UNIVERSITY OF OXFORD

Robustness in Interdependent Networks

MMath CD Dissertation

Candidate Number
267492

March 2014

Abstract

The field of network science has seen massive growth in the last 15 years. Tools have been developed to study the properties of abstract networks, as well as problems involving real-world systems. One such problem concerns the robustness of a network: how well the network continues to function when parts of its structure are destroyed. A rapidly emerging area of network science is the study of multilayer networks. We investigate the robustness of one type of multilayer network: interdependent networks.

We use probability generating functions, along with computer simulations, to analyse the behaviour of interdependent networks under random failure and targeted attacks. Using Erdős-Rényi networks and scale-free networks as examples, we find that the same network can react quite differently, depending on the type of attack. Moreover, for scale-free networks, the interdependent system sometimes displays very different behaviour to the equivalent non-interdependent network.

Acknowledgements

Firstly, I thank my supervisor for his encouragement, advice, and enthusiasm in the field of network science. I also thank James Gleeson, Davide Cellai, and others at the University of Limerick for a warm welcome and interesting discussions. I am grateful to Olaf Sporns, Mikail Rubinov, and the other contributors to the Brain Connectivity Toolbox for MATLAB, as well as Alan Taylor and Des Higham, the developers of the CONTEST toolbox for MATLAB.

Contents

List of Figures

1. Introduction

1.1 Motivation

The study of networks is the study of individuals (called *nodes*), and the connections between them (called *edges*) [23]. For example, networks may be:

- Biological: Food webs, where nodes are organisms and an edge between two organisms indicates that one is eaten by the other; or protein-protein interaction networks.
- Technological: Computer networks, where nodes are terminals and edges are the physical cables that connect them; or the World Wide Web, where the nodes are websites, which are connected by an edge if one site contains a link to the other.
- Social: Nodes correspond to individuals in a group of people, and an edge represents a relationship, such as friendship or enmity, between two people.

We often picture a network as a set of points in space, connected by lines that join one point to another. Two examples of real-life networks, represented in this way, are shown in Fig 1.1.

Specific networks, such as those described above, have been studied by people working in a variety of fields for several decades [22]. However, the pace of mathematical research into networks in general has significantly increased in recent years. One of the reasons for this is the technology boom, which has resulted in network data becoming much easier to assemble and to share (p8, [2]).

The study of networks can be roughly divided into the study of structural properties of networks, and the study of dynamical processes on networks. One dynamical property is the "robustness" of a network: the resilience of the network against the removal of a subset of its nodes or edges. The study of the robustness of networks has clear applications to real-life networks. On the one hand, we might ask how to design a rail network that remains functional, even when some routes are out of service. On the other hand, we might want to determine the best individuals to vaccinate in order to minimise the size of an outbreak of a disease. Investigating the robustness of networks allows us to suggest answers to each of these questions [15].

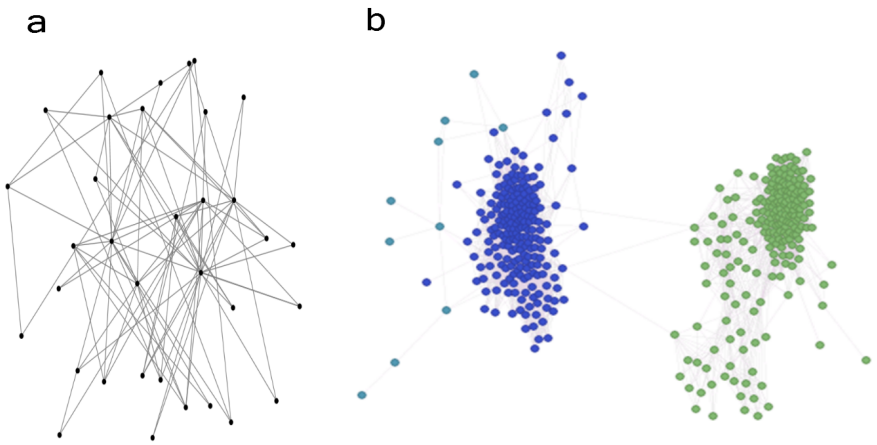

Figure 1.1: Examples of networks. Panel (a) shows the social network of members of a karate club, and panel (b) shows a Facebook network. The image in panel (a) was created from the data in [29], using the NodeXL template for Microsoft Excel [25]. The image in panel (b) was generated using Facebook Report for Wolfram|Alpha [13].

Much of the work on network robustness concerns single networks, that do not interact with other networks in any way. However, many real networks are fundamentally linked to other networks, and by considering these networks in isolation we run the risk of overlooking features that could significantly affect the behaviour of the network. An example is the electrical blackout that took place in Italy in 2003 [6]. Power stations relied on an Internet communications network, which itself needed electricity from the power stations. When a few power stations failed, these dependencies caused the failures to cascade throughout the system, causing widespread disruption.

Consequently, the focus is now shifting towards so-called "multilayer" networks. The 2014 article by Kivelä et al. ([19]) is a comprehensive review of the current state of research regarding multilayer networks. The article gives a general framework for multilayer networks which allows the modelling, amongst other things, of networks whose structure changes with time, networks whose edges can be divided into different categories, and networks that depend on other networks. In this report, however, we concern ourselves only with networks that depend on other networks, such as in the example in the previous paragraph. These are called *interdependent* networks.

1.2 Structure of the Report

Chapter 2 contains background information that is necessary for the calculations in later chapters. I give some basic definitions from graph theory, and define generating functions and random graphs: two useful tools for studying networks. I also summarise the work that has been done on robustness in single-layer

networks, that we can later compare against results for interdependent networks.

Chapter 3 is based on a 2010 paper by Buldyrev et al. ([6]), which analyses the robustness of coupled networks against the failure of nodes uniformly at random (I define coupled networks on p 13). I introduce a process of cascading failures in coupled networks, as described in [6], and replicate results from the paper. I provide detailed derivations, that are not included in the original paper. Finally, I perform computer simulations for two types of coupled networks, and compare some of the simulations with results that were derived analytically.

Chapter 4 concerns targeted attack, where the probability that a node will fail depends on how many other nodes it is joined to. This chapter follows the argument in a 2011 paper by Huang et al. ([16]), where the targeted attack case on coupled networks is reduced to a failure-uniformly-at-random case on modified networks. I duplicate results from [16], then use an analytic method based on the work in [16] to derive results for the same example networks as in Chapter 3. I compare these results to computer simulations.

Chapter 5 contains my own work, concerning robustness against attacks where nodes are targeted based on how many "second neighbours" they have. I attempt to apply a modified version of the method from Chapter 4 to the problem, explain how the method fails, and present the results of computer simulations.

2. Preliminary Information

2.1 Basic Definitions

In order to study networks from a mathematical perspective, we need a mathematical representation for networks. Typically, networks are represented as graphs, which are defined as follows [4].

Definition 2.1. *A **graph** is an pair of sets $G = (V, E)$. The set $V(G) = V$ is the set of **vertices**, or **nodes**, of the graph; $E(G) = E \subseteq V^{(2)}$ (where $V^{(2)} = \{\{u, v\} : u, v \in V, u \neq v\}$) is the set of **edges** of the graph.*

One simple way of comparing two graphs is by calculating the degree distributions.

Definition 2.2. *If u and v are nodes in a graph G, u is a **neighbour** of v if $\{u, v\} \in E(G)$.*

Definition 2.3. *An edge $\{u, v\}$ is **incident** to a vertex w if either $u = w$ or $v = w$.*

Definition 2.4. *For a node $w \in V$, the **degree** $d(w)$ of w is the number of edges that are incident to w.*

Definition 2.5. *The **degree distribution** of a graph G is $P_G = (P_G(0), P_G(1), P_G(2), \ldots)$, where $P_G(k)$ is the probability that a node chosen uniformly at random has degree k. If G is a given graph, with size $|G| = n$, this is equivalent to*

$$P_G(k) = \frac{1}{n}|\{v \in V : d(v) = k\}|.$$

In this report, we investigate whether or not networks remain "functional" under various conditions, where a network is functional if information can spread throughout the network. Therefore, we need to define what it means for parts of a network to be connected.

Definition 2.6. *Two nodes $u, v \in V(G)$ are **connected** if there is a path between u and v along edges of G. A set C of nodes in G is **connected** if any two nodes in C are connected. A connected set C is a **connected component** of G if there is no node $v \in V \setminus C$ such that $C \cup \{v\}$ is connected. Often, connected components are simply called "components".*

2.2 Example Graphs and Networks

2.2.1 Scale-Free Networks

A large number of observed networks have degree distributions that appear to have a "power-law tail", where for suitably large k the probability of a randomly chosen node having degree k is approximately $Ck^{-\beta}$, for constants C and β [23]. Networks with such a degree distribution have a small number of very high degree nodes, and a large number of nodes with low degrees. These networks are often called *scale-free* networks, since the distribution of the degrees within a given range stays about the same if the range is scaled by some constant factor. There is some debate about when it is appropriate to model the degree distribution of a network using a power law [26]. Networks for which it has been shown that a power-law distribution is a reasonable fit for the degree distribution include a protein-protein interaction network, and a citation network (where nodes are academic papers, and an edge exists between two papers if one cites the other) [8].

2.2.2 Random Graphs

If we know the exact structure of a network, we can translate it directly into a graph, and then study the graph. In most cases, however, either the network is very large or complex, or we wish to study a collection of networks that share certain features but have non-identical structures. In these situations, random graphs can be very helpful.

A random graph is a graph that is generated by some random process, but in which certain properties are ensured, often by specifying the values of a set of parameters [23]. Consequently, the study of random graphs usually leads to results that describe the average or expected behaviour of a family, or "ensemble", of networks. We can consider random graphs of fixed size $|G| = |V(G)| = N$, or derive results for arbitrarily large graphs by considering a random-graph-ensemble as $|G|$ approaches infinity. In this report, we will use random graphs to study scale-free networks, as described above, and Erdős-Rényi random graphs, which are defined in the next section.

2.2.3 The Erdős-Rényi Graph

The Erdős-Rényi random graph is named for Paul Erdős and Alfréd Rényi, who studied the model in the 1950s and 1960s (see [11, 12]), although a version was first formulated by Edgar Gilbert in 1959 [14]. The model is defined as follows.

Definition 2.7. *The **Erdős-Rényi random graph** $G(N, q)$ is the graph with nodes $V = \{1, 2, \ldots, N\}$ and edge set E, where each of the $\binom{N}{2}$ possible edges is included in E, independently of other edges, with probability q. The Erdős-Rényi graph is also known as the **Bernoulli random graph**.*

Although we commonly say "*the* Erdős-Rényi graph", we actually mean a graph picked randomly, according to the rule above, from the ensemble of all

possible graphs on N nodes. In particular, if $q = \frac{1}{2}$ then an Erdős-Rényi graph is simply a graph chosen uniformly at random from all graphs with size N.

The simplicity of the definition of the Erdős-Rényi graph allows some of its properties to be calculated easily (Chapter 12, [23]). Edges exist independently and with constant probability, so the total number of edges $e(G) = |E|$ in $G(N, q)$ has distribution Bin $\left(\binom{N}{2}, q\right)$. Therefore, the expected number of edges is $\binom{N}{2}q$.

The degree distribution is calculated in a similar way. For each node in the graph, there are $N - 1$ possible edges to other nodes, each of which exists independently with probability q, so the degree of a node has distribution $\text{Bin}(N - 1, q)$, and the mean degree is $\langle k \rangle = (N - 1)q$.

As $N \to \infty$, with $(N - 1)q$ remaining nearly constant, the binomial distribution with parameters $N - 1$ and q converges to the Poisson distribution with parameter Nq [23]. Therefore, when working with large Erdős-Rényi graphs, we can assume the degree distribution is approximately $\text{Po}(Nq)$. For this reason, the Erdős-Rényi graph is sometimes also called the Poisson random graph.

2.2.4 The Configuration Model

The configuration model gives a method of producing a random graph with a desired degree sequence $(k_1, k_2, \ldots, k_N)$. Each node v_i is assigned k_i "stubs", or half-edges [23]. Pairs of stubs are then joined to create edges between nodes, by choosing two stubs uniformly at random and connecting them, as shown in Fig 2.1.

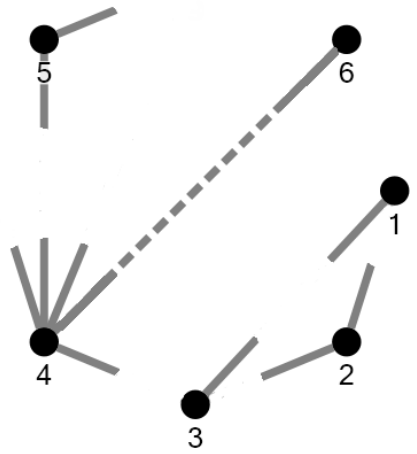

Figure 2.1: Illustration of the formation process for the configuration model. This example generates a random graph with degree sequence (1,2,1,5,2,1). The dotted line shows a pair of edges that has been chosen uniformly at random, and joined to each other. This figure was created using the NodeXL template for Microsoft Excel ([25]).

A slight adjustment to the configuration model allows us to fix a degree distribution, rather than a degree sequence. Instead of directly specifying the degrees of the nodes, we draw a sample of size N from the desired degree distribution, and use this as the degree sequence for the configuration model.[1]

When choosing stubs to combine into edges, it is possible to choose two stubs that originate from the same node. There is also a chance that an edge can be created between two nodes that are already joined by an edge. These self-edges and multiple edges are not included in our definition of a graph (Def. 2.1). However, as the size of the graph increases, the probability of self-edges or multiple edges occurring becomes very small. The results in this report are all for $n \to \infty$, so we will assume that self-edges and multiple edges have little impact on the behaviour of the random graphs in this case.

In this report, when a network is described as an Erdős-Rényi network or a scale-free network we mean that the network is generated by the configuration model, taking the degree sequence from a Poisson distribution, or from a power-law distribution.

