GRAPH DATA ANALYSIS

MARIA CAMERON, PERRIN RUTH

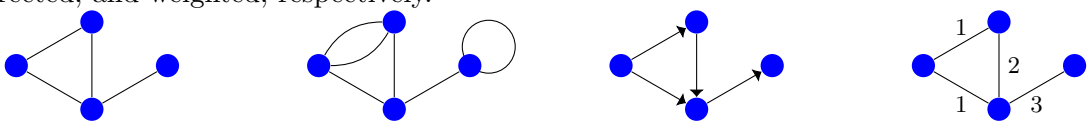
Contents

1. Introduction and Notation

For more resources read:

- chapters 1 and 2 in M. E. J. Newman's review paper [1],
- chapter 1 in A. L. Barabási's book " Network Science",
- the 3-page paper Nature by Watts and Strogatz (1998) [2].

Suppose we have a graph $G(V, E)$ where V is the set of vertices and E is the set of edges. In these notes we will assume G is a simple graph, i.e. G will not contain self-loops or multiple edges. Furthermore, we will assume that G is an unweighted graph. Outside of the next section we will also assume that G is undirected. For clarity the following figure contains a simple unweighted, undirected graph and modifications to it that are not simple, directed, and weighted, respectively.



Unless otherwise specified we will let the number of vertices and edges be $n = |V|$ and $m = |E|$. Frequently we will work with sparse networks, i.e. $m = O(n)$, which reduces the complexity of many classical problems in graph theory. One way to define a graph is by an $n \times n$ *adjacency matrix* A where $A_{i,j} = 1$ if there is an edge from i to j and $A_{i,j} = 0$ otherwise. If G is undirected then A is a symmetric matrix. Another way to define a graph is by an adjacency list which is a list containing n elements where each element is a list corresponding to the neighbors of a node of the graph. An adjacency list can be interpreted as a sparse representation of an adjacency matrix. These ways of defining a graph can easily be extended to weighted graphs.

In lab 11 you will be expected to write implementations for the algorithms in § 2 from scratch. Nonetheless, there are many programs available for studying networks. In Matlab one can create a graph object from an adjacency matrix `A` by running `G=graph(A)` for which there are many pre-defined functions. In Python, the networkx package is commonly used. Mathematica is useful for visualizing networks as well.

2. Basic algorithms for graph exploration

Given a graph, the first natural question is whether it is connected or not. If, not, what are its *connected components*?

Definition 1. *A connected component of a graph $G(V, E)$ is a subset of its vertices $V_1 \subset V$ such that*

(1) *for any pair of vertices $i, j \in V_1$ there is a path in $G(V, E)$ from i to j, and*
(2) *if there is a path from $i \in V_1$ to $j \in V$, then $j \in V_1$, i.e., the connected component is a maximal subset.*

If the graph $G(V, E)$ is directed, and we restrict ourselves to accounting only for directed paths, then such a component is called strongly connected.

The two straightforward algorithms that compute all connected components in time $O(n+m)$ are the *breadth-first search* (BFS) and *depth-fist search* (DFS) – see the so-called CLRS textbook [3](Chapter 22).

2.1. Breadth-first search.

The breadth-first search (BFS) is one of the simplest algorithms for searching a directed or undirected unweighted graph and is a building block for many important graph algorithms. It computes shortest paths from the *source* (any designated vertex) to all other vertices reachable from it.

The BFS uses the data structure called the *queue*. The key feature of a queue is the principle that *the first element entering it is the first one who exits it* (FIFO). This structure is denoted by the symbol `Q` in the pseudocode below (Algorithm 1). The color attribute of a vertex indicates if the vertex

- has never been in the queue (`WHITE`),
- is either in the queue now, or is just out of it and its adjacency list being explored now (`GRAY`),
- is out of the queue and adjacency list has been already explored (`BLACK`).

The other attributes of a vertex v are $v.d$, the shortest distance to the source, and $v.parent$, the predecessor of v in a shortest path from the source to it. The *predecessor subgraph* for

Algorithm 1: Breadth-first search.

```
Input: An unweighted graph G(V,E) , and a root vertex s.
for each vertex u ∈ V do
    u.color = WHITE ;
    u.d = ∞;
    u.parent = NIL;
end
s.color = GRAY;
s.d = 0;
Q = 0;
Add s to Q;
while Q is not empty do
    Take the first vertex u from Q;
    for each v adjacent to u do
        if v.color == WHITE then
            v.color = GRAY;
            v.d = u.d + 1;
            v.parent = u;
            Add v to the end of Q;
        end
    end
    u.color = BLACK;
end
```

Output: The connected component containing `s`, distances and paths from `s` to all vertices in each connected component, and the *breadth-first tree* (the predecessor subgraph).

a given graph $G(V, E)$ and a source vertex $s \in V$ is defined as $G_\pi = (V_\pi, E_\pi)$ where

$$V_\pi = \{v \in V \mid v.parent \neq \texttt{NIL}\} \cup \{s\}, \quad E_\pi = \{(v.parent, v) \mid v \in V_\pi \backslash \{s\}\}.$$

If not all vertices are discovered in a run of BFS, then we can start a new run with an undiscovered vertex chosen as a source.

Remark A *tree* is a special kind of undirected graph that contains no cycles. A tree with n vertices contains $n-1$ edges. Often one of the vertices is specified as a *root*. All vertices that are not a root and have only one adjacent edge are called *leaves*. A union of trees is called a *forest*.

Let us prove the correctness of the BFS [3] (Section 22.2).

Theorem 1. *Let $G(V, E)$ be a directed or undirected graph, and suppose BFS is run on G from a given source $s \in V$.*

(1) *BFS discovers every vertex $v \in V$ that is reachable from the source s;*
(2) *upon termination, for all $v \in V$, $v.d = \delta(s, v)$, the shortest path length from s to v.*
(3) *For any vertex $v \neq s$ that is reachable from s, one of the shortest paths from s to v is a shortest path from s to $v.parent$ followed by the edge $(v.parent, v)$.*

First we will prove some useful inequalities.

