larger ones.

Partitions of real networks are often *heterogeneous*, in that some of the community properties may vary widely from one cluster to another. For instance, there is often a big difference in community size. In the Web, communities correspond roughly to pages or websites dealing with the same or similar topics. Since some topics are more general or popular than others, there are clusters with millions of Web pages along with clusters with just a few hundred or thousand pages. The cohesiveness is also very variable. If we

measure it by the internal community link density, introduced in Section [6.1.1](#), we find that in several real networks this quantity spans orders of magnitude, implying that some clusters are much more cohesive than others. This may reflect a variable capacity of groups of nodes to “attract” and link to each other. But it might also be due to the dynamic character of the community formation process: some communities are fully developed because their nodes have been around for a sufficiently long time, while others may still be developing if many of their members have been introduced more recently.

## 6.2 Related Problems

### 6.2.1 Network Partitioning

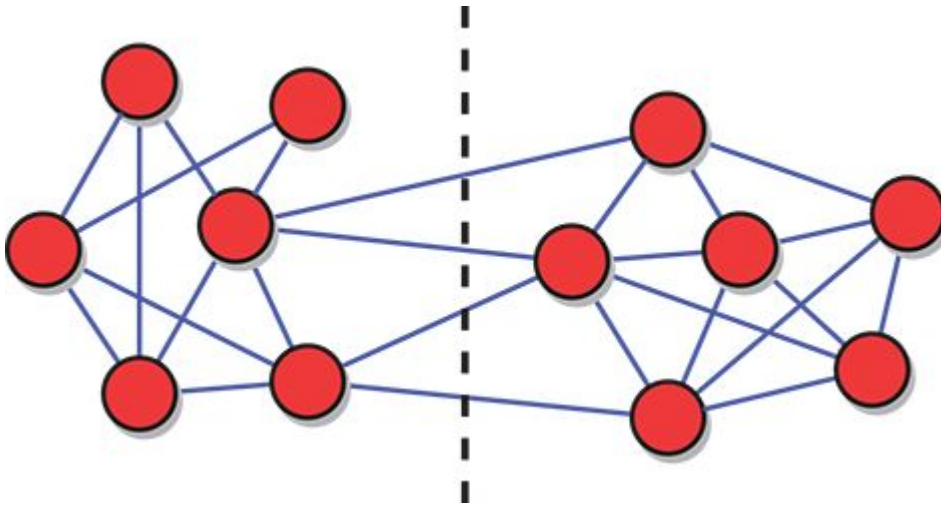
We have seen that communities are usually well separated from each other. Identifying well-separated subnetworks is the goal of *network partitioning*. Here the focus is on the separation, regardless of how many links lie inside the subnetworks. Therefore network partitioning algorithms are not suitable to detect communities, in general. Nevertheless, some partitioning techniques are used to detect communities as well, typically in combination with other procedures. So it is useful to get acquainted with this problem.

Network partitioning is motivated by important practical problems. A classic example is parallel computing, where one aims at distributing tasks to processors, such that the number of communication links between groups of processors handling different tasks is low, to speed up the calculation. Network partitioning has been applied to problems in a large variety of domains: solution of partial differential equations and of sparse linear systems of equations, image processing, fluid dynamics, road networks, mobile communication networks, air traffic control, and more.

The partitioning problem consists of finding a division of a network into a given number of subnetworks, or clusters, of given sizes, such that the total number of links connecting nodes in different subnetworks is minimized. Figure [6.8](#) shows an example in which two clusters of equal size are desired; in this case the problem is also called *graph bisection*. The set of links joining the subnetworks to each other is called a *cut*, because their removal separates the



clusters from each other, and their number is called the *cut size*. This is why the task is also known in the literature as the *minimum cut problem*.



**Fig. 6.8** Graph bisection. The network in the figure has 12 nodes. The goal is to divide it into two parts, having the same number of nodes, such that the number of links joining the parts is minimal. The solution is indicated by the vertical dashed line, which separates the two parts with a minimum cut size of four links.