2.3 Generating Functions

In the analysis of the robustness of networks, we will often need to manipulate probability distributions, such as the distribution of degrees. An extremely useful tool will be probability generating functions:

Definition 2.8. *The **probability generating function** for a discrete distribution $P = (p_1, p_2, \ldots)$ is the function*

$$G(z) = \sum_k p_k z^k.$$

In many cases, it is easier to work with probability generating functions than with the corresponding probability mass functions. In particular, probability generating functions have the following properties (for proofs, see Chapter 6 in [27]):

- At $z = 1$, we have $G(z) = 1$, provided the distribution P is normalised so that $\sum_k p_k = 1$.
- The probabilities $p_1, p_2, \ldots$ can be recovered by differentiating the probability generating function:

  $$p_k = \frac{1}{k!} G^{(k)}(0).$$

  Consequently, the probability generating function exactly determines the underlying distribution.

[1] The configuration model can deal with any degree sequence, as long as $\sum_{i=1}^{n} k_i$ is not odd. This requirement is because each edge uses two of the stubs, so the total number of stubs must be even. If necessary, we can keep drawing sequences from the degree distribution until we obtain a sequence that meets this condition.

- If X is a discrete random variable with distribution P, then $G(z) = \mathbb{E}[z^X]$. Moreover, $G'(1) = \mathbb{E}[X]$.
- If X and Y are independent random variables with probability generating functions $G_X(z)$ and $G_Y(z)$, then the sum $Z = X + Y$ has probability generating function $G_Z(z) = G_X(z)G_Y(z)$.
- If $(X_i\ ;\ i \geq 1)$ is a collection of independent, identically distributed random variables, and T is an independent, non-negative, discrete random variable, then the sum $Y = \sum_{i=1}^{T} X_i$ has probability generating function

$$G_Y(z) = G_T\left(G_{X_1}(z)\right).$$

Table 2.1 summarises the notation used in this report for various probability generating functions. The generating function for the degree of a node chosen uniformly at random from a network A is denoted $G_A(z)$. The excess degree is obtained by choosing a node u uniformly at random and then following an edge, also chosen uniformly at random, from u to a node v. The excess degree of the node v is the number of edges incident to v, excluding the one edge leading from u. The generating function for this distribution is denoted $\widetilde{G}_A(z)$.

Table 2.1: Notation for probability generating functions.

Notation	Underlying probability distribution
$G_A(z)$	Degree distribution of network A
$\widetilde{G}_A(z)$	Distribution of the excess degree of a node reached by following an edge in A

2.4 The Giant Component

In this report we will consider ensembles of random graphs that share a degree distribution, but we will vary the size N of the graphs, allowing N to tend to infinity. We will see that under certain conditions, the size of the largest connected component of a graph tends to a constant, non-zero fraction of the size of the whole graph. When this occurs, we say that the graph possesses a giant component.

Definition 2.9. *A random graph (ensemble) has a* ***giant component*** *if, for some* $\mu \in (0, 1]$*,*

$$N_C \to \mu N \qquad as \qquad N \to \infty,$$

where N_C *is the size of the largest connected component and* N *is the size of the full graph.*

If a component is not a giant component, then its size, as a fraction of the whole graph, tends to zero as the size of the graph tends to infinity. We call

such components *small components*. If a random graph does not possess a giant component then all components are small components.

In the analysis of robustness of networks, we assume that a node only remains functional if it is in the giant connected component of a network.

2.5 Results for Single-Layer Networks

The first work on robustness of networks was carried out on networks that consist of a single connected component, and that do not interact with any other networks. In this section, we give an account of the main results for these single-layer networks. In later chapters, we will compare our results for interdependent networks with the results described here.

The robustness of scale-free networks was studied by Cohen et al., in the years 2000 and 2001 [10, 9]. In the 2000 paper, Cohen et al. investigated how the connectedness of networks with power-law degree distributions (i.e. $P(k) = Ck^{-\beta}$) is affected when nodes are removed uniformly at random. They found that scale-free networks are in general extremely resilient to the failure of nodes uniformly at random [10]. In particular, they showed that if the parameter β is no larger than 3, and we allow the size of the network to tend to infinity, then the network remains well-connected, no matter what proportion of the nodes fail. This seems reasonable, since in a scale-free network most nodes have low degree, and therefore have little influence on the overall connectivity of the network. When removing nodes uniformly at random, we are likely to pick a node with low degree, rather than one of the fewer high-degree nodes, the removal of which might cause more damage. Callaway et al. derived an equivalent result, also in the year 2000, using probability generating functions [7].

This reasoning is reversed when the network is subject to a targeted attack, where nodes are removed in decreasing order of degree [9]. In this case, the nodes with the largest degrees are removed first, significantly reducing the connectivity of the network. The 2001 paper shows that it is only necessary to remove a small fraction of nodes in this way before the network is broken up into small components [9]. In fact, there is no value of β for which it is necessary to remove more than 3% of the nodes in order to destroy the connectivity of the network (p613, [23]). For both the case of failure uniformly at random, and the case of targeted attack in decreasing order of degree, the results given by Cohen et al. are in agreement with the simulations presented by Albert et al. in 2000 [1].

The generating function method in [7] applies equally to the study of the robustness of Erdős-Rényi networks. Newman also considers Erdős-Rényi networks as an example in Chapter 16 of [23]. He shows that if nodes are removed uniformly at random from an Erdős-Rényi network A, with mean degree $\langle k \rangle = a$, then we should expect a giant component to remain, as long as no more than a fraction $1 - \frac{1}{a}$ of the nodes are removed. For example, if the mean degree of the network is 3, we would expect a giant component to survive until the point when two thirds of the nodes have been removed. Consequently, Erdős-Rényi networks with sufficiently large mean degrees can be very resilient against the failure uniformly at random of nodes.

3. Random Node Failure

3.1 The Cascade Process

We study a model consisting of a pair of interdependent networks, as discussed by Buldyrev et al. in 2010 [6]. In the model, we consider two networks A and B with degree distributions $P_A(k)$ and $P_B(k)$ and equal sizes $|A| = |B| = N$. We say that a node i_A in A *depends* on a node j_B in B if the failure of j_B causes the failure of i_A. In this model, we assume that each node i_A of A depends only on one node i_B of B, which in turn depends only on i_A. If two networks are related in this way, we will call them *coupled networks*.

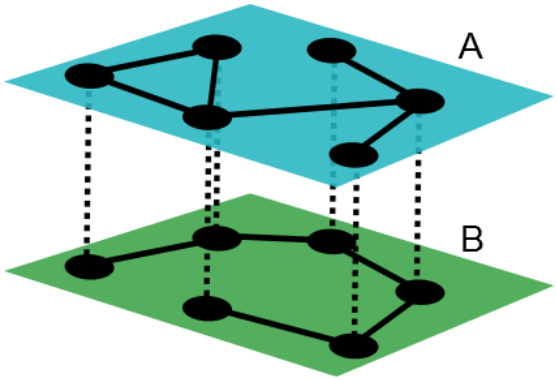

Figure 3.1: An example of coupled networks. Edges within networks are represented by solid lines, and dependencies between networks are represented by dotted lines. A-edges join nodes in A and B-edges join nodes in B. Each node in A depends on, and is depended upon by, exactly one node in B. This figure is adapted, with permission, from Figure 7 in [19], using Inkscape ([17]).

In single-layer networks, connected components are used to measure how well a network is connected. For the coupled networks described above, we generalise the concept of connected components to mutually connected clusters, which are defined as follows.

Definition 3.1. *A set of nodes $S_A \subseteq A$ with corresponding set $S_B \subseteq B$ is a* ***mutually connected set*** *if any two nodes in S_A are connected by a path of A-edges in S_A and any two nodes in S_B are connected by a path of B-edges in S_B.*

A mutually connected set is a ***mutually connected cluster*** *(MCC) if it is impossible to add another node to the set to form a mutually connected set.*

Mutually connected clusters serve the same role in coupled networks as connected components serve in single-layer networks. We can define the giant mutually connected cluster, or *giant cluster*, in the same way as we defined the giant component (Def. 2.9). *Small clusters* are also defined, analogously to small components. To remain functional in a pair of coupled networks, a node must be part of the giant cluster.

Following the argument in [6], we study how this model reacts to the failure of a given fraction of nodes, picked uniformly at random from one of the networks. First, choose a fraction $1 - p$ of the nodes of network A, uniformly at random. Remove these nodes from the network and delete all edges that are incident to the removed nodes. This creates a network A_0 of size $N_0 = pN$. Carry out the following cascade process on A_0 and B (as illustrated in Fig. 3.2):

1. Define the a_1-clusters as the connected components of network A_0 (a_{11}, a_{12} and a_{13} in panel (b) of Fig. 3.2), and the b_1-sets as the sets of B-nodes corresponding to the a_1-clusters.

2. Remove any B-edges that connect two different b_1-sets. Define the b_2-clusters as the connected components of the modified B. In Fig. 3.2, these are b_{21}, b_{22}, b_{23} and b_{24}, in panel (c). Note that any b_1-sets that are also b_2-clusters are mutually connected clusters.

3. Repeat steps 1 and 2 to find a_2-sets, a_3-clusters, b_3-sets, b_4-clusters, and so on.

4. Stop when no more edges are removed (panel (d) in Fig. 3.2). When this occurs, the a_n-clusters will be the same as the b_{n+1}-clusters, which will be the same as the a_{n+2}-clusters, for some $n \in \mathbb{N}$.

We will call the largest mutually connected cluster after step 4 the *surviving cluster*. This is the largest part of the coupled system that remains functional after the cascade of failures. We investigate which values of p allow the survival of a giant mutually connected cluster, and which result in highly fragmented networks which consist entirely of small clusters. In addition, we are interested in how the system transitions from one state to the other. As explained in Chapter 16 in ref. [23], when $p = 1$ there are no node failures, and there is a giant cluster of size N. On the other hand, for small values of p there is no giant cluster. We use the notation p_c for the threshold value of p: the point at which a giant cluster first appears. For $p < p_c$, there is no surviving giant cluster; for $p > p_c$, there is a giant cluster.

3.2 The Generating Function Method

We use probability generating functions to investigate the size of the largest clusters of the networks at each stage of the process. Let $G_A(z)$ be the generating function for $P_A(k)$ and $G_B(z)$ be the generating function for $P_B(k)$. At the nth

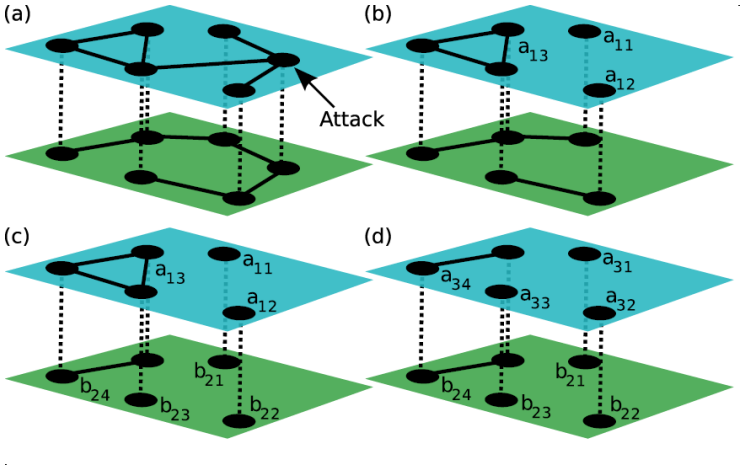

Figure 3.2: The cascade process on coupled networks. Panel (a) shows the initial failure of a node, and the removal of the edges connected to the node. Panel (b) shows the a_1-clusters, as described in step 1. Panel (c) shows the state of the coupled networks after step 2, and panel (d) shows the state of the networks when the cascade process terminates (step 4). This figure is adapted, with permission, from Fig. 7 in [19].

stage of the process, for $n \in \{1, 2, \ldots\}$, let μ_n be the fraction of the nodes of the original network that are included in the largest a_n- or b_n-cluster. It follows that $\mu_\infty = \lim_{n\to\infty} \mu_n$ is the probability that a node chosen uniformly at random from the original network belongs to the surviving cluster. We use the notation A_n and B_n, for $n \in \{1, 2 \ldots\}$, to denote the largest components of the modified networks A and B at the nth stage. For any network X, we write $d_X(v)$ to mean the degree of a (usually randomly chosen) node v in X.

In the first stage of the process, we remove a fraction $1 - p$ of the nodes uniformly at random from A, and this alters the degree distribution of the remaining nodes. Let $G_{A_0}(z)$ be the probability generating function for the degree distribution of network A_0. The function $G_{A_0}(z)$ can then be written in terms of $G_A(z)$ as follows:

$$\begin{aligned}
G_{A_0}(z) &= \sum_{m=0}^{\infty} z^m \Pr(d_{A_0}(v) = m) \\
&= \sum_{m=0}^{\infty} \sum_{k=m}^{\infty} z^m \Pr(d_{A_0}(v) = m \mid d_A(v) = k) \times \Pr(d_A(v) = k) \\
&= \sum_{k=0}^{\infty} \sum_{m=0}^{k} z^m \binom{k}{m} p^m (1-p)^{k-m} P_A(k) \\
&= \sum_{k=0}^{\infty} P_A(k)(1 - p + zp)^k \\
G_{A_0}(z) &= G_A(1 - p(1 - z)). \qquad (3.1)
\end{aligned}$$

In order to study the size of the giant cluster, if one exists, we first consider the sizes of the small clusters. Choose a node v uniformly at random from A_0. Let the distribution of the size of the cluster reachable from v be $P_{\text{clust}} = (q_1, q_2, \ldots)$, if the cluster is a small cluster. We write $H_{A_0}(z)$ for the probability generating function of P_{clust}. Suppose that we choose an edge e of v, uniformly at random. Let $P^{\text{ex}}_{\text{clust}} = (q^{\text{ex}}_1, q^{\text{ex}}_2, \ldots)$ be the probability distribution of the size of the cluster reached by following the edge e, and let $P^{\text{ex}}_{\text{clust}}$ have generating function $\widetilde{H}_{A_0}(z)$. If there is no giant component of A_0, then all the clusters are small clusters, so $\widetilde{H}_{A_0}(1) = \sum_{s=1}^{\infty} q^{\text{ex}}_s = 1$. If there is a giant component, then $\widetilde{H}_{A_0}(1) < 1$.

