# Evolving Network Model:

# A Novel Approach to Disease Simulation

Ayla Safran


Supervised by Dr. Dylan Shepardson

A thesis submitted to the department of Mathematics and Statistics in partial fulfillment of the requirements for the degree of Bachelor of Arts with Honors.

Department of Mathematics and Statistics

Mount Holyoke College

May 2018

## Acknowledgments

The completion of this project would not have been possible without the support of my advisor, Dylan Shepardson. You gave me the opportunity to partake in this project, and have pushed me to become a more capable, independent researcher during its course. Thank you so much for everything.

I would also like to thank Chaitra Gopalappa and Buyannemekh Munkhbat, with whom the project began. I am excited to see where your future work with this project takes you.

Finally, I would like to thank the Four College Biomath Consortium for supporting my research process, and the Mount Holyoke College Mathematics and Statistics department for giving me an academic home over the past three years. I am extremely sad to be leaving, but also excited to come back in the future as an alumna!

# Contents

# 1 Abstract

Disease modeling is an important tool in epidemiology. Effective epidemiological models can assist public health workers and policymakers in the development of effective disease prevention strategies and the efficient deployment of resources to inhibit the spread of disease. Expanding the scope of disease modeling techniques is an area of active research in applied mathematics. The two most commonly used modeling techniques are compartmental and agent-based disease models. Infectious diseases are transmitted through human populations by either direct or indirect contact. While for some diseases transmission is possible through indirect modes like airborne transmission, contact with contaminated objects, or insect or animal vectors, the spread of many is only possible through direct human-human contact. The type of contact required to lead to transmission varies widely based on the disease itself; for example influenza can be transmitted by casual contact, whereas HIV can only be spread through exchange of infected body fluids. The underlying biology that governs differences in transmission between different diseases is important to consider when developing disease models. While compartmental models operate under the assumption of homogeneous population mixing, agent-based models can capture the important features of the epidemiological contact network underlying the disease transmission. As a result, compartmental models make more accurate predictions for highly infectious diseases that spread by casual contact, whereas agent-based models provide increased accuracy for diseases that cannot spread through casual contact. Therefore, in many cases, agent-based models are able to model the spread of infectious diseases more accurately; however, their computational intensity makes their use unrealistic for modeling large population sizes. Thus, compartmental and agent-based models offer a trade-off between accuracy and efficiency (*i.e.* feasibility) in disease simulations.[15]

This project is an effort to create and validate an algorithm for a generating a novel disease model that possesses the network accuracy associated with agent-based models, but maintains some of the efficiency of compartmental models. This new, evolving network model compartmentalizes the majority of the network, while creating contact networks only for infected nodes, instead of for the entire population. As the infection spreads, more of the nodes' contact networks are modeled fully, until ultimately a full agent based network is generated. This method has been shown in preliminary testing to require significantly less computation time than simulating disease spread on full agent-based network models.[10] However, this new modeling technique is not an improvement over previous methods unless it can be shown to model networks as realistically as agent-based models can. Therefore, our goal is to demonstrate the ability of the algorithm for the spread of disease on an evolving network to replicate the desired properties of a target network and thus

accurately construct a network. Ultimately, the hope is that this evolving network algorithm will enable us to dynamically model the spread of a newly emerging or re-emerging disease in a human population by accurately representing salient features of the contact network in which disease transmission occurs. Such a model would help equip public health systems to study transmission scenarios and possible response strategies through the use of computer simulations in order to be able to plan for disease outbreaks and respond quickly when necessary.

There are many ways to measure how faithfully the generated network models real populations. Generally, the network properties should converge to the properties of the full network, both as the network evolves into the full agent-based network, and within any subnetwork at timesteps before that. In order to assess how well the algorithm is doing this, two main metrics, average degree and average clustering coefficient, are used. The inputs to the algorithm are a degree distribution and a clustering probability, so the model should generate a network whose properties match these input values; however we would also like it to replicate other properties of the network without explicitly having them as inputs. In order to assess this, other metrics, such as average path length and diameter, will be considered and studied as well. Finally, the algorithm will be tested using properties drawn from real social network data, in order to determine under what conditions it is able to reproduce a network similar to the true network.

# 2 Introduction

Epidemiology is the study of the health and diseases of human populations. This field is extremely important in prevention of disease and promotion of health, and has a few main functions, including the discovery of factors that affect health in order to prevent disease and injury, establishment of priorities for health related research, identification of at-risk populations for targeted intervention, and evaluation of the effectiveness of health services.[13] Within the field of epidemiology, mathematical modeling and computer simulations are very important tools. Accurately modeling and predicting the spread of infectious diseases can allow for the development of new vaccination and treatment strategies that more effectively target the specific disease and population. Furthermore, mathematical modeling tools enable us to develop prevention strategies for epidemiological events that have not yet occurred, and hopefully never will. As a result, developing disease modeling techniques that are accurate, flexible, and efficient is an essential to advancing epidemiological strategies.

Traditional epidemic theory relies on models that compartmentalize individuals based on their health status. The two most basic compartmental disease models are the susceptible-infectious-recovered (SIR), and susceptible-infectious-recovered-susceptible (SIRS) models. Each consists of a set of three or two differential equations, respectively, that relate the rates of change in the number of individuals in each of the compartments to one another. SIR models are applicable to diseases that an individual can only acquire once in his/her life, or in other words diseases that give an individual life-long immunity once recovered from the infection. SIRS models, by contrast, describe diseases to which an individual becomes once again susceptible after recovering.[15] Many more complicated compartmental models are used to more accurately model the dynamics of different diseases, and modifications to this basic system are pervasive in the field.

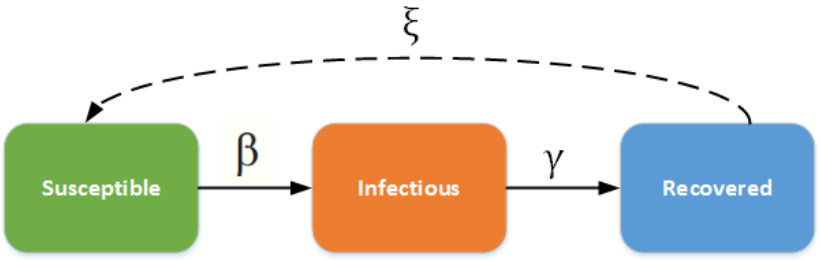



Figure 2.1: Depiction of SIR and SIRS models. The entire population is divided between the three compartments, and the variables over the arrows represent the rate of infection ($\beta$), rate of recovery ($\gamma$), and rate of loss of immunity ($\xi$). The recovery rate is determined by the average duration of infection for the disease. For the SIR model, $\xi = 0$ since the recovered nodes can never become reinfected.[26]

All compartmental models rely on certain assumptions about the disease in question. Since the only information contained in the model is the number of people or proportion of the population in each category, the individuals in the model have no more identity than their disease status. This means that compartmental models rely on the assumption of random mixing within the population; *i.e.* that any two individuals from the population have equal probability of coming into contact with one another. This assumption works well for diseases that are highly infectious, for which a short interaction is just as likely to lead to transmission as an interaction over an extended period of time. However, for infectious diseases with lower transmission rates, this assumption leads to inaccuracies in the model, since transmission is much more likely between, for example, family members than strangers who pass each other on the street. Because of this, compartmental models tend to overestimate disease cases for diseases that are more likely to spread through close contact.[15] Moreover, for diseases such as sexually transmitted infections (STIs), which can only spread through a specific type of interaction, the utility of compartmental models is limited.

In order to overcome this limitation, the scope of epidemiological models must be broadened to include agent-based modeling, which incorporates more information about the individuals in a population. In particular, network models provide a flexible approach to representing the relationships between various objects. Agent-based network models (ABNMs) generally simulate each individual in a population, as well as that individual's connections to others, over time. In the context of infectious diseases, this enables us to track the spread of an infection through a population. Network models allow for increased flexibility and accuracy in modeling the infection progress of a disease. To begin with, because each individual is modeled separately, the random mixing assumption is not necessary. Instead, the properties of a person's connections are modeled more specifically, and can be drawn from observed population demographics. In addition, network modeling allows for the introduction of diversity in connections, such as different types of interactions (e.g. family, social, random) with different weights, representing different transmission probabilities, based on the level of contact.[15] For example, in the case of Ebola, because the disease is transmitted only by direct contact with body fluids of a symptomatic infected person, transmission is much more likely between close contacts, such as family members, than casual contacts like acquaintances.[17] Thus, network modeling allows for much more detailed infection projections than compartmental modeling does.