Lemma 1. *In the settings of Theorem 1:*

$$\delta(s, v) \leq \delta(s, u) + 1 \quad \forall (u, v) \in E. \tag{1}$$

$$v.d \geq \delta(s, v) \quad \forall v \in V. \tag{2}$$

Proof. Prove inequality (1). If u and v are both not reachable form s, then $\infty \leq \infty + 1$ which is true. If both u and v are reachable from s, the shortest path from s to v cannot be longer than the shortest path from s to u followed by the edge (u, v).

Inequality (2) will be proven by induction on the number of additions to the queue. The basis of the induction holds as $s.d = 0 \geq \delta(s, s) = 0$, and $v.d = \infty \geq \delta(s, v)$ for all $v \in V$. Assume that (2) holds right after the first k additions to the queue. Suppose a white vertex v is discovered during the search from u just dequeued. By induction hypothesis,

$$u.d \geq \delta(s, u).$$

Hence, using (1), we obtain (2):

$$v.d \geq u.d + 1 \geq \delta(s, u) + 1 \geq \delta(s, v)$$

The vertex v is then enqueued and made gray. Therefore, $v.d$ no longer can change. Hence the induction hypothesis is maintained. □

Lemma 2. *Suppose that vertices v and w are enqueued during the execution of BFS, and that v is enqueued before w. Then $v.d \leq w.d$ at the time that w is enqueued.*

We will prove that if the queue contains vertices $(v_1, \ldots, v_r)$ where v_1 is the head and v_r is the tail, then

$$v_r.d \leq v_1.d + 1 \quad \text{and} \quad v_i.d \leq v_{i+1}.d \quad \forall i = 1, \ldots, r-1. \tag{3}$$

Once we prove (3), the statement of Lemma 2 will follow immediately.

Proof. The proof is conducted by induction on the number of queue operations. Initially, when the queue contains only s, (3) holds automatically. Suppose that (3) holds at some step of the execution of BFS. To prove the induction step, we want to show that then it also holds after (i) dequeueing v_1 and (ii) enqueueing a new vertex v_{r+1}. Note that v_{r+1} may happen before dequeueing v_1.

As v_1 is dequeued, $v_1.d \leq v_2.d$ by induction assumption. Hence $v_r.d \leq v_1.d+1 \leq v_2.d+1$. Hence (3) is maintained.

Let $(v_1, \ldots, v_r)$ be the current queue, and we are enqueueing a new vertex v_{r+1} discovered from a dequeued vertex u. Since u used to be in the queue right before v_1,

$$v_1.d \geq u.d, \quad \text{and} \quad v_r.d \leq u.d + 1.$$

Hence

$$v_{r+1}.d = u.d + 1 \leq v_1.d + 1, \quad \text{and} \quad v_r.d \leq u.d + 1 = v_{r+1}.d.$$

Therefore, (3) is maintained as a result of any queue operation. □

Now we are ready to prove theorem 1.

Proof. We will prove statement (2) from converse. Let v be the vertex with the minimum $\delta(s, v)$ (length of the shortest path from s) that receives an incorrect d value. Clearly, $v \neq s$. Moreover, v must be reachable from s, since otherwise $\delta(s, v) = \infty \geq v.d \geq \delta(s, v)$ implying that $v.d = \delta(s, v)$.

Thus, $v.d > \delta(s, v)$. Let u be the vertex immediately preceding v in a shortest path from s to v. Then

$$\delta(s, v) = \delta(s, u) + 1.$$

Since $\delta(s, u) < \delta(s, v)$ and because of our choice of v, we have $u.d = \delta(s, u)$ which implies that

$$v.d > \delta(s, v) = \delta(s, u) + 1 = u.d + 1. \tag{4}$$

Now consider the moment of dequeueing u. At this time, v is either white, or gray, or black.

- If v is white, then $v.d = u.d + 1$ which contradicts (4).
- If v black, it means that it was removed from the queue before u, hence by Lemma 2, $v.d \leq u.d$ which contradicts (4).
- If v is gray, then it was discovered from a vertex t and painted gray, and t was removed from the queue before u. Hence, by Lemma 2,

  $$t.d \leq u.d \quad \text{and} \quad v.d = t.d + 1 \leq u.d + 1$$

  which contradicts (4).

Therefore, in all scenarios, we come to a contradiction. Hence $v.d = \delta(s, v)$ for all $v \in V$.

All vertices reachable from s must be discovered, since otherwise we would have

$$\infty = v.d > \delta(s, v).$$

Finally, we observe that if $v.parent = u$ then $v.d = u.d + 1$. Thus, we obtain a shortest path from s to v by adding $(v.parent, v)$ to a shortest path from from s to $v.parent$.

□

2.2. **Depth-first search.** The depth-first search (DFS), like BFS is suitable for unweighted graphs, directed or undirected. DFS creates a *predecessor subgraph* for a given graph $G(V, E)$ defined a bit differently from the one for BFS: $G_\pi = (V, E_\pi)$ where

$$E_\pi = \{(v.parent, v) \mid v \in V,\ v.parent \neq \texttt{NIL}\}.$$

This subgraph is called a *depth-first forest* comprising several *depth-first trees*. In an undirected graph, each depth-first tree spans a connected component of the graph. In general, each depth-first tree contains all vertices reachable from its root. In a directed graph, we say that v is reachable from s if and only if there is a directed path from s to v.

In DFS, the vertices are colored the same way as in BFS. This technique guarantees that each vertex ends up in exactly one depth-first tree, so that the trees are disjoint.

Besides the depth-first forest, DFS also *timestamps* each vertex. Each vertex has two timestamps.