Let $\widetilde{G}_{A_0}(z)$ be the probability generating function for the excess degree distribution $P^{\text{ex}} = (p^{\text{ex}}_1, p^{\text{ex}}_2, \ldots)$ of A_0. When choosing the random edge e, leading from the random vertex v, the probability of ending up at any given node is proportional to the degree of that node. This allows us to write $\widetilde{G}_{A_0}(z)$ in terms of $G_{A_0}(z)$, as in [7]:

$$\begin{aligned}
\widetilde{G}_{A_0}(z) &= \sum_{k=0}^{\infty} z^k p^{\text{ex}}_k \\
&\propto \sum_{k=0}^{\infty} z^k (k+1) \Pr(d_{A_0}(v) = k+1) \\
&= G'_{A_0}(z).
\end{aligned}$$

Normalising gives

$$\widetilde{G}_{A_0}(z) = \frac{G'_{A_0}(z)}{G'_{A_0}(1)}.$$

Additionally, the same reasoning that leads to Eq. (3.1) gives

$$\widetilde{G}_{A_0}(z) = \widetilde{G}_A(1 - p(1-z)). \tag{3.2}$$

We now derive expressions for $H_{A_0}(z)$ and $\widetilde{H}_{A_0}(z)$. Suppose that an edge e, chosen uniformly at random, leads to a node with excess degree k. Each of the k additional edges leads to a cluster, and the sizes of these clusters are independently distributed,[1] with distribution generated by $\widetilde{H}_{A_0}(z)$ (see Fig. 3.3). Therefore, the total size of the cluster that can be reached from e has probability generating function $[\widetilde{H}_{A_0}(z)]^k$. Consequently,

$$\begin{aligned}
\widetilde{H}_{A_0}(z) &= z \sum_{k=0}^{\infty} p^{\text{ex}}_k [\widetilde{H}_{A_0}(z)]^k \\
&= z\widetilde{G}_{A_0}\left(\widetilde{H}_{A_0}(z)\right) \\
&= z\widetilde{G}_A\left(1 - p\left[1 - \widetilde{H}_{A_0}(z)\right]\right).
\end{aligned}$$

[1] Actually, the sizes of the clusters are only independent if the network is *locally tree-like*, which means that the network does not contain many small loops. Networks that are generated by the configuration model, with sufficiently few edges, satisfy this condition [24], so our approach is appropriate in this case. It should be noted, however, that some results derived using generating function methods have been shown to hold, even in networks that contain many small loops (for more information, see ref. [21]).

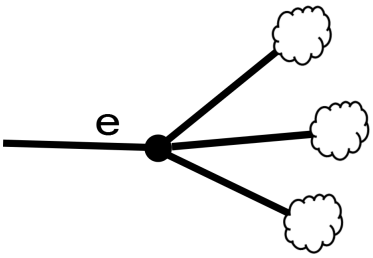

Figure 3.3: The cluster reachable by following a random edge e. In this example, e has excess degree 3. The bubbles represent the clusters reached by following each of the three edges.

By choosing a node v uniformly at random from A_0, and conditioning on the degree of v, we derive a similar result for $H_{A_0}(z)$. Let $c_{A_0}(v)$ denote the size of the cluster containing v. We then calculate

$$\begin{aligned}
H_{A_0}(z) &= z \sum_{k=0}^{\infty} z^k \Pr(c_{A_0}(v) = k+1) \\
&= z \sum_{k=0}^{\infty} z^k \sum_{m=0}^{\infty} \Pr(c_{A_0}(v) = k+1 \mid d_{A_0}(v) = m) \times \Pr(d_{A_0}(v) = m) \\
&= z \sum_{m=0}^{\infty} \Pr(d_{A_0}(v) = m) \sum_{k=0}^{\infty} z^k \Pr(c_{A_0}(v) = k+1 \mid d_{A_0}(v) = m) \\
&= z \sum_{m=0}^{\infty} \Pr(d_{A_0}(v)) \left[\widetilde{H}_{A_0}(z)\right]^m \\
&= z G_{A_0}(\widetilde{H}_{A_0}(z)) \\
H_{A_0}(z) &= z G_A \left(1 - p\left[1 - \widetilde{H}_{A_0}(z)\right]\right).
\end{aligned}$$

3.3 The Size of the Surviving Cluster

The function $H_{A_0}(z)$ only generates the distribution of the sizes of the small clusters. If a giant cluster exists, let $g_A(p)$ be the fraction of the nodes of A_0 that are included in the giant cluster. It follows that the fraction of the nodes of A_0 that are not in the giant cluster is given by

$$\begin{aligned}
1 - g_A(p) &= \sum_{k=1}^{\infty} \Pr(c_{A_0}(v) = k) \\
&= H_{A_0}(1).
\end{aligned}$$

Using the expression for $H_{A_0}(z)$ that we derived earlier, we can write

$$\begin{aligned}
g_A(p) &= 1 - H_{A_0}(1) \\
&= 1 - G_A\left(1 - p\left[1 - \widetilde{H}_{A_0}(1)\right]\right) \\
g_A(p) &= 1 - G_A(1 - p\,[1 - f_A]), \qquad (3.3)
\end{aligned}$$

where $f_A = \widetilde{H}_{A_0}(1)$ satisfies

$$f_A = \widetilde{G}_A(1 - p\,[1 - f_A]). \tag{3.4}$$

Similarly, defining $\widetilde{G}_B(z)$ and $g_B(p)$ in the obvious way yields

$$g_B(p) = 1 - G_B(1 - p\,[1 - f_B]), \tag{3.5}$$

where $f_B = \widetilde{H}_{B_0}(1)$ satisfies

$$f_B = \widetilde{G}_B(1 - p\,[1 - f_B]). \tag{3.6}$$

The largest component of A_0 is denoted A_1, and A_1 has size

$$N_1 = N_0 g_A(p) = p g_A(p) N = \mu_1 N.$$

The second stage of the cascade process takes place on the network B. Any nodes in B that depend on nodes not in A_1 become disconnected from the largest cluster. This amounts to the removal uniformly at random of a fraction μ_1 of the nodes from B. Therefore, by the method explained above, the resulting largest component B_2 has size

$$N_2 = N_1 g_B(\mu_1) = p\, g_A(p)\, g_B(\mu_1) N = \mu_2 N.$$

This gives the relation $\mu_2 = p\, g_A(p)\, g_B(\mu_1)$. We derive similar relations, so that each of the fractions μ_n, for $n \in \{2, 3, \ldots\}$, can be calculated recursively. To find a relation for each μ_n, we look for a fraction μ'_{n-1} such that the nth stage of the cascade process is equivalent to a random attack on the original network, with survival probability μ'_{n-1}. For example, we have already seen that $\mu'_0 = \mu_0 = p$, and $\mu'_1 = \mu_1 = p g_A(p) = \mu'_0 g_A(\mu'_0)$. However, $\mu'_n \neq \mu_n$ for n in general.

At stage $2m$ of the cascade, we remove the $1 - g_A(\mu'_{2m})$ fraction of nodes that are in the largest component B_{2m} but not in $A_{2m+1} \subseteq B_{2m}$. This has the same effect as removing the same fraction of nodes from B_0 uniformly at random, so we can calculate the number of remaining nodes at the $(2m+1)$-stage by randomly removing

$$\begin{aligned}[1 - g_A(\mu'_{2m})]\, N_0 + (1-p)N &= [1 - g_A(\mu'_{2m})]\, pN + (1-p)N \\ &= [1 - p g_A(\mu'_{2m})]\, N\end{aligned}$$

nodes from the original network B. This is equivalent to the first step of the process, except that p is replaced by the fraction μ'_{2m+1}, where

$$\mu'_{2m+1} = p g_A(\mu'_{2m}). \tag{3.7}$$

We can now calculate μ_{2m+1} in the same way as we found μ_1, to get

$$\mu_{2m+1} = \mu'_{2m} g_A(\mu'_{2m}). \tag{3.8}$$

Reversing the roles of A and B and considering stage $2m-1$ of the cascade yields

$$\mu'_{2m} = pg_B(\mu'_{2m-1}), \tag{3.9}$$
$$\mu_{2m} = \mu'_{2m-1} g_B(\mu'_{2m-1}). \tag{3.10}$$

We use Eqs. (3.7–3.10) to calculate $\mu_\infty = \lim_{n\to\infty} \mu_n$. We know that μ_∞ must satisfy $\mu_{2m+1} = \mu_{2m} = \mu_{2m-1}$, and it is also clear that the equality $\mu'_m = \mu'_{m+2}$ must hold as $m \to \infty$. Therefore, we can write $x = \mu'_{2m+1} = \mu'_{2m-1}$ and $y = \mu'_{2m} = \mu'_{2m-2}$, with the understanding that m is large. Therefore $\mu_\infty = xg_B(x) = yg_A(y)$, and Eqs. (3.7) and (3.9) become

$$\begin{cases} x = pg_A(y), \\ y = pg_B(x). \end{cases} \tag{3.11}$$

Recall that Eqs. (3.3–3.6) express g_A and g_B in terms of G_A, $\widetilde{G}_A$, G_B, and $\widetilde{G}_B$. When the degree distributions of A and B are known, the system (3.11) can therefore be solved numerically, and we can obtain μ_∞ as a function of p.

Numerical results give an idea of how the coupled networks behave, but for some networks it is possible to obtain analytic results concerning the values of p that lead to highly fragmented networks, those that allow the survival of a large mutually connected cluster, and the transition between the two states.

3.4 Example: Uniform Random Failure in Erdős-Rényi Networks

If networks A and B are Erdős-Rényi networks, then we can simplify many of the expressions in sections 3.2 and 3.3. In an Erdős-Rényi network, the node degrees are approximately Poisson-distributed, for large N. Therefore, the probability generating functions are

$$G_A(z) = \exp\left[a(z-1)\right],$$

where $a = \langle k \rangle_A$ is the mean node degree of A, and

$$\widetilde{G}_A(z) = \frac{G'_A(z)}{G'_A(1)} = \exp\left[a(z-1)\right] = G_A(z).$$

Similarly,

$$G_B(z) = \widetilde{G}_B(z) = \exp\left[b(z-1)\right],$$

where $b = \langle k \rangle_B$ is the mean node degree of B.
The functions g_A and g_B can then be written as

$$\begin{aligned} g_A(y) &= 1 - G_{A0}(1 - y(1 - f_A)) \\ &= 1 - \exp\left[ay(f_A - 1)\right], \end{aligned}$$

where $f_A = \exp\left[ay(f_A - 1)\right]$, and

$$g_B(x) = 1 - \exp\left[bx(f_B - 1)\right],$$

where $f_B = \exp\left[bx(f_B - 1)\right]$.
Accordingly, the system (3.11) becomes

$$\begin{cases} x = p(1 - f_A), \\ y = p(1 - f_B), \end{cases} \tag{3.12}$$

where f_A and f_B satisfy

$$\begin{cases} f_A = \exp\left[ay(f_A - 1)\right], \\ f_B = \exp\left[bx(f_B - 1)\right]. \end{cases}$$

Eliminating x and y using the equations in (3.12) gives

$$\begin{cases} f_A = \exp\left[-ap(1 - f_A)(1 - f_B)\right], \\ f_B = \exp\left[-bp(1 - f_A)(1 - f_B)\right]. \end{cases} \tag{3.13}$$

Note that ${f_A}^{\frac{1}{a}} = \exp\left[-p(1 - f_A)(1 - f_B)\right] = {f_B}^{\frac{1}{b}}$. Introducing the variable $r = {f_A}^{\frac{1}{a}} = {f_B}^{\frac{1}{b}}$, we write

$$r = \exp\left[-p(r^a - 1)(r^b - 1)\right], \tag{3.14}$$

and this equation can be solved graphically.

Solutions for r occur at the intersections of the line $y = r$ with the curve $y = \exp\left[-p(r^a - 1)(r^b - 1)\right]$. For any value of p, there is a trivial solution at $r = 1$. This might be the only possible solution, but as p varies, the trivial solution might be accompanied by one or two alternative solutions, as can be seen in Fig. 3.4. The figure shows that for $p = 0.3$, the curve $y = \exp\left[-p(r^a - 1)(r^b - 1)\right]$ and the line $y = r$ intersect only at $r = 1$. However, as the value of p increases, the curve crosses the line, producing multiple additional solutions to Eq. (3.14). The critical value p_c of p is the value at which a non-trivial solution is first possible. This occurs when the curve and the line are tangent to each other (which happens at about $p = 0.409$ in the example in Fig. 3.4). Therefore, at p_c, the line and the curve have equal derivatives, so

$$\begin{aligned} 1 &= \frac{\mathrm{d}}{\mathrm{d}r}\left[e^{-p(r^a-1)(r^b-1)}\right] \\ &= -p\left[ar^{a-1}(r^b - 1) + br^{b-1}(r^a - 1)\right] e^{-p(r^a-1)(r^b-1)} \\ &= -p\left[ar^a(r^b - 1) + br^b(r^a - 1)\right] \\ &= -p\left[(a + b)r^{a+b} - ar^a - br^b\right]. \end{aligned}$$

In combination with Eq. (3.14), this gives the following simultaneous equations:

$$\begin{cases} r &= e^{-p(r^a-1)(r^b-1)}, \\ 1 &= p\left[ar^a + br^b - (a + b)r^{a+b}\right]. \end{cases} \tag{3.15}$$

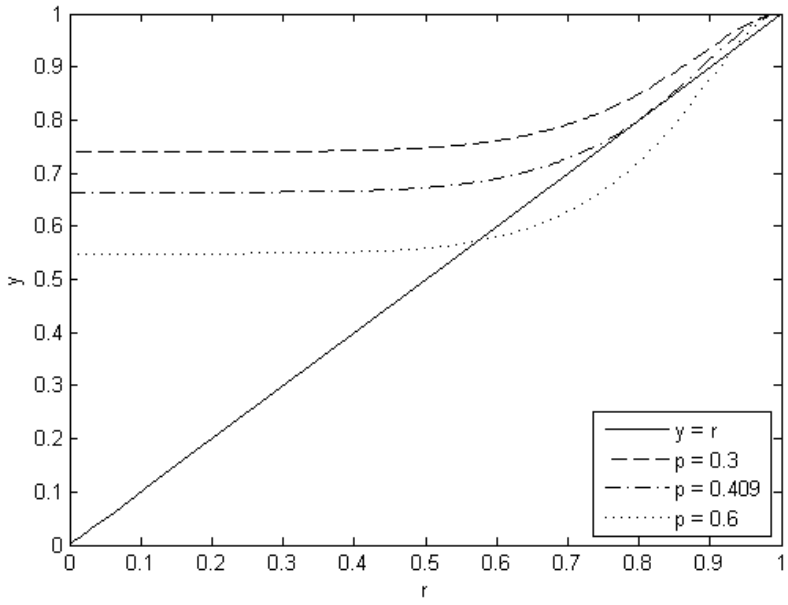

Figure 3.4: A graphical solution to Eq. (3.14), for coupled Erdős-Rényi networks with equal mean degrees ($a = b = 6$). We plot the curve $y = \exp\left[-p(r^a - 1)^2\right]$ for $p = 0.3$, 0.409, and 0.6; and we consider the points of intersection with the line $y = r$.