These types of contact networks are modeled using graph theory. Any social network can be represented by a graph, where the nodes, or vertices, represent individuals, and the edges represent connections between the individuals along which transmission could occur. The main goal of network modeling is for the graph to accurately describe the population of interest. However, it is generally impossible to record all of the

connections in an entire population. As a result, computer simulations are often used, in which the projected network has the properties of a smaller sample size of the target network. While graph theory is a large field, in the case of network modeling, certain graph properties become especially important. One network attribute that is often emphasized is the degree of the nodes, or how many connections each node has to others. This parameter is most commonly quantified as the average degree of all of the nodes in the network, or as the degree distribution of the network, where degree distribution refers to the probability distribution that represents the proportion of nodes with each possible degree. Another important network quality is clustering. Loosely, clustering describes how well subsets of vertices are internally connected to each other, or clustered together. Network clustering is generally measured by an average clustering coefficient, which is calculated by dividing the number of triangles between nodes by the number of connected triplets of nodes. Clustering can also be studied more closely by looking at cluster size and structure, among other properties. Other important network parameters include average path length, diameter, and density.[2]

Agent-based network modeling has proven to be immensely powerful, and allows for extensive network analysis and epidemic predictions. However, the main limitation to ABNMs concerns their computational complexity. Compartmental modeling requires only a calculation of the number of individuals with each disease status at a given time-step, which generally depends only on the transition rates among a relatively small number of compartments. In contrast, network models must contain information about the connections of every member of the population at all times, which makes agent-based simulations extremely time-consuming as the population size increases. Even though, in most situations, the critical part of an epidemic occurs far before the majority of the population is infected, traditional ABNMs simulate the full population at every timestep. However, were they to only simulate a smaller subset of the population, much computational time could be saved.

Compartmental modeling and agent-based network modeling are two techniques that provide different types of epidemiologically relevant information, and offer a trade-off between the accuracy of the predictions they make, and the efficiency of their computations. While compartmental models operate under the assumption of homogeneous mixing, agent-based models incorporate population heterogeneity and therefore are able to handle more diverse disease situations, such as diseases that are low-prevalence in the population as a whole but endemic in certain subpopulations. This provides a huge advantage in that it enables the implementation of disease prevention measures such as targeted intervention for particularly vulnerable subpopulations. The work discussed in this paper proposes a new modeling paradigm that aims to maintain the accuracy, flexibility, and specificity associated with ABNMs while significantly increasing computational effi-

ciency. The proposed evolving network model combines aspects of compartmental and agent-based network models. The word "evolving" in the name of this model describes the way in which the network changes as time progresses, and it is a "network" model in that it generates the contact network of individuals in the population, as in ABNMs.

In short, at the beginning of the evolving network model simulation, the entire population is modeled compartmentally, as in an SIR model, and everyone resides in the susceptible category. However, as individuals are moved into the infected compartment, their contact networks are generated, as in an ABNM. Eventually, if the entire population becomes infected, the full contact network is revealed. Essentially, this algorithm is achieved in four main steps:

1. Infect a single individual.
2. Generate contacts for each newly infected person according to degree distributions drawn from population demographics and a specified clustering probability.
3. Allow the infection to progress.
   (a) The probability of infection for each susceptible individual is $1-(1-\beta)^N$ where $N$ is the number of infected contacts and $\beta$ is the disease transmission probability.[1]
   (b) The probability that a currently infected node recovers is governed by a recovery probability.
4. Repeat steps 2 and 3 until desired stopping point.

A more detailed description of the algorithm is provided in Section 5, and an example of the network construction process is given in Figure 2.2.

This evolving network model significantly reduces the computation time compared to a standard agent based network model, since it only creates contact networks for infected individuals, rather than the entire population.[10] This is especially useful for modeling low prevalence newly emerging or reemerging infectious diseases, where information about the entire population is unnecessary in simulating the local disease trajectory, and where ABMNs are least computationally tractable. However, this model would not be an

[1]This equation comes from the binomial model of transmission. If $\beta$ is the transmission probability during an epidemiological contact event between a susceptible person and an infectious person, then the probability that the susceptible person does *not* become infected is $q = 1-\beta$, where $q$ is called the escape probability because it describes the likelihood that the susceptible person escapes without becoming infected. If the susceptible person has $n$ infected contacts, then the probability that they escape without becoming infected is $q^n = (1-\beta)^n$. Thus, the probability of the susceptible person becoming infected if they have $n$ infected contacts is $1-q^n = 1-(1-\beta)^n$. [12] To understand this equation, consider the effect of changing $\beta$ and $N$. As $\beta$ increases, since $0 \leq \beta \leq 1$, the quantity $(1-\beta)^N$ decreases, so the overall infection probability increases. Similarly, since the quantity $(1-\beta)$ is less than 1, as the number of infected contacts increases, $(1-\beta)^N$ decreases and thus this also causes the infection probability to increase.

improvement unless it also preserved the demonstrated accuracy of agent-based network models. There are two main ways in which this accuracy can be maintained: in the trajectory of the infection progress, and in the properties of the contact network that is generated. This paper will aim to demonstrate the ways in which the proposed evolving network model does in fact generate a network with properties that closely resemble those in an equivalent ABNM, on which the disease spreads similarly.

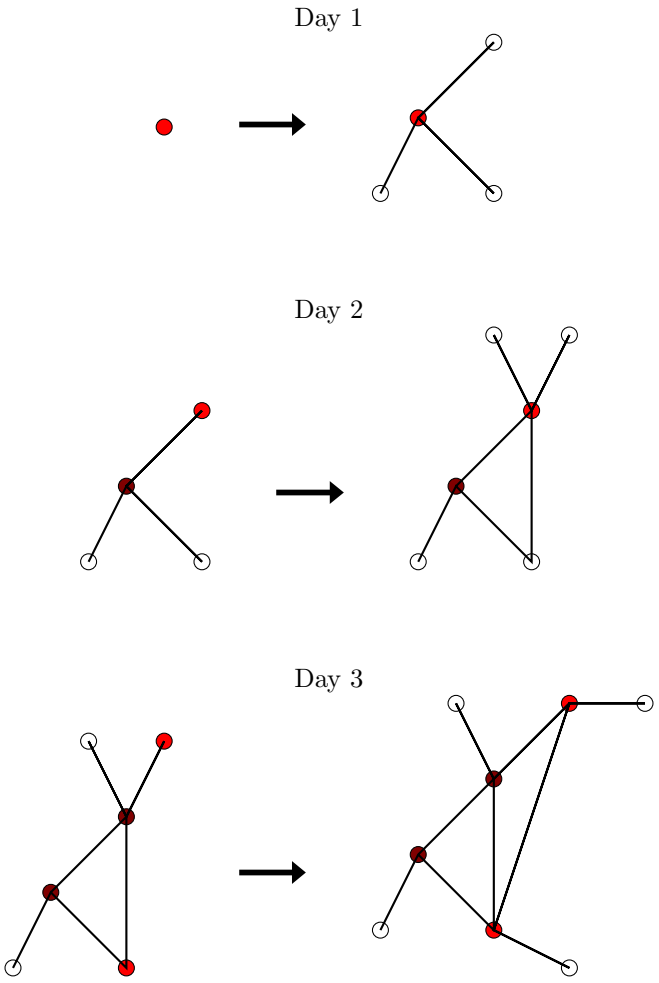



Figure 2.2: This figure illustrates an example of the first three days of an evolving network model simulation. The red nodes are newly infected (infected on that day), the dark red nodes are previously infected, and the white nodes are susceptible. As each node becomes newly infected, its contacts are picked from the larger population, which is not included in the graph.

# 3 Graph Theory Background

## 3.1 Graph Theory Definitions

Before discussing the proposed evolving network model in detail, it is important to review some basic graph theory terminology that will be used throughout this paper. Broadly speaking, a *graph* is a collection of points, which are called *nodes* or *vertices*, connected together in a particular way by line segments, which are called *edges*. Two nodes that have an edge connecting them are often referred to as *neighbors*. Despite the simplicity of this concept, graphs are very diverse and are extremely useful in many different fields. For example, graphs are often used to represent various types of real-world networks, where the nodes represent some element of the network, and the edges represent the connections between these elements. For instance a graph of the World Wide Web might use vertices to represent web pages, and edges the hyperlinks between them. Similarly, a social network could represent people as nodes, and friendships between people using edges.[2] There are a few basic categories of graphs that are widely used and important to consider.