- The first timestamp $v.d$ records when it is first *discovered*; $v.color$ changes from `WHITE` to `GRAY`.
- The second timestamp $v.f$ records when DFS *finishes* examining all v's adjacency list, i.e., discovers all vertices hanging down from v; $v.color$ changes from `GRAY` to `BLACK`.

Note that all timestamps are between 1 and $2|V|$ as there is exactly one discovery event and one finishing event for each vertex. For every vertex u,

$$u.d < u.f.$$

Algorithm 2–3 constitutes a pseudocode for DFS. Algorithm 3 is a *recursive function* as it calls itself. I'd like to remark that `time` can be made into a global variable. In `C` language, I would define it as a local variable of pointer-to-integer type. In the pseudocodes, I also define it as a local variable but in a manner suitable for Matlab implementation.

Algorithm 2: `DFS(G)`

```
Input: An unweighted graph G(V,E).
for each vertex u ∈ V do
    u.color = WHITE ;
    u.parent = NIL;
end
time = 0;
for each vertex u ∈ V do
    if u.color == WHITE then
        time = DFS-VISIT(G,u,time);
    end
end
```

Note that the output of the DFS depends on the order in which the vertices are discovered. The cost of DFS is $O(|V| + |E|)$.

Algorithm 3: `time = DFS-VISIT(G,u,time)`

Input: An unweighted graph `G(V,E)`, a selected vertex `u`, and the `time` variable.

```
time = time + 1;
u.d = time;
u.color = GRAY;
for each v adjacent to u do
    if v.color == WHITE then
        v.parent = u;
        time = DFS-VISIT(G,u,time);
    end
end
u.color = BLACK;
time = time + 1;
u.f = time;
```

Let us discuss some properties of DFS.

Theorem 2. (Parenthesis theorem) *Let DFS be applied to any directed or undirected graph $G(V, E)$. Then for any two vertices u and v, exactly one of the following three conditions holds:*

- *$[u.d, u.f] \cap [v.d, v.f] = \emptyset$, and neither u nor v is a descendant of the other in the depth-first forest,*
- *$[u.d, u.f]$ is entirely inside $[v.d, v.f]$, and u is a descendant of v in a depth-first tree, or*
- *$[v.d, v.f]$ is entirely inside $[u.d, u.f]$, and v is a descendant of u in a depth-first tree.*

Furthermore, v is a descendant of u in the depth-first forest if and only if

$$u.d < v.d < v.f < u.f.$$

The proof directly follows from the structure of Algorithm 2–3.

Theorem 3. (White-path theorem) *In a depth-first forest of a directed or undirected graph $G(V, E)$, vertex v is a descendant of vertex u if and only if at time $u.d$ (when the search discovers u), there is a path from u to v consisting entirely of white vertices.*

Proof. $\Longrightarrow$: If $v = u$ then the path consists of a single vertex u which is still white at the moment when we set $u.d$. If v is a proper descendant of u then $u.d < v.d$, so v is white at time $u.d$. Since v is any descendant of u, all vertices on the unique simple path from u to v in the depth-first forest are white at time $u.d$.

$\Longleftarrow$: Suppose that there is a path of white vertices from u to v at time $u.d$, but v does not become a descendant of u in the depth-first tree. Without the loss of generality, we assume that every vertex other than v along the path becomes a descendant of u. Let w be the predecessor of v in the path, so that w is a descendant of u. Then $w.f < u.f$ by

Theorem 2. Since v must be discovered after u is discovered but before w is finished, by Theorem 2 we have

$$u.d < v.d < w.f \leq u.f.$$

Hence, by Theorem 2, $[v.d, v.f]$ is entirely inside $[u.d, u.f]$ and hence v is a descendant of u. □

3. Random graph models

Network models help us to understand the meaning of such statistical network properties as average shortest path length, clustering coefficient, degree distribution – how they relate to the way the network has been formed and how they relate to each other.

3.1. **Erdös-Renyi random graphs.** The study of network models dates back to 1950s. Solomonoff and Rapoport (1951) [4] and, independently, Erdös and Renyi (1959) [5] proposed random graph models with similar properties:

- [4]: the random graph $G(n, p)$ which is an ensemble of random graphs with n vertices and in which each of $(1/2)n(n-1)$ possible edges exists with probability p;
- [5]: the random graph $G(n, N)$ which is an ensemble of random graphs with n vertices and N edges randomly selected out of $(1/2)n(n-1)$ possible edges.

In both works, the limit $n \to \infty$ was considered, and a number of asymptotic estimates were obtained. The model $G(n, p)$ makes analysis a bit easier, so, we will focus on it. We will also let n tend to infinity in a manner where *the mean degree*

$$z = 2\frac{pn(n-1)}{2n} = p(n-1) \tag{5}$$

remains constant. In this case, the model has a Poisson degree distribution. Indeed, the probability that a vertex has degree k is given by:

$$p_k = \binom{n-1}{k} p^k (1-p)^{n-k-1}. \tag{6}$$

Letting $n \to \infty$, fixing k, and setting $p = z/(n-1)$, we obtain the Poisson distribution:

$$p_k = \lim_{n\to\infty} \frac{(n-1)\dots(n-k)}{k!} \frac{z^k}{(n-1)^k} \left(1 - \frac{z}{n-1}\right)^{n-1} \left(1 - \frac{z}{n-1}\right)^{-k} = \frac{z^k e^{-z}}{k!}. \tag{7}$$

Due to this limit Erdös-Renyi random graphs are often called Poisson random graphs.

Both Solomonoff and Rapoport and Erdös and Renyi discovered the most interesting and important property of the Erdös-Renyi random graph: it possesses a phase transition. If the mean degree $z < 1$, all connected components of the graph are small (with high probability). In contrast, if $z > 1$, there appears the so-called *giant component* containing $O(n)$ vertices. Let u be the probability that a random vertex does not belong to the giant component. It is equal to the probability that none of its neighbors belongs to the giant component. This argument allows us to derive an equation for u:

$$u = \sum_{k=0}^{\infty} p_k u^k = \sum_{k=0}^{\infty} \frac{(uz)^k e^{-z}}{k!} = e^{z(u-1)}. \tag{8}$$

Observing that $S = 1 - u$ is the probability for a random vertex to belong to the giant component, we obtain an equation for S:

$$S = 1 - e^{-zS}. \tag{9}$$

The graphs of $1 - S - \exp(-zS)$ for $z = 1/2$, 1, and 2 are shown in Fig. 1.