The second equation in (3.15) shows that the critical point is

$$p_c = \left[a r_c^a + b r_c^b - (a+b) r_c^{a+b}\right]^{-1} ,$$

so r_c satisfies

$$r_c = \exp\left[-\frac{(r_c^a - 1)(r_c^b - 1)}{a r_c^a + b r_c^b - (a+b) r_c^{a+b}}\right] .$$

Using the equations in (3.12), the fraction μ_∞ of the nodes of A that are included in the surviving cluster can be expressed as

$$\mu_\infty = x g_B(x) = y g_A(y) = p(1 - r_c^a)(1 - r_c^b).$$

We now make a further simplification and suppose that the two Erdős-Rényi networks A and B have the same mean node degree, $a = b$, so that $f = f_A = f_B$. The value f_c of f at the critical point then satisfies

$$\begin{aligned} f_c &= \exp\left[a y (f_c - 1)\right] \\ &= \exp\left[a p_c (1 - r_c^b)(f_c - 1)\right] \\ &= \exp\left[-a p_c (f_c - 1)^2\right] . \end{aligned}$$

When $a = b$, note that

$$\begin{aligned} p_c &= (2a f_c - 2a f_c^2)^{-1} \\ &= (2a f_c)^{-1} (1 - f_c)^{-1}, \end{aligned}$$

so the expression for f_c can be simplified to obtain

$$\begin{aligned} f_c &= \exp\left(\frac{-a(f_c-1)^2}{-2af_c(f_c-1)}\right) \\ &= \exp\left(\frac{f_c-1}{2f_c}\right). \end{aligned} \tag{3.16}$$

Equation (3.16) can be solved numerically (see Appendix A), to give critical values of $f_c \approx 0.28467$, and $p_c \approx \frac{1}{a}(2.4554)$. The corresponding expected size of the surviving cluster, as a fraction of the size of the original graph, is $\mu_\infty \approx \frac{1}{a}(1.2564)$.

3.5 Simulations

3.5.1 Erdős-Rényi Networks

I used MATLAB to check the results derived above against simulations of pairs of interdependent Erdős-Rényi networks. I wrote functions (shown in Appendix A) to replicate the cascade process on coupled Erdős-Rényi networks, with given size and mean degree, for different values of p (recall that $1-p$ is the fraction of nodes that are removed in the initial failure). Figures 3.5 and 3.6 were produced by considering networks with sizes $N = 500, 1000, 2000$, and 5000, all with mean degree $\langle k \rangle = 6$.

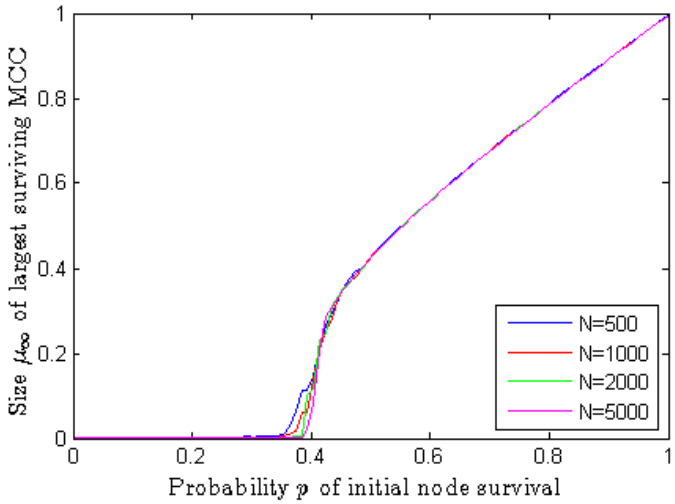

Figure 3.5: Simulations of the cascade process for coupled Erdős-Rényi networks with equal mean degree, $a = b = 6$. The size of the largest surviving mutually connected cluster is given as a fraction μ_∞ of the size of the full network. The curves plotted are averages, taken over 50 simulations.

First, we note that for each of the values of N that we considered, the curve is close to zero when p is small. The surviving cluster is very small in comparison

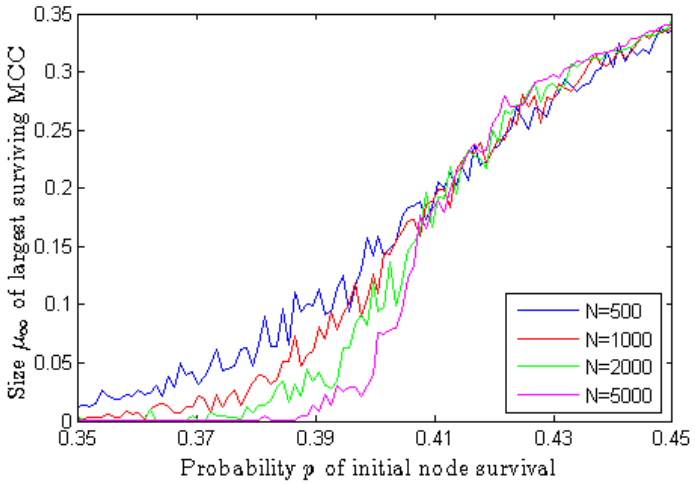

Figure 3.6: A more detailed version of Fig. 3.5. The curves for $N = 500$ and $N = 1000$ are averages over 120 simulations; the curve for $N = 2000$ is an average over 70 simulations, and the curve for $N = 5000$ is an average over 50 simulations.

to the size of the original network, because the cascade of failures has reduced most clusters to one or two nodes.

There is a sudden increase at about $p = 0.4$, after which the size of the surviving cluster is approximately the same as the number of nodes that survive the initial failure. If only a small fraction of nodes are removed in the initial failure, the network is unlikely to become disconnected, so the cascade process does not cause the failure of any additional nodes, beyond the first group of nodes that failed.

To investigate the existence of a giant cluster, we compare the curves as the size N of the network increases. As N increases, the parts of the curves that approach zero correspond to the values of p for which there is no giant cluster. The parts of the curves that tend to a non-zero value correspond to values of p for which there is a giant cluster.

In section 3.4, we showed that for coupled Erdős-Rényi networks with equal mean degree $\langle k \rangle = a$, we expect a giant cluster to exist whenever $p > \frac{1}{a}(2.4554)$. When $a = 6$, this suggests a critical value of $p_c \approx 0.409$, which matches well with the simulations shown in Fig. 3.5. Figure 3.6 gives a more detailed look at the behaviour of the curves in this area. As it is only possible to simulate finite networks, we do not get the immediate transition at $p = 0.409$ that section 3.4 predicts. Instead, there is some rounding of the curves, which is more noticeable for the smaller networks. However, even with the limited simulations performed here, we can see how the transition becomes more sudden as the size of the networks increases. Buldyrev et al. carry out simulations for much larger values of N, and their results confirm this trend (Fig. 3 in [6]).

3.5.2 Scale-Free Networks

I also ran simulations of the cascade process for coupled scale-free networks (MATLAB code in Appendix A), and the results are shown in Fig. 3.7. The curves in Fig. 3.7 are not as smooth as the curves in Fig. 3.5, as the averages were taken over fewer simulations. If more time was available, the curves could be made smoother by averaging over more simulations.

In contrast to the simulations for Erdős-Rényi networks, there is no sudden transition from very small surviving clusters to surviving clusters that occupy a more significant fraction of the network. There is a region, for small p (e.g. $p < 0.4$), where the network is entirely broken into small clusters, but for $p > 0.4$, the size of the surviving cluster appears to grow almost linearly with p, from approximately 0 to 1.

Single-layer scale-free networks are very resilient against the removal uniformly at random of nodes [10]. It is necessary to remove a very large fraction of the nodes in order to destroy the giant component. In Fig. 3.7, we see that this is not the case for the coupled scale-free networks in our simulations. In every simulation, removing 70% of the nodes resulted in the complete destruction of the network into small clusters. This is a much smaller fraction of nodes than the fraction required for single-layer scale-free networks [1]. Therefore, when dealing with a scale-free network it is crucial to take account of any dependency the network has on other networks. If not, a coupled network might be mistakenly treated as a single-layer network, and expected to be more robust than it is.

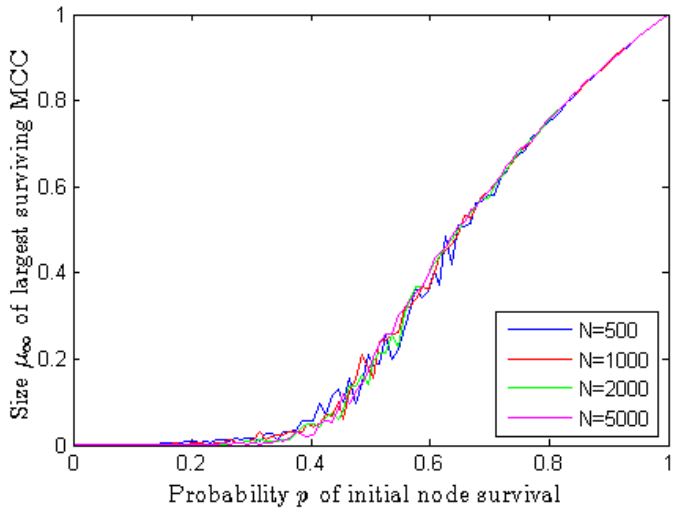

Figure 3.7: Simulations of the cascade process for coupled scale-free networks. For each value of N, the curves are averages over 20 simulations.

4. Targeting by Node Degree

4.1 The Targeted Attack Problem

In the previous chapter, we analysed how a pair of interdependent networks behaved following an initial failure of nodes chosen uniformly at random. In some situations, a constant probability of failure is a reasonable assumption to make. However, it does not allow cases where failure is related in some way to the 'importance' of a node. For example, attackers targeting the World Wide Web might choose to focus their attacks on the websites that have the most connections to other websites. In this chapter, we will consider how the cascade of failures is affected when the probability of initial failure of a node is determined by the degree of the node.

The approach in this section is closely based on the approach described in [16]. We aim to reduce the problem to an equivalent problem where the initial nodes are removed uniformly at random. We can then make use of the results that we have presented in Chapter 3.

To begin, define a family of probability distributions

$$W_\alpha(k_i) = \frac{{k_i}^\alpha}{\sum_{i=1}^{N} {k_i}^\alpha}, \qquad \alpha \in \mathbb{R}, \tag{4.1}$$

where $W_\alpha(k_i)$ is the probability that the node i, which has degree k_i, is removed in the initial attack.

Varying the parameter α allows us to specify the nature of the initial attack. For $\alpha > 0$, nodes with higher degree are more likely to be attacked; for $\alpha < 0$, nodes with lower degree are more vulnerable (e.g., if nodes with high degree are better protected). In the $\alpha = 0$ case, nodes fail independently at random with constant probability $W_0(k_i) = \frac{1}{N}$. If we take $\alpha \to \infty$, then nodes are removed in exact order of decreasing degree. When $\alpha < 0$, it is necessary to first remove all nodes of degree 0 from the network. As these nodes cannot form part of the giant cluster, this will not affect our results.

As in Chapter 3, we consider two networks A and B with equal size $|A| = |B| = N$, where every node is in one-to-one dependency with a node from the other network. We follow the same cascade process, except that we remove $(1-p)N$ nodes according to the probabilities $W_\alpha(k_i)$ during the initial attack, rather than removing nodes uniformly at random.

We seek a network A' such that the targeted attack problem on A and B is equivalent to the problem with failure uniformly at random on networks A' and B.

4.2 Reducing to the Uniform Random Failure Problem

First, it is necessary to find the degree distribution of the nodes that survive the initial attack on A. We consider the network after the initial removal of $(1-p)N$ nodes but before the removal of any newly disconnected edges. We call this network $\bar{A}_0$. The degree of any surviving node remains unchanged in $\bar{A}_0$, as illustrated in Fig. 4.1.

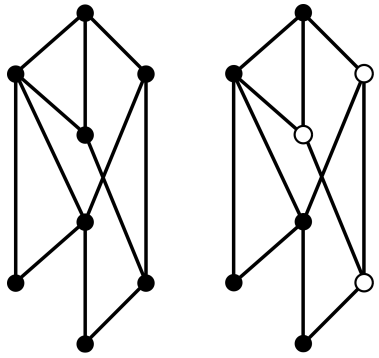

Figure 4.1: The formation of $\bar{A}_0$ from A. The first diagram shows a network A, and the second shows A_0, where nodes that are removed are represented by empty circles.