Graphs can be directed or undirected. In a *directed graph*, edges have a direction associated with them. This means that there are two distinct types of edges between two nodes, A and B: an edge from A to B, and an edge from B to A. By contrast, in an *undirected graph*, there is no direction associated with edges, so an edge from node A to node B is the same as an edge from B to A. The edges in a graph can also have weights, or different numerical values, attached to them. A graph whose edges have weights is called a *weighted graph*, and one whose edges do not is an *unweighted graph*. In many situations, weighted graphs are useful for capturing additional information about the relationships between vertices. For example, in a transportation network nodes may represent cities and edges the roads connecting them, and edge weights could correspond to the average travel time between pairs of cities. In an epidemiological context, if vertices represent individuals in a population and edges indicate that two individuals are in epidemiological contact, edge weights could give the probability of disease transmission occurring between two individuals per unit time. Unweighted graphs can be thought of as weighted graphs where all edges have an equal weight (or a weight of one). Two graphs are called *isomorphic* if they have the same number of vertices, connected in exactly the same way. Formally, this is described as there being a bijection from one graph to the other.[19]

A *simple graph* is an undirected, unweighted graph that does not have edges connecting nodes to themselves or multiple edges between any pair of nodes. Many real-world networks can be represented by simple graphs, and this is the type of graph that will be focused on here. In addition, from now on the term *network* will be used interchangeably with simple, undirected graph. Two vertices in a graph, $u$ and $v$, are *connected*

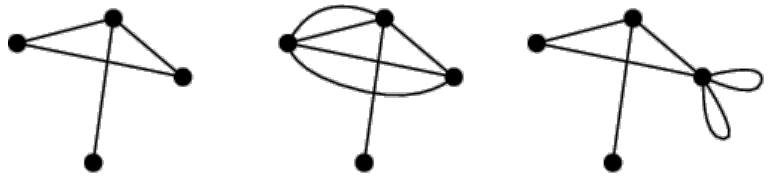


Figure 3.1: Example of a simple graph (left) and two graphs that are not simple because they contain multiple edges (center) or loops (right).[30]

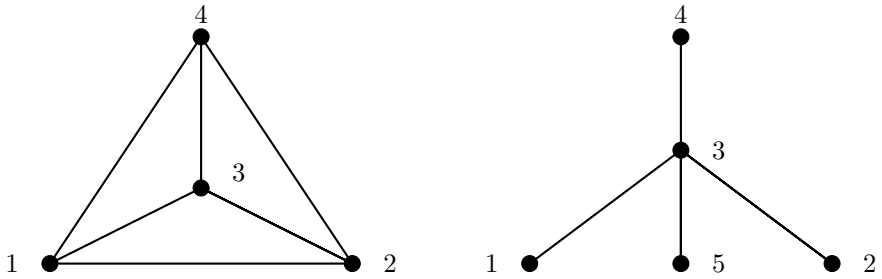
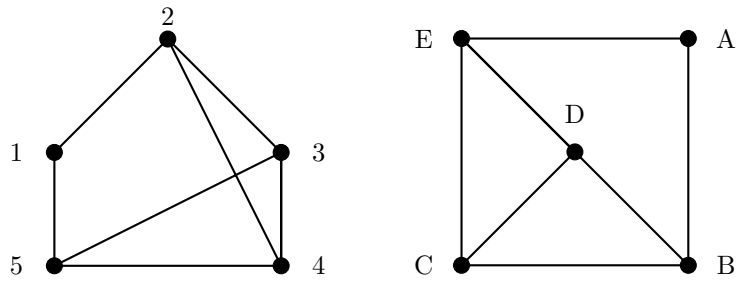



Figure 3.2: A simple, connected graph on 4 vertices (left) and a tree on 5 vertices (right). The graph on the left is not a tree because it contains cycles; for example, the path $1 \to 2 \to 3 \to 1$ is a cycle because it begins and ends at node 1. Both graphs are simple because they contain no loops or multiple edges, and both are connected because every node can be reach from any other in the graph by traveling along edges.

if there is a path, or a sequence of edges, beginning at $u$ and ending at $v$, or vice versa. A graph can also be *connected*, which in this context means that *any* node in the graph can be reached from any other by traveling along the edges. A graph that is not connected is made up of multiple disconnected *components*, or subgraphs. A connected graph is composed of a single component. A *cycle* is a path, or way of traveling along edges between nodes, that begins and ends at the same node without traversing any edge more than once. This can also be thought of as the union of two distinct paths that both begin and end at the same two nodes, but do not share any edges in common. A graph that is connected but does not contain any cycles is called a *tree*. In a tree, there is only one distinct path between any two nodes.[19] Figure 3.2 shows two examples of simple, connected graphs, one of which is a tree. In all of the following work, unless otherwise specified, we will assume that our graphs are simple and undirected.

Graphs are often represented using a few different types of matrices. The *adjacency matrix* for a network containing $N$ nodes is a square matrix with $N$ rows and $N$ columns. In a simple graph, the $(i, j)$ entry in the adjacency matrix is 1 if nodes $i$ and $j$ have an edge connecting them and 0 otherwise. In a simple graph, al of the entries on the main diagonal are zero since no node can have an edge connecting it to itself. The adjacency matrix is a straightforward way to summarize the connectivity of a graph, and two graphs

are isomorphic if and only if they can be represented by the same adjacency matrix.[19] An example of two isomorphic graphs and their adjacency matrix is shown in Figure 3.3. Another matrix sometimes used to describe graphs is the *degree matrix*, which is also an $N$-dimensional square matrix. In this matrix, all of the entries except for those on the main diagonal are zero, and the entries along this diagonal represent the *degree*, or total number of edges, of the corresponding node. For example, the $(i, i)$ entry represents the total number of edges between node $i$ and other nodes in the network. Finally, the *Laplacian matrix* is found by taking the adjacency matrix and subtracting it from the degree matrix (*i.e.* $L = D - A$). As a result, the diagonal entries of the Laplacian matrix represent the degree of each node, and the non-diagonal entries, $(i, j)$, are $-1$ if nodes $i$ and $j$ are neighbors, and 0 otherwise. There is an entire subfield of graph theory known as *spectral graph theory*, which is the study of the properties of graphs in relationship to the characteristic polynomial, eigenvalues, and eigenvectors of these matrices.




Adjacency Matrix (A)

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Degree Matrix (D)

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Laplacian Matrix (L)

$$\begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 3 & -1 & -1 \\ 0 & -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & -1 & 3 \end{bmatrix}$$

Figure 3.3: Loosely, a graph is isomorphic to another graph if it can be obtained from another graph by moving the vertices without changing the edge connections. To see that the two graphs above are isomorphic, imagine moving vertex 1 to vertex A, vertex 2 to vertex B, vertex 3 to vertex C, vertex 4 to vertex D, and vertex 5 to vertex E, while maintaining the same edge connections. As a result of the fact that these graphs are isomorphic, they can be represented by the same adjacency matrix. This matrix, as well as the degree matrix and Laplacian matrix that correspond to these graphs, are shown below the graphs.

## 3.2 Important Network Properties

### 3.2.1 Degree Distribution and Average Degree

As explained earlier, the degree ($k$) of a node represents the total number of edges between that node and other nodes in the network. In a simple graph, nodes cannot have edges connecting to themselves (loops),

and there can only be one edge between any pair of nodes. Thus, in a simple graph containing $N$ nodes, the maximum degree of any node is equal to the number of nodes in the network minus one $(N-1)$. The degree of the nodes in a graph can reveal a lot about the properties of the network. The most common ways to describe the degrees of the nodes in a network are using a probability distribution (the *degree distribution*) and an average value (the *average degree*). The average degree of a network is calculated simply by finding the arithmetic mean of all of the nodes in a network. For example, if we have a network of three nodes, one of which has degree 2 (has two edges, each directly connecting it to one of the other two nodes), and the others of which have degree 1 (see Figure 3.4 below), the average degree of the network is equal to $(2+1+1)/3 = 4/3$. In general, the average degree, $\bar{k}$, of an undirected network is given by:

$$\bar{k} = \frac{\sum_{i=1}^{N} k_i}{N}$$

where $N$ is the total number of nodes in the network and $k_i$ is the degree of the $i^{th}$ node. A higher average degree indicates that a network contains more edges, while a lower average degree suggests that the network is more sparsely connected.

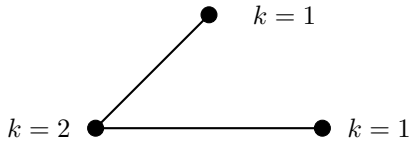



Figure 3.4: In the simple graph above, the average degree, $\bar{k}$ is calculated by summing the degree $(k)$ of each node and dividing by the number of nodes. In this case, the average degree is given by $\frac{1+2+1}{3} = \frac{4}{3}$
.