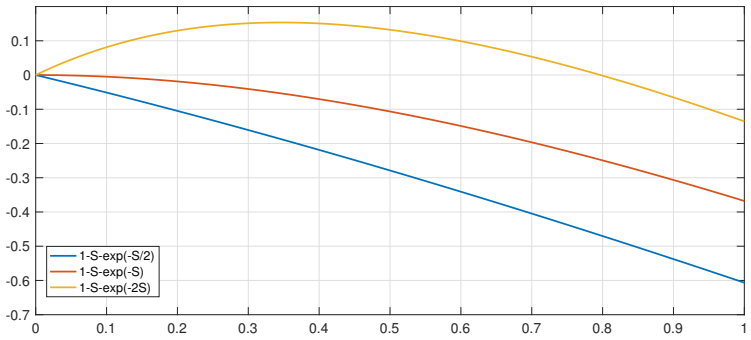


FIGURE 1. Graphs of $f(S) = 1 - S - \exp(-zS)$ where S is the fraction of vertices in the giant component for various mean degrees z. Using elementary calculus, one can show that $S = 0$ is the only root for $z \leq 1$, and there appears an additional solution $S > 0$ for $z > 1$.

In order to obtain more theoretical results for Poisson random graphs, we need to enhance our arsenal of analytical tools. A powerful tool is the method of generating functions widely used by M. Newman and collaborators. A beautiful book "Generatingfuncanology" by H. Wilf is available online.

3.2. **Generating functions.** This brief review follows the one in [6]. Let p_k be a discrete probability distribution. The index k runs from 0 to ∞. If the number of outcomes is finite, then the tail of the distribution is merely 0, and all results obtained in this section will still apply. A generating function for this probability distribution is defined as a power series:

$$G(x) := \sum_{k=0}^{\infty} p_k x^k. \tag{10}$$

Since p_k is a probability distribution, we have

$$G(1) = \sum_{k=0}^{\infty} p_k = 1. \tag{11}$$

The generating function allows us to restore the distribution according to the formula:

$$p_k = \frac{1}{k!} \frac{d^k G(x)}{dx^k}\bigg|_{x=0} = \frac{1}{2\pi i} \oint \frac{G(z)}{z^{k+1}} dz \tag{12}$$

where the integral is over some contour around the origin. Therefore, the function $G(x)$ encapsulates all information available about the distribution. In particular, it "generates" the distribution.

The generating function allows us to calculate all moments for the distribution. In particular, the mean is given by

$$\langle k\rangle = \sum_{k=0}^{\infty} kp_k = \sum_{k=1}^{\infty} kp_k 1^{k-1} = G'(1). \tag{13}$$

The mth moment is given by

$$\langle k^m\rangle = \sum_{k=0}^{\infty} k^m p_k = \left[\left(x\frac{d}{dx}\right)^m G(x)\right]_{x=1}. \tag{14}$$

If p_k is the distribution for a property of an object (i.e., the degree distribution for the vertices of a random graph) and $G(x)$ is its generating function, then the distribution of the sum of this property over m independent realizations is generated by $G(x)^m$. For example, let $m = 2$:

$$\begin{aligned}[G(x)]^2 &= \left[\sum_{k=0}^{\infty} p_k x^k\right]^2 = \sum_{j,k} p_j p_k x^{j+k} \\ &= p_0p_0x^0 + (p_0p_1 + p_1p_0)x + (p_0p_2 + 2p_1p_1 + p_2p_0)x^2 + \dots.\end{aligned}$$

3.3. **Poisson random graphs: mean size of non-giant component.** We will follow the discussion in [6]. For the Poisson random graph with mean degree z, the generating function for the degree distribution is given by

$$G_0(x) = \sum_{k=0}^{\infty} e^{-z}\frac{z^k}{k!}x^k = e^{z(x-1)}. \tag{15}$$

As it should be, $G_0(1) = e^0 = 1$. Let us check that the mean degree is z. Indeed,

$$G_0'(1) = ze^{z(x-1)}\Big|_{x=1} = z.$$

Now we will calculate the average size of non-giant component, i.e., the expected size of the component where a randomly picked vertex that does not lie in the giant component belongs to. Let us pick a vertex v and an edge emanating from it. Suppose that this edge leads to another vertex w. We want to obtain the distribution for the so-called *excess degree* of w, i.e, the probability q_k that k edges other than (v, w) are incident to w. Since the probability to arrive to w from v is proportional to the degree of w which is $k + 1$, we get:

$$q_k = \frac{(k+1)p_{k+1}}{\sum_{k=1}^{\infty} kp_k} = \frac{(k+1)p_{k+1}}{z} = e^{-z}\frac{(k+1)z^{k+1}}{z(k+1)!} = e^{-z}\frac{z^k}{k!} = p_k. \tag{16}$$

Therefore, for the Poisson random graph, the distributions p_k and q_k coincide! Hence, for the Poisson model, the generating function $G_1(x)$ for the excess degree distribution coincides with $G_0(x) = e^{z(x-1)}$.