Let $P_p(k)$, for $k \in \{0, 1, \ldots\}$, be the degree distribution of the nodes in $\bar{A}_0$, and let $A_p(k)$ be the number of nodes in $\bar{A}_0$ with degree k. As $\bar{A}_0$ has size pN, it follows that

$$P_p(k) = \frac{A_p(k)}{pN}. \tag{4.2}$$

We now remove one additional node, so that $pN - 1 = N(p - \frac{1}{N})$ nodes remain in the network. The probability that the node chosen has degree k is proportional to $P_p(k)k^\alpha$, where $k^\alpha = k^\alpha(p)$, which we normalise to obtain

$$\frac{P_p(k)k^\alpha}{\langle k^\alpha \rangle} = \frac{P_p(k)k^\alpha}{\sum_j P_p(j)j^\alpha}.$$

Therefore, the expected number of nodes with degree k in the modified network is

$$A_{(p-\frac{1}{N})}(k) = A_p(k) - \frac{P_p(k)}{\langle k^\alpha \rangle}, \tag{4.3}$$

which we rearrange to give

$$\frac{A_p(k) - A_{(p-\frac{1}{N})}}{\frac{1}{N}} = \frac{NP_p(k)k^\alpha}{\langle k^\alpha \rangle}. \tag{4.4}$$

By taking the limit $N \to \infty$, Eq. (4.4) yields

$$\frac{\mathrm{d}}{\mathrm{d}p}\left[A_p(k)\right] = \frac{NP_p(k)k^\alpha}{\langle k^\alpha \rangle}. \tag{4.5}$$

A second expression for $\frac{\mathrm{d}}{\mathrm{d}p}\left[A_p(k)\right]$ is obtained by differentiating Eq. (4.2) with respect to p:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}p}\left[A_p(k)\right] &= \frac{\mathrm{d}}{\mathrm{d}p}\left[pNP_p(k)\right] \\ &= NP_p(k) + pN\frac{\mathrm{d}}{\mathrm{d}p}\left[P_p(k)\right].\end{aligned} \tag{4.6}$$

Setting the two expressions (4.5) and (4.6) equal to each other gives

$$-p\frac{\mathrm{d}}{\mathrm{d}p}P_p(k) = P_p(k) - \frac{P_p(k)k^\alpha}{\langle k^\alpha \rangle}. \tag{4.7}$$

To solve Eq. (4.7), we define $G_\alpha(x) = \sum_k P(k)x^{k^\alpha}$, where $P(k) = P_A(k)$, and set $h = G_\alpha^{-1}(p)$. Therefore, $G_\alpha(h(p)) = p$, and differentiating both sides of (4.7) yields

$$h'(p) = \frac{1}{{G_\alpha}'(h)}.$$

Taking $P_p(k) = P(k)\frac{h^{k^\alpha}}{G_\alpha(h)} = \frac{1}{p}P(k)h^{k^\alpha}$, with $\langle k^\alpha \rangle = h\frac{{G_\alpha}'(h)}{G_\alpha(h)}$, gives a solution to Eq. (4.7), as we can see with the following calculation:

$$\begin{aligned}-p\frac{\mathrm{d}}{\mathrm{d}p}P_p(k) &= -p\frac{\mathrm{d}}{\mathrm{d}p}\left[\frac{1}{p}P(k)h^{k^\alpha}\right] \\ &= -p\left[-\frac{1}{p^2}P(k)h^{k^\alpha} + \frac{1}{p}P(k)k^\alpha h^{k^\alpha - 1}h'(p)\right] \\ &= \frac{1}{p}P(k)h^{k^\alpha} - \frac{1}{p}P(k)h^{k^\alpha}\frac{k^\alpha p}{h{G_\alpha}'(h)} \\ &= P_p(k) - \frac{P_p(k)k^\alpha}{\langle k^\alpha \rangle}.\end{aligned}$$

Therefore, the degree distribution of the nodes in $\bar{A}_0$ is generated by

$$G_{\bar{A}_0}(x) = \sum_k P_p(k)x^k = \frac{1}{p}\sum_k P(k)h^{k^\alpha}x^k. \tag{4.8}$$

We will now calculate the degree distribution of the nodes in the network A_0, where A_0 is formed from $\bar{A}_0$ by removing any edge that is only connected at one of its endpoints. We write $P_0(k)$ for this degree distribution.

Consider the set of edges from a node v in $\bar{A}_0$. When we form the network A_0, the edges that are removed are the edges that do not lead to another node in $\bar{A}_0$. Therefore, on average we keep a fraction $\tilde{p}$ of the edges from each vertex, where $\tilde{p}$ is the expected fraction of the original edges of A that have at least one endpoint in $\bar{A}_0$.
The value of $\tilde{p}$ is

$$\tilde{p} = \frac{|E(\bar{A}_0)|}{|E(A)|} = \frac{pN\langle k(p)\rangle}{N\langle k\rangle} = \frac{\sum_k P(k)kh^{k^\alpha}}{\sum_k P(k)k},$$

where $\langle k\rangle$ is the expected degree of a node in A, and $\langle k(p)\rangle$ is the expected degree of a node in A_0. It is now possible to find the generating function $G_{A_0}(x)$ for $P_0(k)$, by conditioning on the degree of a node in $\bar{A}_0$. We obtain

$$\begin{aligned}
G_{A_0}(x) &= \sum_{k=0}^{\infty} P_0(k)x^k \\
&= \sum_{k=0}^{\infty}\sum_{m=k}^{\infty} \Pr(d_{A_0}(v) = k \mid d_{\bar{A}_0}(v) = m)\Pr(d_{\bar{A}_0}(v) = m)x^k \\
&= \sum_{m=0}^{\infty} P_p(m)\sum_{k=0}^{m}\binom{m}{k}\tilde{p}^k(1-\tilde{p})^{m-k}x^k \\
&= \sum_{m=0}^{\infty} P_p(m)(1-\tilde{p}+\tilde{p}x)^m \\
G_{A_0}(x) &= G_{\bar{A}_0}(1-\tilde{p}+\tilde{p}x). \qquad (4.9)
\end{aligned}$$

Recall from Eq. (3.1) that for a network X, the degree distribution after the removal — uniformly at random — of a fraction $1-p$ of the nodes is generated by $G_X(1-p+px)$. Therefore, we seek a network A' with degree distribution generated by $G_{A'}(x)$, such that

$$G_{A'}(1-p+px) = G_{\bar{A}_0}(1-\tilde{p}+\tilde{p}x). \qquad (4.10)$$

Suppose that $G_{A'}(z)$ is of the form $G_{A'}(z) = G_{\bar{A}_0}(f(z))$. If we can find a function $f(z)$ that satisfies $f(1-p+px) = 1-\tilde{p}+\tilde{p}x$, then $G_{A'}(z)$ satisfies Eq. (4.10). From this condition on $f(z)$, it follows that

$$\begin{aligned}
z = 1-p+px \qquad &\Leftrightarrow \qquad x = \frac{z-1+p}{p}, \\
f(z) = 1-\tilde{p}+\tilde{p}x \qquad &\Leftrightarrow \qquad x = \frac{f(z)-1+\tilde{p}}{\tilde{p}},
\end{aligned}$$

so

$$\begin{aligned}
x &= \frac{1}{\tilde{p}}(f(z)-1+\tilde{p}) = \frac{1}{p}(z-1+p), \\
f(z) &= 1-\tilde{p}+\frac{\tilde{p}}{p}(z-1+p) = 1+\frac{\tilde{p}}{p}(z-1).
\end{aligned}$$

Therefore, $G_{A'}(x) = G_{\bar{A}_0}(1+\frac{\tilde{p}}{p}(x-1))$ satisfies our conditions. The problem has been reduced to the problem studied in Chapter 3, and it can be solved using the same methods.

4.3 Example: Solving the Reduced System for Erdős-Rényi Networks

Once again, we consider a pair of coupled Erdős-Rényi networks. In [16], Huang et al. use an analytic method to find the critical value p_c in the case when the initial attack removes nodes from both networks in the pair. In order to remain consistent with the rest of this report, we will continue to work with an asymmetric initial attack, where nodes are only removed from the network A. The following section makes use of a combination of the techniques from section (3.4), and ideas from [16], to solve a targeted attack problem on coupled Erdős-Rényi networks with equal mean degrees. We consider the example when $\alpha = 1$, and observe in this case that $G_\alpha(z) = \sum_k P_A(k) z^k = G_A(k)$, so Eq. (4.8) can be written

$$G_{\bar{A}_0}(z) = \frac{1}{p} \sum_k P_A(k) h^k z^k = \frac{1}{p} G_A(hz), \tag{4.11}$$

where $h = G_A^{-1}(p)$. The generating function $G_{A'}(z)$ can therefore be written in terms of $G_A(z)$ as follows:

$$\begin{aligned} G_{A'}(z) &= G_{\bar{A}_0}\left(1 + \frac{\tilde{p}}{p}(z-1)\right) \\ &= \frac{1}{p} G_A\left(h\left[1 + \frac{\tilde{p}}{p}(z-1)\right]\right), \end{aligned} \tag{4.12}$$

where

$$\tilde{p} = \frac{\sum_k P_A(k) k h^k}{\sum_k P_A(k) k} = \frac{h G'_A(h)}{G'_A(1)}.$$

In our example, A and B are Erdős-Rényi networks, each with mean degree a. It then follows that $G_A(z) = \exp[a(z-1)]$, so $h = \frac{1}{a}\log(p) + 1$. Additionally, $G'_A(z) = a\exp[a(z-1)] = aG_A(z)$, so we can calculate

$$\tilde{p} = \frac{ahG_A(h)}{aG_A(1)} = ph.$$

We can now simplify $G_{A'}(z)$ even further:

$$\begin{aligned} G_{A'}(z) &= \frac{1}{p} G_A\left(h + \frac{\tilde{p}}{p} h(z-1)\right) \\ &= \frac{1}{p} G_A(h + h^2(z-1)) \\ &= \frac{1}{p} \exp[a(h - 1 + h^2(z-1)] \\ &= \frac{1}{p} G_A(h) \exp[ah^2(z-1)] \\ G_{A'}(z) &= \exp[ah^2(z-1)]. \end{aligned}$$

The probability generating function uniquely determines a distribution, so we have shown that A' has the same degree distribution as an Erdős-Rényi

network with mean degree ah^2. The equivalent equations to (3.13) are

$$\begin{cases} f_{A'} = \exp[-ah^2 p(1 - f_{A'})(1 - f_B)], \\ f_B = \exp[-ap(1 - f_{A'})(1 - f_B)]. \end{cases} \tag{4.13}$$

Observe that $f_{A'} = f_B^{h^2}$, so the substitution $f = f_B$ yields

$$f = \exp[-ap(1 - f^{h^2})(1 - f)], \tag{4.14}$$

which we solve graphically to find the critical values of $f = f_B$ and of $f_{A'}$.

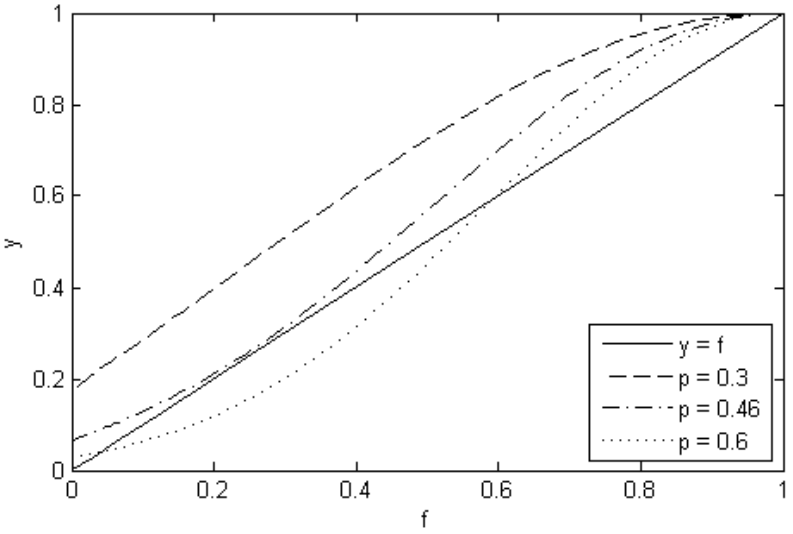

Figure 4.2: A graphical solution to Eq. (4.14), for coupled Erdős-Rényi networks with equal mean degrees ($a = b = 6$). We plot the curve $y = \exp[-ap(1 - f^{h^2})(1 - f)]$ for $p = 0.3$, 0.46 and 0.6; and we consider the points of intersection with the line $y = f$.

We proceed in the same way as in Chapter 3. For any value of p, a trivial solution to Eq. (4.14) is given by $f = 1$. We wish to find the critical value p_c of p, when non-trivial solutions first become possible. The critical value p_c is the value of p for which the line $y = f$ and the curve $y = \exp[-ap(1 - f^{h^2})(1 - f)]$ meet at a tangent. When the mean degree is $a = 6$, Fig. 4.2 suggests that p_c is about 0.46. At p_c, the derivatives of the line and the curve are equal, so we have

$$\begin{aligned} 1 &= \frac{\mathrm{d}}{\mathrm{d}f}\left[\exp[-ap(1 - f^{h^2})(1 - f)]\right] \\ &= \exp[-ap(1 - f^{h^2})(1 - f)] \times \frac{\mathrm{d}}{\mathrm{d}f}\left[-ap(1 - f^{h^2})(1 - f)\right] \\ &= -apf\left[-h^2 f^{(h^2-1)}(1 - f) - (1 - f^{h^2})\right] \\ 1 &= ap\left[h^2 f^{h^2} + f - (h^2 + 1)f^{(h^2+1)}\right]. \end{aligned} \tag{4.15}$$

Simultaneously solving Eq. (4.14) and (4.15) numerically, when $a = 6$, yields the approximate solution $p_c \approx 0.4711$. The corresponding value of μ_∞ is approximately 0.2386. For other values of a between 3 and 8, the numerical solutions for p_c and f, along with the corresponding values of μ_∞, are shown in Fig. 4.3 (for MATLAB code, see Appendix B).

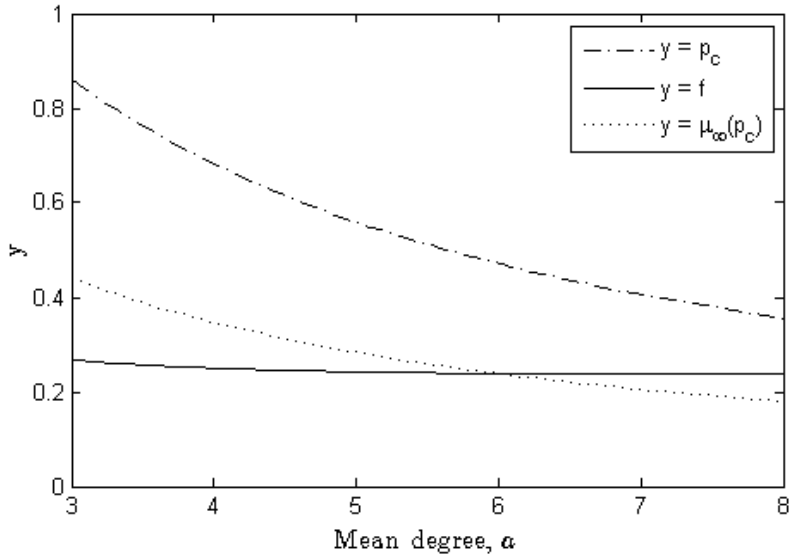

Figure 4.3: Numerical solutions of Eqs. (4.14) and (4.15).