When more detail than just the average is desired about the degree of the nodes in a network, a degree distribution is used. The degree distribution, $P(k)$, represents the number of nodes in the graph that have degree $k$, where $k = 0, 1, 2, ..., N-1$. Thus, the degree distribution provides information about the variation and spread in the degrees of the nodes as well as the mean, and can also be used to estimate properties such as the median and mode of the degrees.[2] Sometimes a *degree sequence* is also used to describe the degree of the nodes in a graph, which is simply an ordered list containing the degree of each node in the graph. Generally degree sequences are non-increasing, meaning that the degrees are listed in order from larger to smaller.[7] Neither the degree distribution nor the degree sequence uniquely specify a graph, meaning that multiple non-isomorphic graphs can share the same degree distribution and degree sequence. However, these metrics can still serve as important ways of characterizing networks.

### 3.2.2 Graph Density

Graphs are often described as either dense or sparse, where the *density* of a graph refers to the number of edges in the graph relative to the maximum possible number of edges. For a simple, undirected graph on $N$ vertices, the maximum number of edges is $\frac{N(N-1)}{2}$, and therefore the density can be calculated as $D = \frac{2E}{N(N-1)}$ where $E$ is the number of edges in the graph. Graph density for a simple graph can range from zero (which would indicate that there are no edges) to one (the density of a complete graph, in which all nodes are connected to all other nodes). Like the degree distribution, graph density does not uniquely specify a graph, but is nevertheless useful in graph characterization.

### 3.2.3 Clustering Coefficient

Another important property of graphs is their level of clustering. *Clustering* describes the tendency of nodes in a graph to be connected preferentially to other nodes that share neighbors with them over other arbitrary nodes in the network, or the tendency to form cliques. For example, graphs representing social networks tend to display relatively high levels of clustering when compared to random graphs because people often have friends in common with their friends, based on shared location, activities, etc. The level of clustering in a graph is often measured using an average clustering coefficient. The most common way to calculate this quantity is by looking at the ratio of the number of triangles (three nodes connected by three edges, where each is directly connected to each other by a single edge) in a graph to the total number of connected triples (three vertices that connected by two or three edges). Using this definition:

$$C = \frac{3 \times \text{number of triangles}}{\text{number of conneted triplets of vertices}}$$

An example of this calculation is shown in Figure 3.2.3.

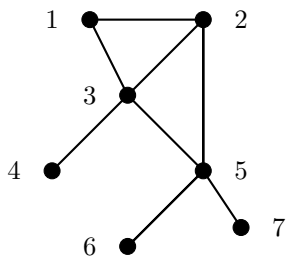



Figure 3.5: In the graph above, there are 12 sets of connected triplets: $\{1,2,3\}$, $\{2,3,4\}$, $\{1,3,4\}$, $\{1,3,5\}$, $\{1,2,5\}$, $\{3,4,5\}$, $\{2,3,5\}$, $\{3,5,6\}$, $\{2,5,6\}$, $\{3,5,7\}$, $\{2,5,7\}$, and $\{5,6,7\}$. However, only two of these are triangles: $\{1,2,3\}$ and $\{2,3,5\}$. Thus, the average clustering coefficient of this network is $\frac{3\times 2}{12} = \frac{1}{2}$.

A different definition of clustering coefficient was proposed by Watts and Strogatz (1998), which looks at clustering in a slightly more complex way. This definition requires the calculation of the local clustering coefficient of each node in the network, where local clustering coefficient for node $i$ is defined as the fraction of edges amongst neighbors of $i$ divided by the total number of possible edges between these neighbors. Mathematically, this can be represented as:

$$c_i = \frac{2|\{e_{j,h}\}_i|}{k_i(k_i - 1)}$$

where $k_i$ is the degree of node $i$ and $|\{e_{j,h}\}_i|$ is the number of edges that exist amongst all neighbors of $i$. Then, the global average clustering coefficient is obtained by taking the arithmetic mean of the local clustering coefficient for each node in the network. An average clustering coefficient of 1 would describe a connected graph, in which every node in the network is connected to every other node. This would mean that, for a network of size $N$, each node has degree $N-1$. A clustering coefficient closer to zero suggests less clustering behavior.[29] This second definition, while more descriptive, is also much more computationally intensive to calculate, and therefore this paper relies on the use of the first, simpler definition.

#### 3.2.4 Average Path Length and Diameter

In a graph, the *geodesic distance* between two vertices is the length of the shortest possible path between them (*i.e.* the minimum number of edges required to travel between them), and can be a useful property in characterizing networks. The longest geodesic path in a network is known as the graph's *diameter*, and the arithmetic mean of all of the geodesic distances in a graph is called the *average path length*. Both of these metrics can help describe the structure of a graph, but before they can be calculated, one must first find the shortest distances between each pair of nodes in the network. There are a few common approaches to this shortest-path problem, many of which are *graph traversal algorithms*, or algorithms that visit each vertex in a graph, checking or modifying these vertices along the way. These algorithms are usually classified by the order in which they visit the vertices. The first one we will consider is Dijkstra's algorithm.

*Dijkstra's algorithm* calculates the shortest path from a source node to a target node. This algorithm is known as a *greedy algorithm* because it chooses the best option locally at each step, in hopes of ultimately finding the best global option (*i.e.* the shortest path). Essentially, this algorithm occurs in steps, where each step establishes the shortest path from the source node, $u_0$, to some new node. Initially, it begins with a subgraph of the network that contains only the source node, $S_0 = \{u_0\}$. It then constructs a sequence of

subgraphs $S_0, S_1, \ldots S_{v-1}$ such that each subgraph contains one additional node. At the end of stage $i$, the length of the shortest path from $u_0$ to each other node in $S_i$ is known. Through successive repetitions of this process, the algorithm ultimately generates a tree such that the source node is connected to each other node in the tree by the shortest possible path, also known as the shortest path tree with root $u_0$. However, Dijkstra's algorithm does not keep track of the actual paths, just their lengths. It does this by labeling each node with its distance from the source node in the tree; initially all nodes are given the label infinity since they are not contained in the subgraph, but as they are incorporated into the subgraph this label is updated to reflect the geodesic distance between each of these nodes and the source node, which is also the distance between them in the tree. Thus, the algorithm terminates when the subgraph contains all nodes in the network (unless the network is not connected, in which case it would terminate when all nodes in the component containing the source node have been reached), and the output is the shortest distance from the source node to each other node in the network.

Generally, Dijkstra's algorithm is used on weighted graphs, although edges cannot have negative weights for this algorithm, and thus the algorithm seeks the "lowest cost" path between the source and target nodes. However, in the case of an unweighted graph, where we can imagine that all edges in the network are equally weighted, Dijkstra's algorithm reduces to a *breadth-first search* (BFS). In short, a BFS begins at an arbitrary node in the network, explores that node's neighbors, and then moves on to the neighbors of those nodes. In doing this, it will construct a tree that, as in Dijkstra's algorithm, has the property that the source node is connected to each other node by the shortest possible path. Since the edges are unweighted, the cost that this algorithm minimizes refers solely to the number of edges between vertices, and thus this type of search calculates the geodesic distance between the source node and each other node in the network. In order to ensure that all of the geodesic distances in a network are found, this process needs to be repeated using each node in the network as the source node, so for a network of size $N$, this process would be completed $N$ times. Both Dijkstra's algorithm and BFS run in polynomial time, which means that they are known to be computationally efficient algorithms.

Finally, the *Bellman-Ford algorithm* is a generalization of Dijkstra's algorithm that can accommodate negative edge weights. However, it is still limited in that it cannot calculate the shortest path lengths for a graph containing a negative cycle, or a cycle in which the edge weights sum to a negative number. This is because if a graph contains a negative cycle, then there is no cheapest path since any path containing a vertex on the negative cycle can be made cheaper by simply traveling around the negative cycle an additional time. For this reason, the Bellman-Ford algorithm is often used to detect the presence of negative cycles

in graphs. In short, the algorithm begins in the same way as Dijkstra's algorithm, initializing the distance to the source node to 0 and all other nodes to infinity. Then for all outgoing edges from the source node, if the distance to the destination node can be shortened by taking that edge, the distance is updated from infinity to the new lower value. At the $i^{th}$ iteration, the algorithm has found all shortest paths of at most length $i$ edges. Ultimately, a final scan of all the edges is performed and if any distance is updated, then a path of length greater than or equal to the number of the vertices in the graph has been found, which can only happen if there is a negative cycle in the graph. In this case, the algorithm terminates and reports the existence of a negative cycle. Otherwise, it reports the shortest path lengths as in Dijkstra's algorithm or a breadth-first search.[3] Bellman-Ford, like Dijkstra's algorithm and BFS, is a polynomial time algorithm.