Let $H_1(x)$ be the generating function for the total number of vertices reachable by choosing a random edge, not belonging to the giant component if any, and following to one of its ends. It must satisfy the following self-consistency equation obtained under the assumption that all small components are tree-like, i.e., contain no cycles:

$$H_1(x) = xq_0 + xq_1H_1(x) + xq_2[H_1(x)]^2 + xq_3[H_1(x)]^3 + \ldots = xG_1(H_1(x)). \tag{17}$$

Let us clarify why we think about each non-giant component as a tree. Remember that we assume that n is very large and the probability of an edge between any pair of vertices scales as $O(1/n)$. The number of vertices in a non-giant component is a small fraction of n. Hence the probability that there is an extra edge connecting two vertices of the same small component also scales as $O(1/n)$ and hence tends to zero as $n \to \infty$. In [6], (17) is illustrated with the following diagram:

Assume for now that there is no giant component. We randomly select a vertex v and look at the probability distribution for the size of the connected component containing it. It is generated by the function

$$H_0(x) = xp_0 + xp_1H_1(x) + xp_2[H_1(x)]^2 + xp_3[H_1(x)]^3 + \ldots = xG_0(H_1(x)). \tag{18}$$

The mean size of a non-giant component is given by

$$\langle s\rangle = H_0'(1) = G_0(H_1(1)) + G_0'(H_1(1))H_1'(1) = 1 + G_0'(1)H_1'(1). \tag{19}$$

The derivative $H'(1)$ can be found from (17):

$$H_1'(1) = G_1(H_1(1)) + G_1'(H_1(1))H_1'(1) = 1 + zH_1'(1). \tag{20}$$

Therefore,

$$H_1'(1) = \frac{1}{1-z}. \tag{21}$$

Hence, we find that

$$\langle s\rangle = 1 + \frac{z}{1-z} = \frac{1}{1-z} \tag{22}$$

These results can be extended to random graphs with a giant connected components as shown in [6].

3.4. **Random graphs with arbitrary degree distributions.** While Poisson random graphs offer an analytically solvable model in the limit $n \to \infty$, they do not resemble real-life networks in a number of aspects. One such an aspect it the degree distribution. In many real-world networks, a power degree distribution is observed – see Fig. 3.2 in [1]:

the internet, the World-Wide Web, protein interactions, collaborations in mathematics, citations networks all exhibit degree distribution of the form

$$p_k = \frac{k^{-\alpha}}{\zeta(\alpha)}, \quad \text{where} \quad \zeta(\alpha) = \sum_{k=1}^{\infty} k^{-\alpha} \ \text{ is the Riemann zeta function.} \tag{23}$$

Note that the Riemann zeta function is finite for all α such that $\mathsf{Re}(\alpha) > 1$. Another degree distribution that is observed in real-world networks is exponential, e.g., in the power-grid network (Fig. 3.2 in [1]).

Motivated by this fact, random graphs with a specified degree distribution were introduced – see [6] and references therein. These graphs can be sampled by generating vertices with numbers of "stubs" distributed according to the given distribution p_k and then randomly matching the stubs. The only restriction of this approach is that the total number of stubs must be even.

The method of generating functions used in the previous section is straightforwardly transferable to a model with specified degree distribution. As before, we obtain the excess degree distribution

$$q_k = \frac{(k+1)p_{k+1}}{\sum_{j=1}^{\infty} jp_j} = \frac{(k+1)p_{k+1}}{z} \quad \text{where} \quad z = \sum_{j=1}^{\infty} jp_j \quad \text{is the mean degree.}$$

The distributions p_k and q_k are generated by $G_0(x)$ and $G_1(x)$ respectively which no longer need to coincide. Note that

$$G_1(x) = \frac{G_0'(x)}{z}. \tag{24}$$

Random graphs with a given degree may or may not have a giant component, and, what is most interesting for me, a criterion for the existence of giant component can be derived [6].

3.4.1. *Configuration Model.* The configuration model was initially developed in the late 1970s in order to generate random graphs from a given degree distribution. In particular, this model draws a graph from the possible graphs that satisfy the given degree distribution uniformly at random. Thus, the configuration model works well as a baseline random graph model while still accounting for varying degrees. Some resources for the history and analysis of this model can be found in §4.2 of Newman's review paper [1].

Assume we have a probability mass function for which we can generate a degree distribution. Then if a given node has degree d we assign it d "stubs", and pairs of stubs are combined as edges uniformly at random. A pseudocode version of this is given in Algorithm 4.

When running the configuration model, it is possible to generate self-loops and multi-edges. However, the expected number of these errors is constant in n. Typically, one corrects for this by removing self-loops and multi-edges. If the configuration model were adjusted to not allow erroneous edges then it would have a different probabilistic prior.

Algorithm 4: Configuration Model

Input: Degree sequence `d(0),d(1),...,d(n)`
Initialize an empty list `L`;
for `i = 1:n` **do**
  Add `d(i)` copies of `i` to `L`;
end
Attach elements from `L` pairwise uniformly at random;

3.4.2. *Phase transition.* Let us pick a vertex v and consider its first neighbors, second neighbors, ..., mth neighbors, and so on as in M. Newman, "Random graphs as models for networks" (2002). The excess degree distribution for the first neighbors is q_k. The mean excess degree is given by

$$\sum_{k=0}^{\infty} kq_k = \frac{1}{z}\sum_{k=0}^{\infty} k(k+1)p_{k+1} = \frac{1}{z}\sum_{k=0}^{\infty}(k-1)kp_k = \frac{\langle k^2\rangle - \langle k\rangle}{\langle k\rangle}, \tag{25}$$

where $z \equiv z_1 \equiv \langle k\rangle$ is the mean degree. The expected number of second neighbors is equal to the mean excess degree of the first neighbors times the expected number of first neighbors:

$$z_2 = \langle k\rangle \frac{\langle k^2\rangle - \langle k\rangle}{\langle k\rangle} = \langle k^2\rangle - \langle k\rangle \tag{26}$$

Then we take an arbitrary second neighbor of v and apply the same argument to calculate the expected number of third neighbors. The average excess degree for second neighbors is still given by (25). The probability that any of these excess edges reconnect to the first neighbor of v or to v itself tends to zero as $n \to \infty$. Therefore, the expected number of third neighbors is the expected number of second neighbors times the mean excess degree:

$$z_3 = z_2 \frac{\langle k^2\rangle - \langle k\rangle}{\langle k\rangle} = \frac{z_2}{z_1} z_2 = \left[\frac{z_2}{z_1}\right]^2 z_1.$$