Why is it necessary to specify the number of clusters of the partition beforehand? After all, we could let the partitioning procedure find the optimal number. This is not an option though, because it gives a trivial solution: since we seek to minimize the cut size, the best possible partition has a single cluster, including the entire network, which yields a cut size of zero. The next question is why we must specify the sizes of the clusters. The reason again is to avoid trivial, non-informative solutions. For instance, if the network has one leaf (a node with degree one), the bipartition consisting of



the leaf on one side and the rest of the network on the other has a cut size of one, as there is a single link separating the clusters. Such a solution cannot be beaten, but is not helpful. The main focus in network partitioning is the search for balanced solutions (i.e. partitions whose clusters have approximately the same size — as in Figure [6.8](#)). In the case of bisection, if the network has an odd number of nodes, one cluster will have one node more than the other.

One of the first and most popular methods to solve the graph bisection problem was the *Kernighan–Lin algorithm*. It is based on a very simple idea: given an initial bisection of the network, we swap pairs of nodes between the clusters, such as to obtain the greatest decrease of the cut size, while the size of the clusters does not change.

We start from an arbitrary partition  $P$  of the network into two clusters  $A$  and  $B$ . For instance, we can select half of the nodes at random and put them in one cluster, and the rest in the other cluster. Each iteration of the algorithm consists of the following steps:

1. For each pair of nodes  $i, j$ , with  $i \in A$  and  $j \in B$ , compute the variation in cut size between the current partition and the one obtained by swapping  $i$  and  $j$ .
2. The pair of nodes  $i^*$  and  $j^*$  yielding the largest decrease in the cut size is selected and swapped. This pair of nodes is locked; they will not be touched again during this iteration.
3. Repeat steps 1 and 2 until no more swaps of unlocked nodes yield a decrease in the cut size. This yields a new partition  $P'$ , that is used as a starting configuration for the next iteration.

The procedure ends when the cut size of partitions obtained after consecutive iterations is the same, meaning that the algorithm is unable to improve the result. The Kernighan–Lin algorithm can easily be extended to partitions with more than two clusters, by swapping nodes between pairs of clusters.

The solutions delivered by the Kernighan–Lin algorithm depend on the choice of the initial partitions. The poorer the quality

of the initial partition — the higher its cut size — the worse the final solution and the longer the time to reach convergence. To obtain better outcomes, we can consider multiple random partitions and choose the one with the lowest cut size as the initial partition. Another limit is that Kernighan–Lin is a *greedy algorithm*, in that it tries to minimize the cut size at each step. A drawback of greedy strategies is that they are likely to get stuck in *local optima*, solutions with sub-optimal cut size such that any local swap leads to worse solutions. A more advanced version of the algorithm mitigates this limitation by occasionally swapping a pair of nodes that yields an increase in cut size. Accepting such moves may help escape sub-optimal solutions and more closely approach the absolute minimum of the cut size.

The Kernighan–Lin algorithm is widely applied as a post-processing technique, to improve partitions delivered by other methods. Such partitions can be used as starting points for the algorithm, which might return solutions with lower cut size.

Clusters identified via network partitioning are well separated but not necessarily cohesive, so they may not be good communities according to the widely accepted high-level definition we gave earlier. In addition, network partitioning requires the specification of the number of clusters to be found. While this is also a feature of a number of community detection methods, it would be preferable to be able to deduce this number directly from the data.

NetworkX has a function for bisectioning a network with the Kernighan–Lin algorithm:

```
# minimum cut bisection: returns a pair of sets of nodes  
partition = nx.community.kernighan_lin_bisection(G)
```

### 6.2.2 Data Clustering

As we have seen, communities in networks tend to group nodes that are similar to each other somehow: papers or websites dealing with the same or related topics, people working in the same area or department, proteins having the same or similar cellular functions, and so on. Therefore, community detection is a special version of the much more general problem of *data clustering* (i.e. grouping data elements into clusters based on some notion of similarity, such that elements in the same cluster are more similar to each other than they are to elements in different clusters). Data clustering offers a valuable set of concepts and tools that are regularly used in network clustering as well.

There are two main classes of algorithms for data clustering: *hierarchical clustering*, which delivers a nested series of partitions, and *partitional clustering*, which yields only one partition. Hierarchical clustering is much more frequently adopted in network community detection than partitional clustering, so let us briefly discuss it here.

The main ingredient is a *similarity measure* between nodes. Such a measure may be derived from specific properties of the nodes. For instance, in a social network, it could indicate how close the profiles of two individuals are according to their interests. If the nodes can be embedded in a geometric space, which is often possible to do via suitable transformations, the distance between the points corresponding to a pair of nodes can be used as a dissimilarity measure for the nodes, so that points that are nearer to each other