In MATLAB, there is a built-in command, *distances*, that finds the length of the shortest path between each pair of nodes in a graph. The input to this command is a graph (which can be directed or undirected, weighted or unweighted, as long as it doesn't contain a negative cycle), and the output is a matrix $d$, where the $(i, j)$ entry represents the length of the shortest path between nodes $i$ and $j$. If the graph is disconnected and no path exists between nodes $i$ and $j$, then the $(i, j)$ entry has a value of $\infty$. This command uses a combination of the methods discussed to find the shortest paths. For an unweighted graph, the command uses a breadth-first search. If the graph has weighted edges which are all non-negative, Dijkstra's algorithm is used. Finally, if the graph has any negative edge weights, the Bellman-Ford algorithm is used.

These shortest path lengths can then be used to find the diameter (which is simply the largest entry in the distances matrix) and average path length of the graph. However, calculating the diameter and average path length is slightly more complicated for graphs that are not connected. The diameter of a disconnected graph is technically infinite, but is often defined as the diameter of the largest component in the network.[2] In this case, the diameter can be found by simply taking the largest *finite* entry in the distances matrix. The average path length of a disconnected graph is sometimes this handled by simply calculating the average path length for each connected component.[22] However, if a single metric is desired as the output, one could conceive of two possible ways to calculate the metric for the entire graph. In either case, since it is not possible to average infinite values, the infinite values in the distances matrix must be replaced with zeros. The entries of this new, updated matrix are then summed, and then there are two options for the next step. One could either divide this sum by the total number of entries in the distances matrix (which is generally how we would take an arithmetic mean), or by the number of nonzero entries in the matrix (which represents the number of paths). The first approach could result in an underestimation of the lengths of the paths that are present in the graph, and as a result, this paper relies on the second approach.

### 3.2.5 Spectral Graph Theory

Within graph theory, *spectral graph theory* specifically studies the properties of the matrices associated with a graph, namely the adjacency and Laplacian matrices. The main properties of these matrices that are studied are the characteristic polynomial, eigenvalues, and eigenvectors. Before discussing the relevance if these properties to spectral graph theory, we must understand what they represent in linear algebra. For any matrix $A$, the *eigenvalues* of this matrix are scalars $\lambda$ such that $AX = \lambda X$ for some vector $X$. The vectors $X$ that satisfy this relationship are known as *eigenvectors*. The *characteristic polynomial* of a square matrix $A$ is the polynomial whose roots are the eigenvalues of $A$. Formally, the characteristic polynomial of a square matrix $A$ is given by $p(\lambda) = det(A - \lambda I)$, where $I$ is the identity matrix of the same dimensions as $A$. Generally, the spectrum of a graph is defined as the set of eigenvalues of the adjacency matrix together with their multiplicities. Similarly, the Laplace spectrum of a simple undirected graph refers to the eigenvalues and their multiplicities for the Laplacian matrix of the graph.[8]

When considering a simple, undirected graph, since its adjacency matrix is real and symmetric, all of its eigenvalues are real. Similarly, the Laplacian matrix is always real and symmetric so it too has all real eigenvalues. One interesting concept in spectral graph theory is the idea of cospectral graphs. Two graphs are *cospectral* or *isospectral* if they have the same adjacency spectrum. Interestingly, two graphs can be isospectral without being isomorphic. For this reason, spectral graph theory has the potential to be useful in comparing graphs. It is important to keep in mind, however, that finding the spectrum for a large graph is very time consuming and generally not computationally feasible, since the the computation of the eigenvalues and eigenvectors of the matrices has time complexity $\mathcal{O}(n^3)$ where n is the dimension of the matrix.[27] (This operation can sometimes be achieved using the Coppersmith–Winograd algorithm which is $\mathcal{O}(n^{2.376})$, but this is still extremely computationally intense for large matrices.[9])

## 3.3 Types of Networks

Within the study of networks, there are a few specific types of networks often considered. In order to provide a framework within which to place our evolving network model, we will first consider a framework for constructing networks that can also be theoretically useful for studying networks in 3.3.1. Then, in 3.3.2 and 3.3.3, we will discuss some important properties often seen in networks.

### 3.3.1 Random Networks (Erdős-Rényi)

A *random graph* is one in which the edges follow a random distribution. There is a lot of interest in generating these types of graphs because they can be used to model real-life networks of which the exact structures are not known. In addition, when the properties governing the organization of graphs are unknown, these graphs often appear random. Thus, random graphs can be useful tools in both creating models of networks and analyzing the graphs of existing networks. The classic random graph model was proposed by Paul Erdős and Alfred Rényi, and is called the Erdős-Rényi model. In this model, a random graph is made up of $N$ nodes connected by $n$ edges that are chosen randomly. For a graph on $N$ nodes, there are $\frac{N(N-1)}{2}$ possible edges, so there are $\binom{N(N-1)/2}{n}$ possible ways to pick the edges in a random graph with $N$ vertices and $n$ edges, and thus $\binom{N(N-1)/2}{n}$ distinct random graphs (although these may not actually all be distinct, since many could be isomorphic copies of one another). This collection of graphs can be thought of as a probability space in which all outcomes have equal probability of occurring.

Alternatively, one can construct a random graph by beginning with $N$ nodes, for which each pair have a certain probability, $p$, of being connected by an edge. In this construct, the number of edges is variable, and depends on the value of $p$. As the number of samples gets large, the average degree of the resulting graph will converge to $\bar{k} = Np$. Thus it is easy to see how this model can be used to construct a random graph with a desired degree distribution. Generally, the distribution of the degree of the nodes in a random graph can be expressed as $p_k = \binom{N}{k}\ p^k(1-p)^{N-k}$ where $N$ is the number of vertices in the network, $k$ is the degree, and $p$ is the probability of forming an edge. As $N$ gets large, this distribution converges to $p_k = \frac{z^k e^{-z}}{k!}$, where $z$ is the average degree of the network.[21] This can be recognized as a Poisson distribution, which reflects the fact that the graph is constructed in such a way that the presence of one edge does not affect the presence of any of the others. Many properties of random graphs have been studied, including degree distribution, diameter, clustering coefficient, and graph spectra.[2][22]

### 3.3.2 Watts-Strogatz Small World Network

Random graphs often tend to possess small-world character, meaning that they have a relatively small average path length relative to their size. Based on this property, Watts and Strogatz (1998) defined a *small-world network* as one which has an average path length similar to that of a random network, but a relatively larger clustering coefficient.[29] Many real world networks have this property, and sometimes a network is characterized by its "small-worldness". This property can be captured using the *proximity ratio*, which is

the ratio of the average clustering coefficient divided by the average path length, to the average clustering coefficient of a random network of the same size and average degree divided by its average path length, or $\frac{\bar{C}/L}{\bar{C}_r/L_r}$.[22] Small world networks have many interesting properties that have been studied extensively, including the average path length, clustering coefficient, degree distribution, and spectral properties.

### 3.3.3 Scale-Free Networks

Often, real-world networks differ from random graphs in their degree distribution. Specifically, the degree distribution of many real networks follows what is known as a *power-law*, $P(k) \sim k^{-\gamma}$, where $\gamma$ is a constant greater than 1, and $k = 1, 2, \ldots, N$. What this means is that the probability of a node having a certain degree decreases proportionally relative to an increase in the degree. This relationship is independent of the scale, so the probability varies strictly as a power of the degree. For this reason, networks whose degree distribution follows a power law are often called scale-free networks.[2] Since random graphs do not have this degree distribution quality, one way to modify the construction of random graphs is by constructing a network with the degree distribution as an input, while all other aspects remain random (*i.e.* the edges still connect random nodes, but a constraint is imposed to ensure that the degree distribution follows a power-law). This process will be discussed in more detail in the following section.

# 4 Existing Models

## 4.1 Evolving Network Models

The work in this project concerns the design of an algorithm for an evolving network model, meaning that the network on which the infection spreads changes over time. There has been some work done on evolving networks in the past, and the most common approach taken is that of *rewiring*. This section will review a few pre-existing evolving network models, and illustrate some of the ways in which the evolving network model proposed in this paper differs from previous models.

### 4.1.1 Evolution via Addition, Removal, and Rewiring of Edges

While there is much to be learned from the study of static graphs, they are not always ideal for modeling real-world situations. For example, hyperlinks on the World Wide Web are constantly changing, as are roads between geographic locations and friendships in social networks. Traditionally, epidemic network modeling has assumed that the network on which the disease spreads is constant over the course of the epidemic. This is a reasonable assumption for highly infectious diseases that spread very quickly, because it is unlikely that the underlying social structure will change drastically during the course of the infection. However, one example in which a changing network might be useful is for modeling the spread of a disease over a longer period of time, during which the connections between people might change.