By a similar argument, we find the expected number of mth neighbors:

$$z_m = \frac{z_2}{z_1} z_{m-1} = \left[\frac{z_2}{z_1}\right]^{m-1} z_1. \tag{27}$$

Equation (27) allows us find a criterion for existence of the giant component. Summing (27) over all m and adding the vertex v, we find the expected size of a connected component containing v:

$$\langle s\rangle = 1 + z_1 \sum_{m=1}^{\infty} \left[\frac{z_2}{z_1}\right]^{m-1} = \begin{cases} 1 + \frac{z_1^2}{z_1 - z_2}, & z_1 > z_2 \\ \infty, & \text{otherwise.} \end{cases} \tag{28}$$

If this expression is finite, i.e., if $z_2 < z_1$, there is no giant component. Otherwise, there is. Recalling (26) for z_2, we write the condition for the phase transition: $z_1 = z_2$, i.e.,

$$\boxed{0 = z_2 - z_1 = \langle k^2 \rangle - 2\langle k \rangle = \sum_{k=0}^{\infty} k(k-2)p_k.} \tag{29}$$

We remark that (29) for the phase transition was originally derived by Molloy and Reed (1995) [7] using a different approach.

3.4.3. *Average shortest path length.* Equation (27) allows us to estimate the average shortest path length in the giant component of a random graph with a specified degree distribution provided that there is one. Let us assume that the giant component embraces almost all vertices in the graph. We assume that $z_2 \gg z_1$ and set the expected size of lth neighborhood to be equal to the whole graph n:

$$z_l = \left[\frac{z_2}{z_1}\right]^{l-1} z_1 \simeq n.$$

For here, we obtain the following estimate for the average shortest path length:

$$l \simeq 1 + \frac{\log(n/z_1)}{\log(z_2/z_1)}. \tag{30}$$

For the special case of Erdös-Renyi random graph we have $z_2 = z_1^2 = z^2$ (as $G_1 = G_0$), hence

$$l \simeq 1 + \frac{\log n - \log z}{\log z} = \frac{\log n}{\log z}.$$

3.4.4. *Clustering coefficient.* Another important quantity of interest is the *clustering coefficient*. There are different non-equivalent definitions of clustering coefficient. Here, we will define it as the ratio of the tripled number of triangles in the network to the number of connected triples:

$$C = \frac{3 \times \text{number of triangles in the network}}{\text{number of connected triples of vertices}}. \tag{31}$$

Let v be an arbitrary vertex, and let k_i and k_j be excess degrees of two of its neighbors. The distributions for the excess degrees are q_k. The probability that these two neighbors are connected to each other is

$$\frac{k_i k_j}{nz}. \tag{32}$$

Indeed, we have k_i stubs emanating from i. For each of them, the probability to connect to one out of k_j stubs of j is k_j/nz as nz is the total number of stubs. Averaging this probability over the joint distribution for k_i and k_j which is merely the square of the distribution for the excess degree due to the independence of k_i and k_j and recalling (25), we get:

$$C = \frac{\langle k_i k_j \rangle}{nz} = \frac{1}{nz} \sum_{k,l=0}^{\infty} k l q_k q_l = \frac{1}{nz} \left[\sum_{k=0}^{\infty} k q_k\right]^2 = \frac{1}{nz} \left[\frac{\langle k^2 \rangle - \langle k \rangle}{\langle k \rangle}\right]^2. \tag{33}$$

This expression can be rewritten in terms of the coefficient of variation of the degree distribution c_v defined by

$$c_v^2 := \frac{\langle k^2\rangle - \langle k\rangle^2}{\langle k\rangle^2}, \tag{34}$$

i..e., c_v is the ratio of the standard deviation to the mean. Using it, we obtain:

$$C = \frac{z}{n}\left[\frac{\langle k^2\rangle - \langle k\rangle^2 + \langle k\rangle^2 - \langle k\rangle}{\langle k\rangle^2}\right]^2 = \frac{z}{n}\left[c_v^2 + \frac{z-1}{z}\right]^2. \tag{35}$$

Thus, we see that C scales with n as $O(z/n)$, however, the factor by which z/n is multiplied can be large.

3.4.5. *Average size of non-giant components.* In a manner similar to the one used for the Poisson random graphs, one can calculate the average size of a non-giant component [6]:

$$\langle s\rangle = \frac{H_0'(1)}{H_0(1)} = 1 + \frac{zu^2}{[1-S][1-G_1'(u)]}, \tag{36}$$

where $u \equiv H_1(1)$ is the smallest non-negative solution to $u = G_1(u)$, and S is the fraction of graph occupied by the giant component: $S = 1 - G_0(u)$.

Exercise Give a detailed derivation of (36).

4. Percolation

Broadly speaking, this section will focus on the construction of random graphs within networks. From a destructive point of view we may remove nodes or edges from a network and look at how this effects its structure. This is applicable to the robustness of the internet and technological networks. From a more constructive point of view we may study the affect of nodes/edges with varying states. For instance, we might look at the spread of an epidemic or information along a social network. For general resources on these topics read:

- chapter 8 of M. E. J. Newman's review paper [1]
- chapters 8 and 10 in A. L. Barabási's book Network Science

4.1. **Node Percolation.** Consider a process where we remove nodes uniformly at random from a graph G that is assumed to have a giant connected component. One natural question is how many nodes can we remove before destroying the giant component? The analysis of this will follow the paper [8].