4.4 Simulations

I simulated the cascade process that follows from an initial attack where nodes are targeted with probability proportional to their degree (i.e. $\alpha = 1$ in Eq. (4.1)). In this section, we restrict the value of p to between 0.1 and 1. This is to ensure that there are always enough nodes with non-zero degree to carry out the initial attack. My MATLAB code is included in Appendix B.

4.4.1 Erdős-Rényi Networks

The results for a targeted attack on coupled Erdős-Rényi networks look very similar to the results for cascades caused by the failure of nodes uniformly at random. As in the previous chapter, when p is small the network tends to break into small clusters, and when p is larger a giant cluster survives. As before, there is a sudden transition between the two states, but in this case the transition happens at a slightly larger value of p, at about $p = 0.46$. This agrees with the critical value p_c derived in section 4.3, and suggests that, for coupled Erdős-Rényi networks, a targeted attack is more effective than an attack uniformly at random, but not by very much. The reason for this is that the degree distribution in an Erdős-Rényi network is not very broad, so even the

highest degrees are relatively close to the mean degree. Therefore, choosing nodes based on their degrees is quite similar to choosing nodes uniformly at random.

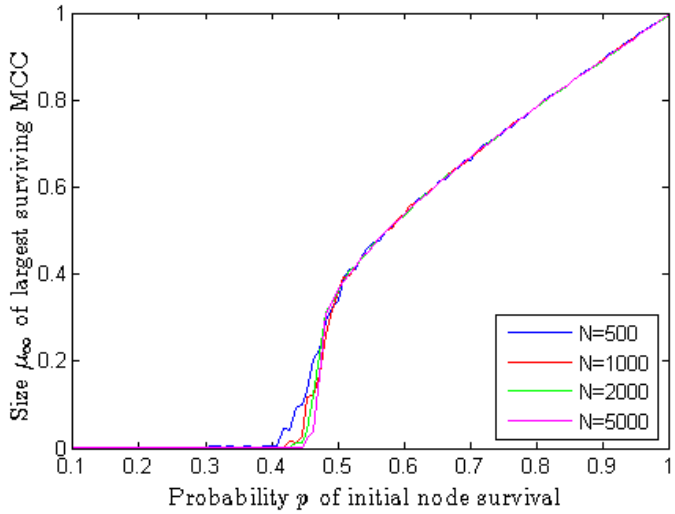

Figure 4.4: Simulations of the cascade process for an attack targeted by node degree on coupled Erdős-Rényi networks with equal mean degree ($a = b = 6$). All curves are averages of 20 simulations.

4.4.2 Scale-Free Networks

Figure 4.5 (page 33) shows that attacks where nodes are targeted with probability proportional to their degree are extremely effective on coupled scale-free networks. This is not surprising, since a necessary condition for the existence of a giant mutually connected component in a pair of interdependent networks is the existence of a giant component in each of the networks. It has been shown that targeted attacks can quickly destroy the giant component in single-layer scale-free networks [9], which would clearly make it impossible for a giant mutually connected component to exist in a coupled system.

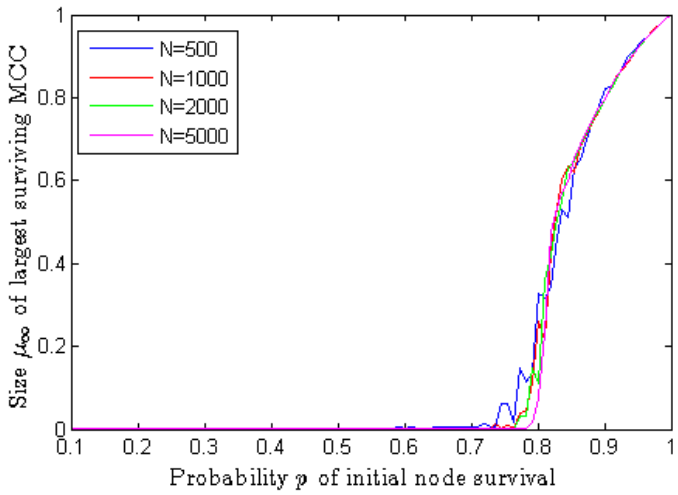

Figure 4.5: Simulations of the cascade process for coupled scale-free networks. All curves are averages of 20 simulations.

5. Targeting by Number of Second-Neighbours

Node degree is not the only measure of a node's importance in a network. We call quantities that measure the importance of a node *centrality measures* (page 9, [23]). When attacking a network, one can choose to target nodes according to any centrality measure. A natural extension to the work in Chapter 4 is to ask how robust interdependent networks are against an attack where nodes are targeted according to how many "second-neighbours" they have. If it is effective to base an attack on the degree of nodes, it might be even more effective to base an attack on the *2-degree* of nodes.

Definition 5.1. *In a network A, a node u is a* ***2-neighbour*** *of a node v if $u \neq v$ and u and v have a shared neighbour. The* ***2-degree*** *of v is*

$$d_A^2(v) = |\{u \in A \mid u \text{ is a 2-neighbour of } v\}|.$$

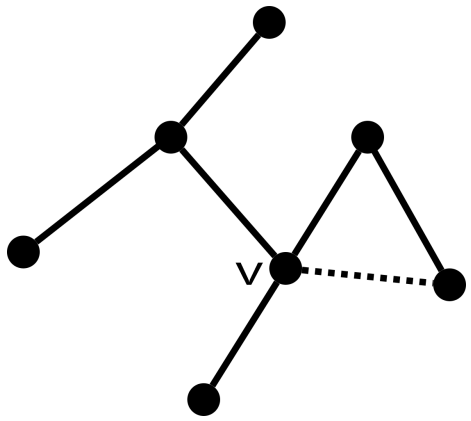

Figure 5.1: The neighbours and 2-neighbours of a node v. In the network made up only of the solid edges, the degree and the 2-degree of v are both 3. If the dotted edge is included, the degree of v is 4, and the 2-degree of v is also 4, since the two nodes that make up the triangle with v count both as neighbours and as 2-neighbours of v.

For example, in a friendship network, the degree of a person is the number of friends they have, and the second degree is the number of individuals who

are "a-friend-of-a-friend". In addition to the degree of a node, the 2-degree gives extra information about the "reach" of a node in a network, whilst still being simple to calculate. As is shown in Fig. 5.1, it is possible for a node to be both a neighbour and a 2-neighbour of another node. However, this only occurs when there are small loops in the network. Therefore, if a network is locally tree-like (see footnote on page 16), we can assume that the set of neighbours of a node v and the set of 2-neighbours of v are disjoint sets.

In this chapter, we modify the method from [16], seen in Chapter 4, to model the cascade of failures that results from an initial attack that targets nodes based on their 2-degrees. In the previous chapter, the analysis was reduced to a system of equations in terms of the generating functions $G_A(z), G_B(z)$ for the degree distributions of the coupled networks A and B. In this chapter, we instead require the generating functions for the distributions of 2-degrees in the two networks.

5.1 Generating Functions for the Distribution of 2-Degrees

We aim to reduce the targeted attack case to a case of the failure of nodes uniformly at random. Therefore, we first consider the generating function for the 2-degrees in a network A.

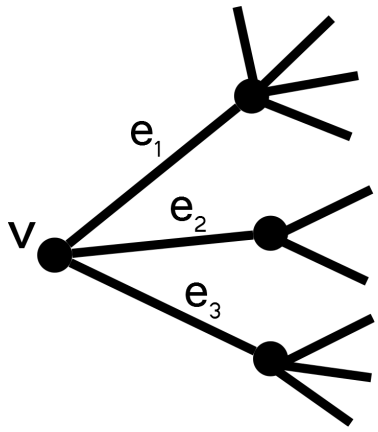

Figure 5.2: Calculating the 2-degree of a node. The node v has edges $\{e_1, e_2, e_3\}$, that lead to nodes with excess degrees $k_1^{\text{ex}} = 4, k_2^{\text{ex}} = 2$, and $k_3^{\text{ex}} = 3$.

Let $G_X^s(z)$ be the generating function for the distribution of 2-degrees in a network X. We suppose that the network A has size N, and we consider the 2-degree of a node $v \in A$. Suppose that v has edges $\{e_1, e_2, \ldots, e_k\}$, where $k = d_A(v)$, and let k_j^{ex} be the excess degree of the node reached by following the edge e_j, as illustrated in Fig. 5.2. Observe that in locally-tree-like networks, the variables $(k_i^{\text{ex}} \, ; \, i \geq 1)$ are independent and identically distributed, with probability generating function $\widetilde{G}_A(z)$, and the collection is independent of the degree of v. The 2-degree of v is then given by $d_A^2(v) = \sum_{j=1}^{k} k_j^{\text{ex}}$. We can write

the generating function $G^s_A(z)$ in terms of $G_A(z)$ and $\widetilde{G}_A(z)$, using the random sum property of generating functions (page 11).

$$G^s_A(z) = G_A(\widetilde{G}_A(z)). \tag{5.1}$$

We can now calculate the generating function for the 2-degrees in the network A_0, which is formed (as in Chapter 4) by removing a fraction $1-p$ of the nodes of A, uniformly at random. We write $G^s_{A_0}(z)$ in terms of $G^s_A(z)$, which yields an equivalent expression to Eq. (3.1).

$$\begin{aligned}
G^s_{A_0}(z) &= \sum_{m=0}^{\infty} z^m \Pr\left(d^2_{A_0}(v) = m\right) \\
&= \sum_{m=0}^{\infty} \sum_{r=m}^{\infty} z^m \Pr\left(d^2_{A_0}(v) = m \mid d^2_A(v) = r\right) \times \Pr\left(d^2_A(v) = r\right) \\
&= \sum_{r=0}^{\infty} \sum_{m=0}^{r} z^m \binom{r}{m} p^m (1-p)^{r-m} \Pr\left(d^2_A(v) = r\right) \\
&= \sum_{r=0}^{\infty} \Pr\left(d^2_A(v) = r\right) (1-p+pz)^r \\
G^s_{A_0}(z) &= G^s_A(1-p(1-z)).
\end{aligned} \tag{5.2}$$

5.2 Modifying the Method, for Initial Attack According to 2-degree

We modify the method given in Chapter 4, to analyse the effect of carrying out an initial attack in which nodes are targeted according to their 2-degree. We define a family of probability distributions $Y_\alpha(s_i)$ (the equivalent of the distributions $W_\alpha(k_i)$ on page 25) as follows:

$$Y_\alpha(s_i) = \frac{s_i^\alpha}{\sum_{i=1}^N s_i^\alpha}, \qquad \alpha \in \mathbb{R},$$

where s_i is the 2-degree of the node i. As before, if $\alpha < 0$, we ignore any nodes with $s_i = 0$.

We will follow the method, hoping to find an equivalent to Eq. (4.7). Equation (4.7) is in terms of $W_\alpha(k_i)$, and has a solution that is based on the degree distribution of the network A. If we could find an equivalent equation, it would most likely be in terms of the distribution $Y_\alpha(s_i)$, so we expect any solutions to be based on the distribution of 2-degrees in A. Such a solution would give an expression for $G^s_{A_0}(z)$, from which we can derive an expression for $G^s_{A_0}(z)$.

The final step in the method is to find a network A', with $|A'| = N$, such that the generating function $G^s_{A'}(z)$ satisfies

$$G^s_{A'}(1-p(1-z)) = G^s_{A_0}(z).$$

At this stage, if it is possible to find a degree distribution (and therefore an excess degree distribution) for A' that satisfies $G_{A'}\left(\widetilde{G}_{A'}(z)\right) = G^s_{A'}(z)$, then the problem can be solved using the same techniques as in the case of the failure of nodes uniformly at random.

5.3 Applying the Method

The network $\bar{A}_0$ is defined as the network formed from A by randomly removing a fraction $1-p$ of the nodes, according to the probabilities $Y_\alpha(s_i)$, but leaving the edges incident to the removed nodes. Let $P^s_p(s)$ be the 2-degree distribution of the nodes in $\bar{A}_0$, and let $A^s_p(s)$ be the number of nodes in $\bar{A}_0$ with 2-degree s, so $P^s_p(s) = \frac{A^s_p(s)}{pN}$.

We remove one additional node, say node u, according to the probabilities $Y_\alpha(s_i)$. The probability that u has 2-degree s is normalised as

$$\frac{P^s_p(s)s^\alpha}{\sum\limits_r P^s_p(r)r^\alpha} = \frac{P^s_p(s)s^\alpha}{\langle s^\alpha\rangle}.$$

For $s \in \{1, 2, \ldots, N-1\}$, the expected number of nodes with 2-degree s remaining in the modified $\bar{A}_0$ is given by

$$A^s_{\left(p-\frac{1}{N}\right)} = A^s_p(s) - \frac{P^s_p(s)s^\alpha}{\langle s^\alpha\rangle} - \delta^-_s + \delta^+_s,$$

where δ^-_s is the expected number of nodes that have 2-degree s before the removal of u, and 2-degree less than s after the removal of u; and δ^+_s is the expected number of nodes that have 2-degree larger than s before the removal of u, and 2-degree equal to s after the removal of node u. In Eq. (4.3), when we only considered the degrees of nodes, there were no equivalent quantities to δ^-_s and δ^+_s. This is because — as illustrated in Fig. 4.1 — removing nodes, but not their edges, from a network does not change the degrees of the remaining nodes (although it does change the degree distribution of the modified network). In contrast, Fig. 5.3 (on page 38) demonstrates how removing a node from a network can affect the 2-degrees of remaining nodes, even before any edges are removed.

If our method is going to work for initial attacks based on 2-degree, we will have to deal with the extra terms δ^-_s and δ^+_s. However, these quantities are not simple to calculate, and it is not obvious how to proceed analytically. Therefore, we will instead consider computer simulations of attacks based on 2-degree, in the hope that we can observe empirically how the system behaves.

5.4 Simulations

As in previous chapters, we simulate the cascade process on coupled Erdős-Rényi and scale-free networks. The code for an attack targeted by 2-degree is shown in Appendix C.