The most common approach to this problem takes an initial network and reshapes it through the addition and removal of edges, nodes or both. A rewiring event is one in which a node $i$ is randomly selected to have one of its edges removed, and then to have a new edge generated that connects it to another node in the network. One model that makes use of these network evolution strategies was proposed by Albert and Barabási in 2000.[2] Essentially, this algorithm begins with $N$ isolated nodes and no edges, and two different probabilities govern the evolution of the network: $p$, which is the probability that $m$ new edges are added, and $q$, the probability that $m$ existing edges are rewired. A new node can also be added to the network with probability $1 - p - q$, and when added it is given $m$ edges connecting it to existing nodes. At each timestep, any one of these three events can happen, and the network grows over time as a result.

Similarly, in a paper by Dorogvtsev and Mendes, a developing network model is proposed, where edges in the network can be either added or removed.[2] In this model, at each timestep, $c$ new edges are added to the network (where $c$ can be positive or negative, corresponding to a growing or shrinking network). The probability of a new edge connecting nodes $i$ and $j$ is directly proportional to the product of the degrees

of the two nodes, meaning that nodes with a higher degree are more likely to gain more edges. This is a phenomenon knows as *preferential attachment*, which is common in real-world networks, and which is a significant departure from the way in which Erdős-Rényi random graphs are generated.

Both of these models use the addition, removal and rewiring of edges to evolve networks over time. However, this approach is not ideal for the purpose of this project for a number of reasons. The type of evolution occurring in the algorithm proposed in this paper has a different nature than the "evolution" in the models described above. For our purposes, the goal of this evolving network model is to exploit the fact that a newly emerging or re-emerging disease starts with low prevalence in a population, allowing us to simulate disease progression on a small subset of the whole population by studying only infected individuals and their contacts. This allows us to save computation time by avoiding the need to model all of the nodes in the network at all times. This means that, like in the model proposed by Albert and Barabási, it makes sense to begin with a network of isolated nodes. However, rather than allow the addition, removal, or rewiring of edges to occur anywhere in the network, we will only initiate the addition of edges from nodes that are newly infected to nodes that are either not yet infected, or also newly infected. In this way, we will keep the edges of infected nodes static once they have been established. Furthermore, we will not allow for removal or rewiring of edges. Further details of the network evolution are discussed in Section 5.

## 4.2 Models with Tunable Parameters

In addition to evolution, the evolving network model discussed in this paper also incorporates the use of tunable parameters in generating a stochastic graph. The following sections will review a few previously proposed algorithms for generating random graphs with particular properties.

### 4.2.1 Random Graphs with Specific Degree Distributions

As introduced in the previous section, a random graph is one which is generated by forming arbitrary edges between vertices with a certain probability $p$, and which demonstrates a random distribution of the degrees of the nodes. Many graphs of real-world networks appear to have randomly distributed edges, so random graphs are heavily studied in hopes of better understanding real-world networks. However, real-world networks that possess some characteristics associated with random graphs often have degree distributions that are not Poisson. Modeling these networks with a traditional random graph, in which the only features that are controlled are the number of vertices and the probability of forming edges, would cause certain important features of these networks to be missed. In their 2001 paper, Newman, Strogatz, and Watts generalize the

statistical properties of random graphs to networks that demonstrate other types of degree distributions.[21] In theory, this model can be applied to a network with any arbitrary degree distribution, and thus is a very useful tool in modeling real-world networks.

One important feature of random graphs with arbitrary degree distributions is the fact that they are entirely random except for their degree distribution; meaning that the degrees of each of the vertices are independent and identically distributed (i.i.d.) random variables drawn from the distribution of choice. In order to determine the degree of each of the vertices given the desired degree distribution, this model relies on generating functions, which are functions that encode infinite sequences by treating them as the coefficients of a power series. In other words, for an infinite sequence $a_0, a_1, a_2, \ldots$ the value of the generating function at $x$ is $\sum_{i=0}^{\infty} a_i x^i$.[5] In particular, this model uses the generating function $G_0(x)$ to represent the probability distribution of vertex degrees $k$. Thus, this generating function can be defined as:

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k$$

where $p_k$ is the probability that a randomly chosen vertex has degree $k$. The reason that this is a helpful tool is because this probability is given by the $k^{th}$ derivative of the generating function evaluated at zero,

$$p_k = \frac{1}{k!} \frac{d^k}{dx^k} (G_0(x)) \bigg|_{x=0}$$

Thus, the generating function $G_0(x)$ for a degree distribution encodes all of the information contained in the probability distribution, so it is said to "generate" the degree distribution.[21] This type of generating function can be used to represent virtually any degree distribution that might be desired, including a distribution that follows a power-law or one that is exponentially distributed. It can also be used to find important characteristics of the network it represents, such as the average degree and the number of second neighbors of any vertex. These ideas are discussed at length in [21]. In order to generate a random graph with the desired degree distribution, the following procedure is carried out:

1. Generate a set $\{k_i\}$ of $N$ random numbers to represent the degrees of the $N$ vertices in the graph.

2. Use an appropriate transformation or rejection method, or a hybrid of the two, to make the set $\{k_i\}$ fit the desired distribution. (See [20] for more on transformations of sets of random numbers.)

3. If the sum of the elements in the set $\{k_i\}$ is odd, throw out the set and generate a new one. This is because each edge in a graph must have two ends, and thus the sum of the degrees of all of the nodes

in any graph must be equal.

4. Choose pairs of vertices at random, and create an edge joining the pair if both nodes in the pair still have fewer neighbors than their assigned degree.

This procedure generates all graphs with the given set of vertex degrees with equal probability.[21] Using this model, one can generate a graph with any desired degree distribution that can be summarized in a generating function. Theoretically, a similar process could be used to generate a graph with an arbitrary degree distribution that does not fit a known probability distribution, such as one measured from real data, where the only difference in the method would be that the degrees would be assigned using this distribution, rather than a distribution from a known model.

#### 4.2.2 ClustRNet

In a paper from 2009, Bansal, Khandelwal and Meyers take the model for generating random graphs with arbitrary degree distribution one step further through the introduction of an algorithm, ClustRNet, that generates a simple, undirected random graph with a specified degree sequence and level of clustering.[4] The authors explicitly state that the purpose of this algorithm is not to try to replicate the properties of real-world networks, but rather to generate networks as null models in order to enable a closer study of the effects of clustering on other network properties.

This algorithm begins by generating a random graph with a specified degree sequence using a method slightly different than that in [21], but achieving fundamentally the same result of a random graph with a desired degree distribution. Then, triangles are introduced into the graph through the rewiring of two edges at a time, in order to increase the level of clustering without altering the degree sequence of the graph.[4] This procedure uses a Markov Chain process, which in short is a process where the probability of moving between two states does not depend on what state the system was in before the current state.[11] The details of this process are beyond the scope of this paper, but in essence ClustRNet is an algorithm that takes an input degree sequence and clustering value, and produces an output of a random graph with the degree sequence and with a level of clustering that is greater than or equal to the input value (the level of clustering in the graphs produced is always at least equal to the input value because the algorithm utilizes a loop that continuously rewires edges while the level of clustering in the graph is less than the desired level). The algorithm ends when either the graph achieves at least the desired level of clustering, or when the number of unsuccessful rewiring attempts reaches a certain threshold.

While this algorithm *does* faithfully replicate the input degree distribution and clustering level, controlling clustering in this way can also influence other network properties. This paper specifically considers the effect of the algorithm on degree correlations (*i.e.* the relationship between one node's degree and the degrees of its neighbors) and average path length. While the introduction of clustering through the ClustRNet algorithm does not seem to establish degree correlations in the network, it does appear to cause a slight increase in the average path length. The authors of this paper theorize that this is because the increase in clustering causes the graph to become more separated into subgroups, which in turn means that the nodes in different groups are further away from each other.[4]

#### 4.2.3 Volz (2008)

In a paper from 2008, Volz describes an algorithm that produces a random graph with a specific degree distribution and level clustering by growing the network from a single node.[28] This model bears many similarities to the one proposed in this paper, however it also differs in a few non-trivial ways.

The inputs to this algorithm are the network size, desired level of clustering, and desired degree distribution. It begins by assigning each node in the network a degree drawn from the desired distribution (thereby ensuring that the final network will have the desired degree distribution), and then makes a list of all of the nodes which do not yet have neighbors (which, at the start of the simulation, is all of the nodes). A starting node is randomly chosen, and its neighbors are selected according to a certain probability. If these neighbors are not the same as the starting node and not already connected to the starting node, the connection is formed. Otherwise, the process is repeated until the starting node has enough neighbors to fulfill its assigned degree.

Finally, a list is created of all of the nodes that are two steps away from each other. Each pair of nodes in this list are connected to each other with a certain probability, $C$, that is determined by the input clustering parameter. This process is repeated until either all connections have been formed, or a connected component has been formed. If all connections have been formed the process ends; otherwise, a new starting node is chosen from those not in the connected component and the process occurs once again.[28] A schematic of the construction of the network is shown in Figure 4.1.