We will assume G is a random graph (i.e. from the configuration model) with probability p_k of having a node of degree k. Let q be the probability that a node is removed from G. Following the notation of Cohen et al., quantities after the breakdown of G will be denoted by a prime, e.g. the new graph will be called G' and degree distribution following p_k'. If the degree of a given node is initially k_0 then the probability its new degree $0 \le k \le k_0$

follows a binomial distribution $\text{Bin}(1-q, k_0)$. The updated degree distribution may then be computed as follows

$$p'_k = \sum_{k_0=k}^{\infty} p_{k_0} \binom{k_0}{k} (1-q)^k q^{k-k_0} \tag{37}$$

The expected degree of G' is then

$$\begin{aligned}
\langle k \rangle' &= \sum_{k=0}^{\infty} k \sum_{k_0 \geq k} p_{k_0} \binom{k_0}{k} (1-q)^k q^{k-k_0} \\
&= \sum_{k_0=0}^{\infty} p_{k_0} \sum_{1 \leq k \leq k_0} k \binom{k_0}{k} (1-q)^k q^{k-k_0} \\
&= \sum_{k_0=0}^{\infty} (1-q) k_0 p_{k_0} \sum_{1 \leq k \leq k_0} \binom{k_0-1}{k-1} (1-q)^{k-1} q^{k-k_0} \\
&= (1-q)\langle k_0 \rangle
\end{aligned}$$

A similar argument may be used to show that $\langle k^2 \rangle' = (1-q)^2 \langle k_0^2 \rangle + q(1-q)\langle k_0 \rangle$. Then, by the Molloy-Reed criterion the giant connected component breaks when $q = q_c$

$$(1-q_c)^2 \langle k_0^2 \rangle + q_c(1-q_c)\langle k_0 \rangle - 2(1-q_c)\langle k_0 \rangle = 0. \tag{38}$$

If we define $\kappa_0 = \langle k_0^2 \rangle / \langle k_0 \rangle$ then

$$\boxed{q_c = \frac{\kappa_0 - 2}{\kappa_0 - 1}} \tag{39}$$

For a random Erdös-Renyi graph with mean degree $z = \langle k_0 \rangle$ we may use a Poisson distribution to show $\kappa_0 = 1 + z$. Thus, the main connected component breaks when

$$q_c = 1 - \frac{1}{z}. \tag{40}$$

Next, assume that the degree distribution follows a power law $p_k = ck^{-\alpha}$. It is typical to assume that there is a minimum and maximum degree m and K where $K \to \infty$ as $n \to \infty$.Here we may choose K such that the probability a node has degree higher than it is $\int_K^{\infty} x^{-\alpha} dx = \frac{1}{n}$. Then, the moments of a power law may be estimated by the integral

$$\langle k_0^{\ell} \rangle = c \int_m^K x^{-\alpha} x^{\ell} dx.$$

With the additional assumption $K \gg m$ we find

$$\kappa_0 \approx \left| \frac{2-\alpha}{3-\alpha} \right| \times \begin{cases} m, & \text{if } \alpha > 3 \\ m^{\alpha-2} K^{3-\alpha}, & \text{if } 2 < \alpha < 3 \\ K, & \text{if } 1 < \alpha < 2. \end{cases} \tag{41}$$

For $\alpha > 3$ we have the parameter κ_0 is fixed and we get a fixed breakdown point q_c. For $\alpha < 3$ we have κ_0 grows with K (or n). Thus, as $n \to \infty$ the point at which the connected component breaks down is

$$q_c = \frac{1 - 2/\kappa_0}{1 - 1/\kappa_0} \to 1$$

or the network is immune to these random attacks. As a point of reference Cohen et al. [8] consider the internet which has $\alpha \approx 2.5$ making it robust to these attacks.

4.2. **SIR Model.** Next we will consider the formation of epidemics on a network following the formulation in [9]. In the SIR model individuals may be in 3 states

- **S**usceptible: a healthy individual who has yet to catch a disease,
- **I**nfected: an individual with the disease that may spread it to others,
- **R**ecovered: an individual who no longer has the disease (or died).

First, we consider the all-to-all case, i.e. the any infected individual can infect any susceptible individual without bias. In this scenario we need only look at the fraction of individuals in each state. Namely, $s(t)$, $i(t)$, and $r(t)$ are the proportions of the population that are susceptible, infected, and recovered at time t, respectively. Assume that there is a uniform rate of transmission and recovery. This results in a system of ODEs:

$$\begin{cases} \frac{ds}{dt} = -\beta is \\ \frac{di}{dt} = \beta is - \gamma i \\ \frac{dr}{dt} = \gamma i \end{cases} \tag{42}$$

where $i + s + r = 1$. This was originally developed L. Reed and W. H. Frost in the 1920s, but it was never published. At the start of a disease the infected population is small and $s \approx 1$. Thus, at the start of a disease we get the infected ODE is

$$\frac{di}{dt} \approx \beta i(1 - \frac{\gamma}{\beta}). \tag{43}$$

This motivates the constant $R_0 := \frac{\gamma}{\beta}$. For a short time if $R_0 > 1$ the infected population grows exponentially and for $R_0 < 1$ it decays exponentially. Thus, we find the epidemic threshold $R_0 = 1$.

Next, we will consider the SIR model on networks, initially developed in [10]. In this model, individuals are represented by nodes and contacts by edges. Consider the case where u and v are two neighboring nodes where u is infected and v is susceptible. Then the disease may transfer from u to v with rate β, and u recovers with rate γ. Then the probability v catches the disease from u is given by the race condition $T = \frac{\beta}{\beta+\gamma}$. The parameter T is also known as the transmission rate.

One way of simulating the SIR model is by discrete time steps. Over an interval of length Δt an infection spreads over an edge with probability $\beta \Delta t$ and each infected node recovers with probability $\gamma \Delta t$. As initial conditions, it is sufficient to infect a single node at random. Typically the time a node is in the infected state is set to a fixed value τ. This is arguably more realistic and does not complicate future analysis. However, changing the infection time to a constant changes the transmission rate to $T = 1 - e^{-\beta\tau}$.