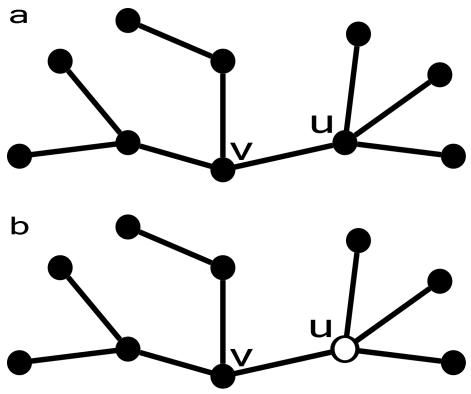

Figure 5.3: Removing a node from the network can affect the 2-degrees of the remaining nodes. Panel (a) shows a network A, and panel (b) shows the network after the removal of node u, but before removing any edges that were joined to u. In A, the node v has degree 3 and 2-degree 6. After removal of u, the degree of v is unchanged, but v now has a 2-degree of 3.

We have seen that an attack targeted by node degree on coupled Erdős-Rényi networks has a very similar effect to the failure of nodes uniformly at random. The simulations whose results are depicted in Fig. 5.4 (page 39) show that an attack targeted by 2-degree is practically equivalent to an attack targeted by degree. The plots in Figs. 4.4 and 5.4 are almost identical, discounting differences in the smoothness of the curves that are due to the different numbers of simulations performed. This is because in Erdős-Rényi networks, degree and 2-degree are strongly correlated, as we can see in Fig. 5.6 (page 40).

The figure for scale-free networks (Fig. 5.5, page 39) also looks similar to the corresponding figure in the previous chapter (Fig. 4.5), and this indicates that a targeted attack based on 2-degree is an effective way to destroy the giant cluster in coupled scale-free networks. However, the transition from the non-existence to the existence of a giant cluster takes place at lower values of p in Fig. 5.5 than in Fig. 4.5, so it would usually be more effective to target nodes based on their degree, rather than their 2-degree.

The reason for this behaviour is as follows. There are two main ways that a node v can have a high 2-degree. The first way is if the node itself has high degree, so it has many neighbours that each contribute their own neighbours to the 2-degree of v. In Fig. 5.7 (page 40), these are the points in the top right of the scatter plot. The second way is if the node itself has a low degree, but it has a neighbour with a high degree. These are the points on the left-hand side of Fig. 5.7, that have degrees of 1,2 or 3, but 2-degrees of about 100. In the first case, the node v is likely to play an important role in the connectedness of the network, and removing it could cause the network to become fragmented. In the second case, however, there is no reason that the removal of v would have a significant impact on the connectedness of the network. Therefore, an attack targeted by 2-degree can be seen as a weakened form of an attack targeted by degree, and this is reflected in Fig. 5.5.

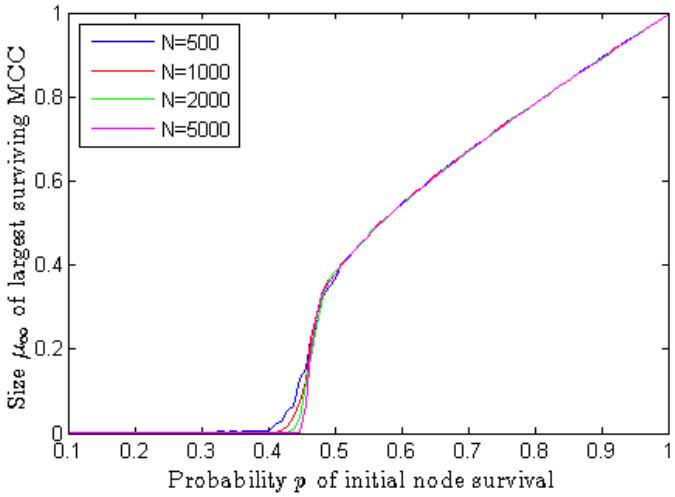

Figure 5.4: Simulations of the cascade process for an attack targeted by 2-degree on coupled Erdős-Rényi networks, with mean degrees $a = b = 6$. 100 simulations were carried out for $N = 500$, 50 for $N = 1000$, and 20 each for $N = 2000$ and $N = 5000$.

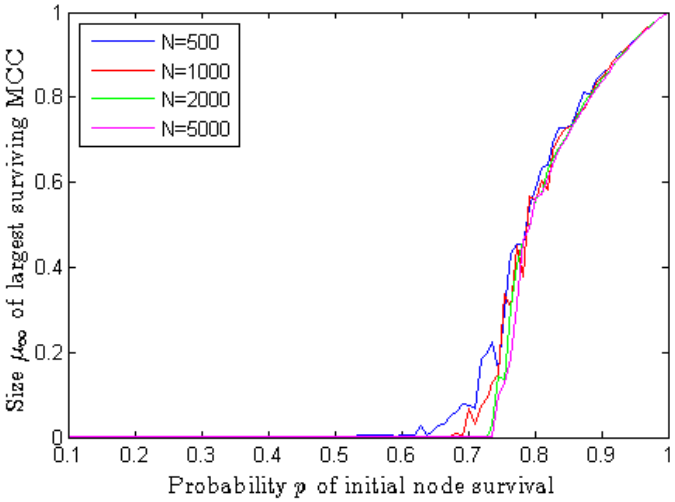

Figure 5.5: Simulations of the cascade process for an attack targeted by 2-degree on coupled scale-free networks. All curves are averages over 20 simulations.

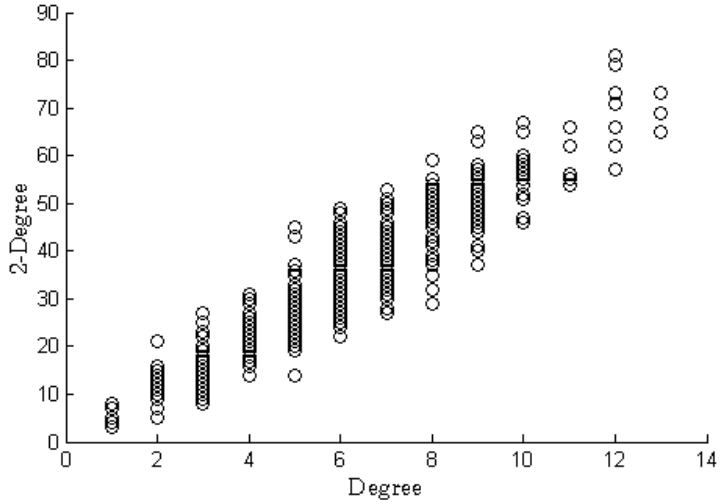

Figure 5.6: Scatter diagram of node degrees and 2-degrees in an Erdős-Rényi network with 500 nodes, and mean degree 6.

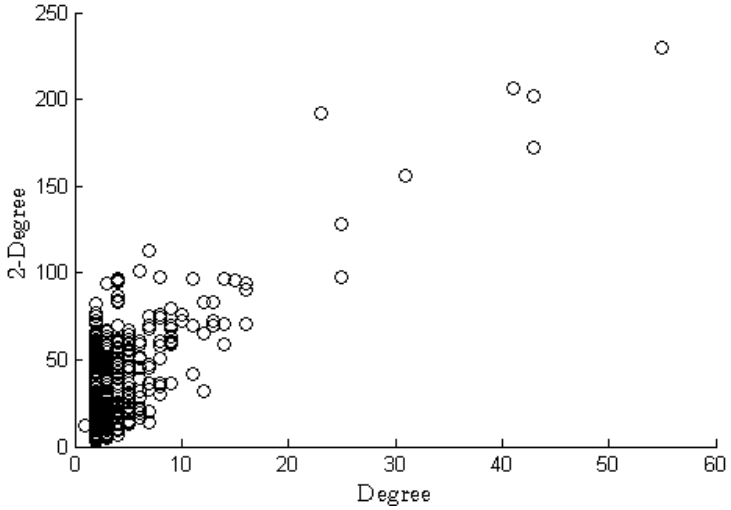

Figure 5.7: Scatter diagram of node degrees and 2-degrees in a scale-free network with 500 nodes.

6. Conclusion

In this dissertation, I have investigated the robustness of pairs of interdependent networks against the removal of nodes chosen in three ways: uniformly at random, randomly according to degree, or randomly according to 2-degree. I used probability generating functions to examine the problem, applied analytic methods for specific examples, then performed simulations for coupled Erdős-Rényi and scale-free networks.

In the case where nodes initially fail uniformly at random, I calculated the critical value p_c of p (where p is the fraction of nodes that survive the initial failure) for coupled Erdős-Rényi networks with equal mean degree a, and found that the critical value decreases as we consider larger mean degrees. Simulations showed that coupled scale-free networks are more vulnerable than single-layer scale-free networks to the removal of nodes uniformly at random, though there was no sudden jump in the size of the largest surviving cluster, as seen in the simulations for coupled Erdős-Rényi networks.

When the initial attack removes nodes with probabilities proportional to their degrees, the cascade of failures behaves very similarly to the failure uniformly-at-random case on coupled Erdős-Rényi networks. In contrast, I showed that for coupled scale-free networks, an attack targeted by degree is much more effective than an attack that removes nodes uniformly at random.

Although the generating function method from Chapter 4 was not successful in the study of an initial attack targeted by 2-degree, simulations showed that for both coupled Erdős-Rényi networks and coupled scale-free networks, the cascade of failures behaves in a similar way to the cascade resulting from an initial attack that is targeted according to degree. In the Erdős-Rényi case, the two types of targeted attack were practically equivalent, whereas for scale-free networks, targeting based on 2-degree was shown to be a little less effective.

The simulations that I performed were limited by time and computer power. Averaging over a larger number of simulations would reduce the impact of fluctuations in the data, producing smoother, clearer plots. Running simulations on graphs with more nodes would require more computing power, but would allow better comparisons to be made between simulations and analytic results. The code that I used to simulate scale-free networks did not have an option to specify the parameters of the distribution. If the simulations were repeated, it might be instructive to consider scale-free networks with a variety of parameter values.

There are several possible extensions to the work in this dissertation. We have assumed that in a pair of coupled networks, the structures of the two

networks are independent from each other. However, it is reasonable that in some situations there would be a level of correlation between the degrees of a node in each of the layers. Lee et al. consider this problem, for Erdős-Rényi networks, in [20].

Considering attacks that are targeted by degree allowed us to use a generating function method to investigate the cascades of failures. However, it could be interesting to target attacks based on other centrality measures. This is covered, for single-layer networks, in [18].

Finally, in the investigation of attacks targeted by 2-degree, a natural question is whether the high-degree nodes tend to be connected to other high-degree nodes (a property known as assortative mixing [23]). By considering networks with different levels of assortative mixing, we could gain further insights into the behaviour of this type of targeted attack.

Appendix A

MATLAB Code for Chapter 3

A.1 Solving Eq. (3.16) Numerically

A.1.1 findf.m

The MATLAB script **findf.m** solves Eq. (3.16) numerically to obtain the critical value f_c of f. The first part of the script plots the line $y = x$ and the curve $y = \exp(\frac{x-1}{2x})$. From this plot, we spot a suitable starting value ($x = 0.3$), then use the built-in function, "fzero", to find a numerical solution.

```
% FINDF.M
% Plot graph to find approximate solution:
fplot('x',[0,1])
hold on
func1 = @(x) exp((x-1)/(2*x));
fplot(func1,[0,1],'r')

% Find solution numerically:
syms x;
func2 = @(x) exp((x-1)/(2*x))-x;
f = fzero(func2,0.3)
atimesp = 1/(2*f*(1-f))
atimesmu = atimesp*(1-f)^2
```

A.2 Simulating Random Node Failure in Erdős-Rényi Networks

A.2.1 buldyrevGraph.m

When applied to the parameters *(t, N, edgeprob, linecolour)*, the function **buldyrevGraph.m** simulates t pairs of interdependent Erdős-Rényi networks with size

N, where each edge exists with probability *edgeprob*. The function simulates the cascade process, as described in Chapter 3, for 100 values of p between 0 and 1. Finally, the function plots the average size of the largest surviving mutually connected cluster, as a fraction of the size of the full network, against p (recall that p is the probability that a node is not removed in the initial failure).

```
% BULDYREVGRAPH.M
function [X,Y] = buldyrevGraph(t,N,edgeprob,linecolour)

tic
X = linspace(0,1);
Y = arrayfun(@(p) simBuldyrev(t,N,edgeprob,p),X);
plot(X,Y,'color',linecolour)
xlabel('Probability p of initial node survival')
ylabel('Size of largest component, as fraction of size of graph')
toc
end
```

A.2.2 simBuldyrev.m

When applied to parameters *(t, N, edgeprob, p)*, the function **simBuldyrev.m** simulates the cascade process with probability of initial node failure $1 - p$ on t pairs of interdependent Erdős-Rényi networks, each with N nodes, and edges that exist independently with probability *edgeprob*. The function returns the average size of the surviving cluster, as a fraction of the full network. This function uses the function "makerandCIJ_und" from the Brain Connectivity Toolbox ([5]), which, when given parameters *(N, noOfEdges)*, creates an Erdős-Rényi random graph with N nodes and *noOfEdges* edges.

```
% SIMBULDYREV.M
function avgsize = simBuldyrev(t,N,edgeprob,p)

possEdges = nchoosek(N,2);
largeComps = 1:t;

for j = 1:t,
    numberOfEdges1 = binornd(possEdges,edgeprob);
    numberOfEdges2 = binornd(possEdges,edgeprob);
    graph1 = sparse(makerandCIJ_und(N,numberOfEdges1));
    graph2 = sparse(makerandCIJ_und(N,numberOfEdges2));

    endgraph = cascadeP(graph1,graph2,p);
    sizeLargeComp = length(largest_comp(endgraph));
    largeComps(j) = sizeLargeComp;
end

avgsize = (sum(largeComps))/(t*N);

end
```

A.2.3 cascadeP.m

When applied to *(graph1,graph2,p)*, the function **cascadeP.m** removes a fraction $1-p$ of the nodes of *graph1* uniformly at random, then runs the cascade process described in Chapter 3. The output of the function is the resulting network.

```
% CASCADEP.M
function endgraph = cascadeP( graph1,graph2,p )

q = 1-p;
N = size(graph1,1);

s = floor(q*N);
failedNodes = randsample(N,s);

firstfails1 = removeNodes(failedNodes,graph1);
firstfails2 = removeNodes(failedNodes,graph2);

endgraph = runCascades(firstfails1,firstfails2);

end
```