As will become evident shortly, the proposed evolving network model shares many features in common with the one just described. However, there are two important distinctions. One is the way in which the nodes are assigned neighbors. In our model, this is governed by the spread of the disease, where edges are only added between newly infected nodes and other nodes in the network, which evidently is not a component

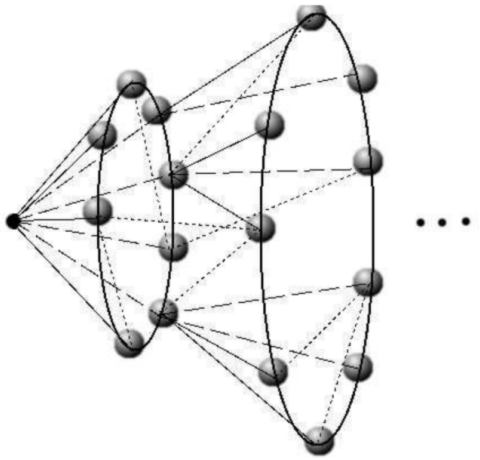


Figure 4.1: Schematic representing the construction of a network using the algorithm from [28]. The black node on the left is the randomly chosen starting node. The group of nodes in the smaller circle are then chosen as the starting node's neighbors. The larger circle of nodes are chosen as neighbors of nodes in the smaller circle, and some are connected to the starting node, two steps back from them, with probability $C$. This process is continued, forming new waves and creating new triads of nodes.[28]

of Volz's algorithm. The second significant departure is the way that the level of clustering in the network is controlled. In Volz's model, edges between the nodes are added in waves, so as to force the average clustering coefficient of the network generated to match the input parameter. However, this departs from the way in which the evolution of actual systems is governed. In our model, instead of having the average clustering coefficient as an input to the algorithm, there is a certain probability that friends of friends are chosen preferentially over other nodes in finding new neighbors. This concept and the motivation behind it are discussed more fully in Section 5.

## 4.3 Ebola Models

As mentioned in the introduction, an important application of the evolving network model proposed in this paper is in simulating and predicting the spread of newly emerging or reemerging infectious diseases. Specifically, this project was formulated with the West African Ebola outbreak in mind. In two papers from 2014 and 2016, a model for the 2014 Ebola outbreak is discussed.[17][16] This model will be briefly reviewed here.

The Ebola outbreak occurred in three different West African countries, Guinea, Sierra Leone, and Liberia, and in each country the dynamics of the epidemic were slightly different. One of the important features of Ebola is that it spreads easily between close contacts (such as family members), but has a relatively

low transmission probability between casual contacts. A model can take this into consideration by having two categories of contacts with different types of edges connecting them, and different transmission rates associated with each type.

Kiskowski's approach to modeling aims to identify the mechanisms behind the unique sub-exponential growth of the Ebola epidemic in West Africa, and combines a stochastic susceptible-exposed-infected-recovered (SEIR) model with a three-scale community network model. The SEIR model differs from the SIR model described in the introduction in that it includes a compartment for individuals who have been exposed to the disease but are not yet infected. This is an important consideration for Ebola because the disease has an incubation period of 2-21 days, and individuals cannot transmit the disease until they begin showing symptoms.[31] Furthermore, in this context, three-scale refers to the fact that the individuals in the network are organized into families, which are then organized into modular local communities that are subsets of the entire population. This structure is defined by the number of individuals per household, as well as the community radius, which specifies the number of households per community. In order to more accurately model the transmission of Ebola, the transmission probability for individuals within the same household is higher than that for those in different households within the same community. The households do not necessarily strictly represent nuclear families, but instead portray groups of any close contacts with a high transmission rate for Ebola, such as family members, medical caretakers, or funereal workers. Each household can be represented as a complete graph on $H$ vertices, where $H$ is the number of individuals in the household, while each community composes a complete graph on $C \cdot H$ vertices, where $C$ is the number of households in each community. The number of households in each community is related to the community radius, $r$, by $C = 2r + 1$. This general network structure is illustrated in Figure 4.2. Edges within each household are of a different type than those between households, and thus can carry different transmission probabilities.[17]

The simulation begins with all individuals in the susceptible state except for one infected individual. At each timestep, the infection progress is governed by three transition probabilities, $p(S \to E)$, $p(E \to I)$, and $p(I \to R)$, which depend on the two transmission rates, $t_H$ and $t_C$, the average incubation period, and the average infectious period. The model tracks the states of the individuals over time, and this infection progress was compared to the infection trajectory observed in the three West African countries. Using different input parameters (sizes of households and communities as well as transmission probabilities between the contact types) to model the outbreak in each country, best fits were found for modeling the number of cumulative Ebola cases over time in each country.[17] In [16], the effects of varying parameters on epidemic dynamics

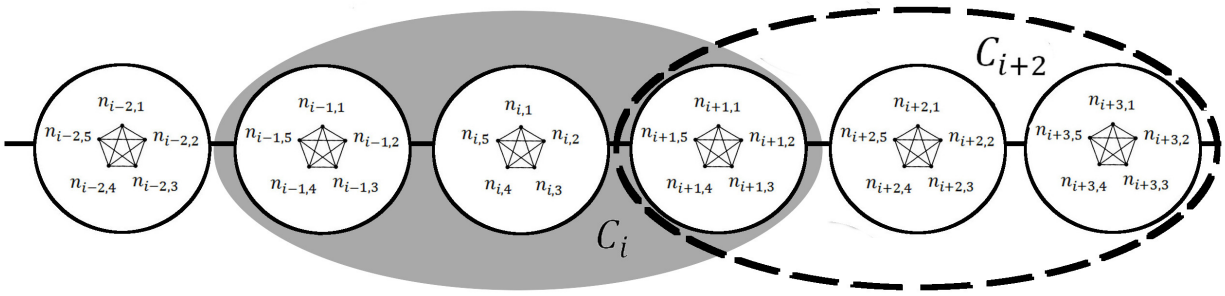



Figure 4.2: Example of a three-scale network model with household size 5 and community radius 1. Each household consists of a complete graph on 5 vertices, and each individual within household $i$ is connected to each individual in households $i-1$ and $i+1$, which lie within its community. Thus, each household is a part of tw distinct communities, and each node is connected to 14 other nodes. However, the edges within households are distinct from the edges between households, and thus have a different transmission probability associated with them.[17]

are explored more fully using the same model.

The model discussed in these papers bears some similarities to our evolving network model, mainly in the deviation from modeling the epidemic in a strictly compartmental or network-based way. Like our evolving network model, Kiskowski's model combines elements of a traditional compartmental model with a more detailed network model; however one major difference lies in the nature of the network. Kiskowski's model is simplified by the consideration of an extremely homogeneous network. Each node in the network has exactly the same number of neighbors of each type, and thus this model lacks some of the advantages associated with agent-based network modeling. In contrast, our evolving network model allows for a more complex network structure by assigning nodes a degree drawn from a distribution and then creating the appropriate edges based on those degrees. This increased complexity could improve the accuracy of disease trajectory predictions. However, our model does not currently accommodate different edge types in the way that Kiskowski's does, which is a serious limitation in modeling diseases such as Ebola that demonstrate significantly different transmission probabilities between different types of contacts. Future work could incorporate the feature into the evolving network model in order to improve its accuracy for applications such as this one.

# 5 Evolving Network Algorithm

## 5.1 Algorithm Overview

While we have thus far only given a vague description of the proposed evolving network algorithm that we developed, it is now time to discuss it in detail. This section may be skipped if a detailed understanding of the algorithm is not desired. There are three main sections of the code: initialization, construction of network, and determination/calculation of outputs.[2]

## 5.2 Initialization

The inputs to the algorithm are a population size ($n$), a degree distribution, a clustering probability ($c$), a transmission rate ($\beta$), and a recovery probability ($r$). The values of $c$, $\beta$, and $r$ must be between zero and one, and the degree distribution must be input as a cumulative probability distribution. For the model with no recovery, $r = 0$. In addition, there is a binary parameter $k$ that determines whether the algorithm will build the entire network or simply the largest connected component. A $k$ value of 0 turns off the secondary network construction process, while a value of 1 turns it on. For purposes of examining properties of the entire network, it is recommended to construct the entire network; however, for predicting the spread of disease assuming only a single initial infection, building the entire network is not necessary and increases computation time needlessly.

At the next step in the algorithm, the degrees of all of the nodes in the network are chosen. An $n \times 2$ matrix of all zeros is created to keep track of the degree assigned to each node. The first column is updated to contain the index of each node, and the second column is updated by drawing the degree of each node randomly from the cumulative distribution. Once fully updated, this matrix stays constant throughout the simulation. (Another version of the code was written in which this matrix could be altered under certain conditions in order to improve the precision of the clustering of the network without reducing the accuracy of the degree distribution. However, this modification did not show any apparent improvement over the original algorithm, so it will not be discussed in further detail.)