The spread of disease over a network can also be described by a deterministic model. Let the probability that node u is susceptible, infected, or recovered be given by $s_u(t)$, $i_u(t)$, and $r_u(t)$, respectively. Then, these equations can be evolved by

$$\begin{cases} \frac{ds_u}{dt} = -\beta s_u \sum_{v\sim u} i_v \\ \frac{di_u}{dt} = \beta s_u \sum_{v\sim u} i_v - \gamma i_u \\ \frac{dr_u}{dt} = \gamma i_u. \end{cases} \tag{44}$$

4.3. **Edge Percolation and SIR Model on Random Networks.** As noted in the previous section the probability that a disease can spread over an edge is given by T. This probability does not depend on which node gets infected first. If we only wanted to find the total number of individuals it would be sufficient to replace our dynamics with passing on the disease with probability T. Furthermore, we could initially assign each edge as a transmitting edge. Then a node will get infected if it is connected to the source by transmitting edges only. An epidemic would be described by a giant connected component formed by transmitting edges. The following will use a generating function approach [9].

Let $G_0(x)$ be the generating function for the degree distribution of our graph. Let $G_1(x) = G_0(x)/z$ be the generating function for the degree distribution. The generating function for transmitting edges from a node is given by

$$\begin{aligned} G_0(x;T) &= \sum_{m=0}^{\infty}\sum_{k=m}^{\infty} p_k \binom{k}{m} T^m (1-T)^{m-k} x^m \\ &= \sum_{k=0}^{\infty} p_k \sum_{m=0}^{k} \binom{k}{m} (xT)^m (1-T)^{m-k} \\ &= \sum_{k=0}^{\infty} p_k (1+(x-1)T)^k = G_0(1+(x-1)T) \end{aligned}$$

The generating function for excess transmitting edges is similarly given by $G_1(x;T) = G_1(1+(x-1)T)$.

Next we want to compute a threshold for the size of the giant connected component. Let $P_S(T)$ be the probability that a randomly chosen (source) node causes an outbreak of size S. The outbreak size is simply the size of the connected component of the subgraph induced by transmitting edges. Define $H_0(x;T)$ to be the generating function with coefficients $P_S(T)$. Let $H_1(x;T)$ be the generating function for outbreak sizes along one side of an edge. We will also assume the subgraph induced by transmitting edges is locally tree-like. The size of our outbreak is 1 (for the source infection) plus the size of the outbreak caused by each person infected by the source. In generating function terminology we get the recurrence

$$H_0(x;T) = x^1 G_0(H_1(x;T);T). \tag{45}$$

Similarly we find the recurrence for the excess outbreak distribution to be

$$H_1(x;T) = xG_1(H_1(x;T);T) \tag{46}$$

To solve the recurrence for H_1 it may work to set an initial guess x_0 and iterate $x_{n+1} = x_n G_1(H_1(x_n;T);T)$. Otherwise a nonlinear solver can be used. To find the coefficients of $H_0(x;T)$ use Eq. 12.

The mean size of the outbreaks is given by

$$\langle s\rangle = \frac{dH_0(x;T)}{dx}\Big|_{x=1} = 1 + G_0'(1;T)H_1'(1;T) \tag{47}$$

This may be simplified by differentiating Eq. 46:

$$H_1'(1;T) = 1 + G_1'(1;T)H_1'(1;T) = \frac{1}{1-G_1'(1;T)}. \tag{48}$$

Thus, our average outbreak size is

$$\langle s\rangle = 1 + \frac{TG_0'(1)}{1-TG_1'(1)}. \tag{49}$$

The critically transmissability for an epidemic, T_c, occurs when $\langle s\rangle$ diverges to infinity, i.e.:

$$\boxed{T_c = \frac{1}{G_1'(1)}}. \tag{50}$$

For an Erdös-Renyi random graph with mean degree z we have that $G_1'(z) = G_0'(z) = z$. Thus, we have for these graphs $T_c = \frac{1}{z}$. Recall that $T = \frac{\beta}{\beta+\gamma}$ where β is the rate of infection along an edge and γ is the rate of recovery. To imitate the all-to-all case define $\bar{\beta} = \beta z$ as the average rate of infection from all neighbors. Similarly, define $\bar{R_0} = \frac{\bar{\beta}}{\gamma}$. Then, we find an epidemic occurs when

$$\bar{R_0} = \frac{z}{z-1} > 1. \tag{51}$$

In other terms, it is harder to create an epidemic in an Erdös-Renyi random graph than the all to all case. However, they are equal in the limit $z \to \infty$.

References

[1] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[2] D. J. Watt and S. H. Strogatz, "Collective dynamics of "small-world" networks," *Nature*, vol. 393, pp. 440–442, 1998.

[3] T. H. Cormen, E. Leiserson, Charles, R. L. Rivest, and C. Stein, *Introduction to algorithms, Third edition*. The MIT press, 2009.

[4] R. Solomonoff and A. Rapoport, "Connectivity of random nets," *Bulletin of Mathematical Biophysics*, vol. 13, pp. 107–117, 1951.

[5] P. Erdös and A. Renyi, "On random graphs," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[6] M. E. J. Newman, H. Strogats, Steven, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review E*, vol. 64, p. 026118, 2001.

[7] M. Molloy and B. Reed, "A critical point for random graphs with a given degree sequence," *Random Structures and Algorithms*, vol. 6, pp. 161–180, 1995.

[8] R. Cohen, K. Erez, D. Ben-Avraham, and S. Havlin, "Resilience of the internet to random breakdowns," *Physical review letters*, vol. 85, no. 21, p. 4626, 2000.

[9] M. E. Newman, "Spread of epidemic disease on networks," *Physical review E*, vol. 66, no. 1, p. 016128, 2002.

[10] P. Grassberger, "On the critical behavior of the general epidemic process and dynamical percolation," *Mathematical Biosciences*, vol. 63, no. 2, pp. 157–172, 1983.