A.2.4 runCascades.m

The function **runCascades.m** carries out steps 1–3 of the cascade process, stopping when no more edges are removed. The function returns the network that remains after step 4 of the process.

```
% RUNCASCADES.M
function endgraph = runCascades( graph1,graph2 )

d = 1; % Any non-zero will do.
next1 = graph1;
next2 = graph2;

while d~=0,
    [n1,n2,d1] = cascadesparse(next1,next2);
    [next2,next1,d2] = cascadesparse(n2,n1);
    d = d1+d2; % Indicates if edges have been removed.
end

next1;
next2;
endgraph = next1;
end
```

A.2.5 cascadesparse.m

When applied to *(graph1,graph2)*, the function **cascadesparse.m** checks each edge of *graph2* to see whether the endpoints of the edge are in the same connected component of *graph1*. If not, the edge is deleted from *graph2*. The function returns *graph1*, the updated version of *graph2*, and a variable which indicates whether or not edges were deleted. This function uses the function "get_components" from the Brain Connectivity Toolbox ([5]), which takes a graph as input, and returns a vector with one entry for each node, indicating which connected component the node is part of.

```
% CASCADESPARSE.M
function [newg1,newg2,noDel]=cascadesparse(graph1,graph2)

comps = get_components(graph1);
n = size(graph1,1);
[is,js,elems] = find(graph2);

mask = comps(is)==comps(js);

newis = is(mask);
newjs = js(mask);
newelems = elems(mask);

dels = length(is)-length(newis);
newg2 = sparse(newis,newjs,newelems,n,n);

newg1 = graph1;
noDel = dels;

end
```

A.2.6 largest_comp.m

The function **largest_comp.m** returns the largest connected component of a graph, in the form of a vector of nodes.

```
% LARGEST_COMP.M
function nodes = largest_comp( graph )

[comps,sizes] = get_components(graph); .
[s,c] = max(sizes);
l = length(comps);
elems = 1:s;
i = 1;

for j = 1:l,
    if comps(j) == c
        elems(i) = j;
        i = i+1;
```

```
    end
end

nodes = elems;
end
```

A.2.7 removeNodes.m

Given a list of nodes and a graph, the function **removeNodes.m** removes each node in the list from the graph, and removes all edges incident to the removed nodes.

```
% REMOVENODES.M
function newgraph = removeNodes( nodes,graph )

sortNodes = sort(nodes);
l = length(nodes);

for i=0:(l-1),
    nextgraph = removeNode(sortNodes(l-i),graph);
    graph = nextgraph;
end

newgraph = graph;

end
```

A.2.8 removeNode.m

Given a node i and a graph, the function **removeNode.m** removes i from the graph, and deletes any edges incident to i. The remaining nodes in the graph are renumbered after node i is removed.

```
% REMOVENODE.M
function   newgraph=removeNode(i,graph)

newgraph = graph([1:(i-1),(i+1):end],[1:(i-1),(i+1):end]);

end
```

A.3 Simulating Random Node Failure in Scale-Free Networks

Most of the MATLAB code that I used to simulate the failure uniformly at random of nodes in coupled scale-free networks is the same as the code for Erdős-Rényi networks. The only difference is the way that the coupled networks are first generated.

A.3.1 buldyrevGraphSF.m

The function **buldyrevGraphSF.m** is the equivalent of **buldyrevGraph.m**, for coupled scale-free networks. Instead of taking the two parameters N and *edgeprob*, the function just takes the parameter N, which is the size of the networks that are generated.

```
% BULDYREVGRAPHSF.M
function [X,Y] = buldyrevGraphSF(t,N,linecolour)

tic
X = linspace(0,1);
Y = arrayfun(@(p) simBuldyrevSF(t,N,p),X);
plot(X,Y,'color',linecolour)
xlabel('Probability p of initial node survival')
ylabel('Size of largest component, as fraction of size of graph')
toc
end
```

A.3.2 simBuldyrevSF.m

The function **simBuldyrevSF.m** is the equivalent of **simBuldyrev.m**. The function "pref.m", from the CONTEST toolbox for MATLAB [28], is used to generate random scale-free networks of size N, represented as sparse matrices.

```
% SIMBULDYREVSF.M
function avgsize = simBuldyrevSF(t,N,p)

largeComps = 1:t;

for j = 1:t,
    graph1 = pref(N);
    graph2 = pref(N);

    endgraph = cascadeP(graph1,graph2,p);
    sizeLargeComp = length(largest_comp(endgraph));
    largeComps(j) = sizeLargeComp;
end

avgsize = (sum(largeComps))/(t*N);

end
```

Appendix B

MATLAB Code for Chapter 4

B.1 Solving Eq. (4.14) Numerically

B.1.1 degFindf.m

The script **degFindf.m** solves Eqs. (4.14) and (4.15) numerically, for 100 values of a between 3 and 8, then calculates the corresponding values of μ_∞. The initial guess of $p = 0.5$, $f = 0.3$ is taken from Fig. 4.2. The quantities p_c, f and μ_∞ are plotted on one figure, as functions of the mean degree a.

```
% DEGFINDF.M

as = linspace(3,8);
ps = linspace(3,8);
fs = linspace(3,8);

for i=1:100
    a = as(i);
    aSimEqs = @(X) SimEqs([X(1);X(2);a]);
    X0 = [0.5;0.3]; % Initial guesses for p and f.
    [x,fval] = fsolve(aSimEqs,X0);
    p = x(1);
    ps(i) = p;
    f = x(2);
    fs(i) = f;
end

plot(as,ps,'b')
hold on
plot(as,fs,'r')

mus = ps.*(1-fs.^(((1./as).*log(ps) + 1).^2)).*(1-fs);
plot(as,mus,'m')
```

B.1.2 SimEqs.m

The script **SimEqs.m** defines Eqs. (4.14) and (4.15) as a MATLAB function of a vector X, where the first entry of X is p, the second entry is f, and the third entry is the mean degree a.

```
% SIMEQS.M
function F = SimEqs( X)
%  X(1) -> p, X(2) -> f, X(3) -> a
F = [X(2)-exp(-X(3)*X(1)*(1-X(2)^(((1/X(3))*log(X(1))+1)^2))*(1-X(2)));
1-X(3)*X(1)*((((1/X(3))*log(X(1))+1)^2)*X(2)^(((1/X(3))*log(X(1))+1)^2)
+X(2)-(((1/X(3))*log(X(1))+1)^2+1)*X(2)^(((1/X(3))*log(X(1))+1)^2+1))];
end
```

B.2 Simulating Attacks Targeted by Degree

The simulations of an attack targeted by degree, as described in Chapter 4, use the same MATLAB code as the code in Appendix A, except for a modification which changes the nature of the initial node failure. The modification is in the function **degreeCascadeP.m**, which takes the place of the function **cascadeP.m**.

B.2.1 degreeCascadeP.m

Given input *(graph1,graph2,p)*, the function **degreeCascadeP.m** removes a fraction $1-p$ of the nodes of *graph1*, where the probability that each node is removed is proportional to the degree of the node. Nodes with zero degree are ignored. The function then performs the cascade process on the two graphs *graph1* and *graph2*, and returns the resulting graph as output.

```
% DEGREECASCADEP.M
function endgraph = degreeCascadeP( graph1,graph2,p )

q = 1-p;
n = size(graph1,1); % Number of nodes.
degs = sum(graph1); % Vector of node degrees.
mask = degs>0; % To identify nodes with non-zero degree.
nodes = 1:n;
eligiblenodes = nodes(mask);
ws = degs(mask); % Degrees of non-zero-degree nodes.

s = floor(q*n); % Number of nodes to remove.
failedNodes = datasample(eligiblenodes,s,'replace',false,'weights',ws);

firstfails1 = removeNodes(failedNodes,graph1);
firstfails2 = removeNodes(failedNodes,graph2);

endgraph = runCascades(firstfails1,firstfails2);

end
```

Appendix C

MATLAB Code for Chapter 5

C.1 Simulating Attacks Targeted by 2-Degree

The simulations of the cascade of failures caused by an initial attack targeted by 2-degree use the same code as the simulations for cascades caused by failure uniformly at random, except for a modified version of **cascadeP.m**.

C.1.1 sndDegreeCascadeP.m

Given input *(graph1,graph2,p)*, the function **sndDegreeCascadeP.m** removes a fraction $1-p$ of the nodes of *graph1*, where the probability that each node is removed is proportional to the 2-degree of the node. Nodes with a 2-degree of zero are ignored. The function then performs the cascade process on the two graphs *graph1* and *graph2*, and returns the resulting graph as output.

```
% SNDDEGREECASCADEP.M
function endgraph = sndDegreeCascadeP( graph1,graph2,p )

q = 1-p;      n = size(graph1,1);
twopaths = graph1*graph1;
reduced2paths = twopaths>0; % Replace any non-zero element with 1.
twodegs = sum(reduced2paths)-1; % Vector of node 2-degrees.
mask = twodegs>0; % To identify nodes with non-zero 2-degree.
nodes = 1:n;
eligiblenodes = nodes(mask); % Nodes with non-zero 2-degrees.
ws = twodegs(mask); % 2-degrees of non-zero-2-degree nodes.

s = floor(q*n); % Number of nodes to remove.
failedNodes = datasample(eligiblenodes,s,'replace',false,'weights',ws);
firstfails1 = removeNodes(failedNodes,graph1);
firstfails2 = removeNodes(failedNodes,graph2);
endgraph = runCascades(firstfails1,firstfails2);
end
```

Bibliography

[1] R. Albert, H. Jeong and A.-L. Barabási, 'Error and attack tolerance of complex networks', *Nature*, 406 (2000), 378–382.

[2] A-L. Barabási, *Network Science* [pdf version], 2012, `http://barabasilab.neu.edu/networksciencebook/`, (accessed 15th March 2014).

[3] B. Berche, C. von Ferber, T. Holovatch and Y. Holovatch, 'Public transport networks under random failure and directed attack', arXiv:1002.2300v1 [physics.soc-ph] (2010).

[4] B. Bollobás, *Modern Graph Theory*, Springer, New York 1998.

[5] *Brain Connectivity Toolbox for MATLAB* (2010) [MATLAB toolbox], available at `https://sites.google.com/site/bctnet/` (accessed 13th November 2013).

[6] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley and S. Havlin, 'Catastrophic cascade of failures in interdependent networks', *Nature*, 464 (2010), 1025–1028.

[7] D. S. Callaway, M. E. J. Newman, S. H. Strogatz and D. Watts, 'Network robustness and fragility: percolation on random graphs', *Physical Review Letters*, 85 (2000), 5468–5471.

[8] A. Clauset, C. R. Shalizi and M. E. J. Newman, 'Power-law distributions in empirical data', *SIAM Review*, 51 (2009), 661–703.

[9] R. Cohen, K. Erez, D. Ben-Avraham, and S. Havlin, 'Breakdown of the Internet under intentional attack', *Physical Review Letters*, 86 (2001) 3682–3685.

[10] R. Cohen, K. Erez, D. Ben-Avraham, and S. Havlin, 'Resilience of the Internet to random breakdowns', *Physical Review Letters*, 85 (2000), 4626–4628.

[11] P. Erdős and A. Rényi, 'On the evolution of random graphs', *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5 (1960), 17–61.

[12] P. Erdős and A. Rényi, 'On the strength of connectedness of a random graph', *Acta Mathematica Scientia Hungary*, 12 (1961), 261–267.

[13] *Facebook Report for Wolfram|Alpha*, http://www.wolframalpha.com/input/?i=facebook+report (accessed 21st March 2014).

[14] E. N. Gilbert, 'Random Graphs', *The Annals of Mathematical Statistics*, 30 (1959), no.4, 1141–1144.

[15] P. Holme, 'Efficient local strategies for vaccination and network attack', *Europhysics Letters*, 68 (2004), 908–914.

[16] X. Huang, J. Gao, S. V. Buldyrev, S. Havlin and H. E. Stanley, 'Robustness of interdependent networks under targeted attack', *Physical Review*, 83 (2011), 065101.

[17] *Inkscape vector graphics editor* (version 0.48.4, 2012) [computer program], available at http://inkscape.org/en/ (accessed 18th March 2014).

[18] S. Iyer, T. Killingback, B. Sundaram and Z. Wang, 'Attack robustness and centrality of complex networks', *PLoS ONE*, 8 (2013), e59613.

[19] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, 'Multilayer Networks', arXiv:1309.7233v4 [physics.soc-ph] (2014).

[20] K.-M. Lee, J. Y. Kim, W.-K. Cho, K.-I. Goh and I.-M. Kim, 'Correlated multiplexity and connectivity of multiplex random networks', *New Journal of Physics*, 14 (2012), 033027.

[21] S. Melnik, A. Hackett, M. A. Porter, P. J. Mucha, and J. P. Gleeson, 'The unreasonable effectiveness of tree-based theory for networks with clustering', *Physical Review E*, 83 (2011), 036112.

[22] J.L. Moreno, *Who Shall Survive?*, Beacon House, New York 1934.

[23] M. E. J. Newman, *Networks, An Introduction*, Oxford University Press, Oxford 2010.

[24] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, 'Random graphs with arbitrary degree distributions and their applications', *Physical Review E*, 64 (2001), 026118.

[25] *NodeXL Excel Template* (version 1.0.1.251, 2014) [computer program], available at http://nodexl.codeplex.com/ (accessed 16th March 2014).

[26] C. Shalizi, *So you think you have a power law, do you?* [Presentation slides], 2010, http://www.stat.cmu.edu/~cshalizi/2010-10-18-Meetup.pdf (accessed 21st March 2014).

[27] D. Stirzaker, *Elementary Probability*, Cambridge University Press, Cambridge, 1994.

[28] A. Taylor and D. Higman, *CONTEST: A controllable test matrix toolbox for MATLAB* (2008) [MATLAB toolbox], available at http://www.mathstat.strath.ac.uk/outreach/contest/ (accessed 2nd February 2014).

[29] W. Zachary, 'An information flow model for conflict and fission in small groups' *Journal of Anthropological Research*, 33 (1977), 452-473.