Next, the algorithm creates a number of matrices and arrays to keep track of various features of the network. The first is the adjacency matrix, which is initially an $n \times n$ matrix composed of all zeros. During the simulation, as connections are formed between nodes in the network, the adjacency matrix is updated accordingly. Each time that the $(i, j)$ entry is updated, so is the $(j, i)$ entry, and thus this matrix remains

[2] A version of the code is included in Appendix 9.2.

symmetric throughout the simulation. In addition, two empty arrays are created to keep track of the indices of nodes that are infected and newly infected (infected on the previous day), which are also updated at each time step. The Infection Progress array is also updated each day with the number of infected nodes on that day, which enables the infection progress to be tracked and plotted if desired. Finally, an $n \times 2$ matrix is created in order to monitor the number of infected contacts of each node (initially 0 but updated at each timestep), which will be used in calculating infection probabilities during the spread of the disease.

The remaining elements required in the setup of the algorithm are variables to keep track of the number of iterations for various processes, the number of days in the simulation, and the number of days since the last new case of the disease. These can all be used in determining an appropriate stopping condition in the simulation, as well as in examining details of the infection dynamics. In addition, the definition of a maximum number of iterations, days, or days without any new infections allows for the implementation of various termination criteria. Now that the simulation has been set up, it is time to begin the network evolution process.

## 5.3 Construction of Network

The network construction process begins with an initial random infection. The index of this node, $x$, is added to the arrays of infected and newly infected nodes. At the very beginning of the simulation, there will only be a single newly infected node; however, as time progresses, there are likely to be multiple new infections on a given day. For each newly infected node, the algorithm now builds that node's contact network. This is done by randomly picking contacts from the population, with certain restrictions. Suppose newly infected node $x$ has been assigned a degree of 3, and currently has one contact (the node from whom the infection was contracted). This means that the algorithm must find 2 new contacts for node $x$. In order to pick each of these contacts, a uniform random number between 0 and 1 is drawn. If this number is less than the clustering probability, $c$, then this means that the new contact should also be a mutual friend, or a neighbor of one of node $x$'s existing neighbors. By enforcing this feature in the network construction process, we can influence the number of triangles, and thereby the average clustering coefficient can be modulated as well. In most real contact networks, the likelihood of forming friendships (or having epidemiological contact) with a friend of a friend is higher than that of forming friendship with an individual unknown to your other contacts, and increasing the level of clustering in the network allows us to replicate this feature. A larger value of $c$ makes it more likely that the random number will be less than this parameter, and thus more likely that the neighbors chosen will lead to the formation of triangles. In this way, $c$ represents a clustering probability, or

the likelihood that nodes will form clusters with each other. The goal of using this probability to influence clustering, rather than strictly enforcing the formation of triangles by rewiring of edges as implemented in previous work[28], is to allow for a more organic network construction process. By using only this local information, the aim is to influence the global properties of the network in such a way as to successfully mimic properties of real social networks. However, a way to determine $c$ from real network data has not yet been found.

Regardless, if the uniform random number is less than the value of $c$, the algorithm picks a random node from the list of "friends of friends" of node $x$, *i.e.* a node that shares a common contact with node $x$. Assuming that this node is not already connected to node $x$, the algorithm makes the connection and moves on. However, if this node is already a neighbor of node $x$, or already has its full assigned degree and therefore cannot be given any new neighbors, then the process begins again, and a new random number is selected. If the random number chosen is larger than or equal to $c$, then rather than choosing a contact from the list of friends of friends of node $x$, the algorithm instead chooses a random node from the entire population. Before making the connection, it must once again ensure that the randomly chosen node is not (a) the same as node $x$, (b) already a neighbor of node $x$, or (c) already connected to the number of neighbors that it has been assigned. The first two requirements ensure that the graph generated is simple (does not contain loops or multiple edges), and the third ensures that the degree distribution of the network accurately represents that which was used as an input to the algorithm. If any of these requirements are not satisfied, then a new random number is drawn and the process is repeated again.

This contact selection process is repeated until enough eligible neighbors are chosen for node $x$ so as to give it its full assigned degree. In the case that the process is repeated many times and no eligible neighbors are found, a maximum number of iterations can be used to exit this loop. This generally only occurs at the very end of the simulation, when most nodes already have all of their assigned neighbors and thus there are very few left in the network who still need neighbors, or when the clustering probability is very high, but there are no eligible friends of friends. If the loop is exited before node $x$ is given the number of neighbors corresponding to its assigned degree, this could result in a slight decrease in the overall average degree of the network, which is a trend that was sometimes observed (see Section 6.4 for examples of this).

This process is completed for each newly infected node, and the adjacency matrix is updated to reflect the new edges that have been added to the network. Then, the list of newly infected nodes is reset, and the number of infected contacts of each node in the network is calculated in order to prepare for the spread of the infection. Each node in the network that is not already infected has a probability of becoming infected

of $1-(1-\beta)^N$ where $N$ is the number of infected contacts of that node. Thus, regardless of the transmission probability, if a node does not have any infected contacts, this probability is

$$1-(1-\beta)^0 = 1-1 = 0$$

so a node cannot become infected unless it has at least one infected contact. As $\beta$ increases, the quantity $(1-\beta)^N$ decreases, so the overall infection probability increases. Similarly, since the quantity $(1-\beta)$ is less than 1, as the number of infected contacts increases, $(1-\beta)^N$ decreases and thus this also causes the infection probability to increase. For each node in the network, if the node is not already infected, a uniform random number between 0 and 1 is drawn. If the infection probability is greater than this number, the node becomes infected and is added to the Infected and Newly Infected arrays. At this point in the simulation, the number of days between new infections is also recorded, in order to allow for another potential stopping condition in the simulation. If the simulation has been running for many days without any new cases of the disease, it is likely that the infection has terminated and will not spread any further, so the simulation may be stopped at this point.

When recovery is allowed for by the model, each infected node is given the same recovery probability, $r$. For each infected node, a random number between 0 and 1 is drawn. If this number is *less* than the recovery probability, then that node is removed from the list of infected nodes. This means that a higher recovery probability makes it more likely for nodes to recover. This is similar to the way in which compartmental models allow for the transition of nodes from the infected compartment to the recovered compartment. While the amount of time that a node has spent in the infected state does not directly influence its chances of recovery, the longer a node spends infected, the more random numbers that are drawn for that particular node, and thus over time it is likely to recover eventually. Furthermore, depending on what disease is being modeled, these recovered nodes may or may not be susceptible to infection once again. For this project these two scenarios were described as "SIS" and "SIR", respectively, because of the disease progression features that they share with these compartmental models. For the SIS version of the code, the recovered nodes are included as susceptible during the disease transmission process. In contrast, for the SIR version of the code, the algorithm does not allow the infection to spread to nodes that have already become infected and then recovered.

This process continues until one of the stopping criteria (number of iterations, number of days, number of days without any new infections, etc.) is reached, or until the entire population becomes infected or recovered.

If the generation of only one connected component is desired ($k = 0$), then the network generation process ends at this point. However, if one wishes to generate the entire network ($k = 1$), then a new random infection occurs, and this entire process is repeated over and over again until the entire population is infected and the structure of the entire network is revealed. Finally, after this whole process occurs, the algorithm calculates certain output parameters, which can be used in characterization of the network.

## 5.4 Outputs

After the simulation is finished, the first thing that the algorithm does is create a graph, $G$, from the adjacency matrix, $A$. This is done using a built-in MATLAB command and can be used to plot the graph so as to allow visualization of the network. Next, two of the most salient properties of the graph, the average degree and average clustering coefficient, are measured. The average degree is found simply by summing along each row of the adjacency matrix and dividing by the total number of nodes in the population, $n$. In the case where only the largest connected component is generated, it makes sense to find the average degree of only these nodes, which simply requires dividing by the number of nodes with at least one neighbor, rather than the total number of nodes. However, it is important to keep in mind that this would fail to consider the nodes in the network with degree zero, and thus would cause the measured average degree to increase slightly. To calculate the average clustering coefficient, a MATLAB function file written by Gregorio Alanis-Lobato is used, which calculates the fraction of closed triangles in the graph, or what we have been calling the average clustering coefficient.[1]

Other output metrics include the graph density, which is calculated using the formula described in Section 3, as well as the average path length and diameter. The latter two metrics are calculated using the built-in MATLAB command *distances* which calculates the geodesic distance between every pair of nodes in the network (further explanation of this process is also given in Section 3). The algorithm then has the capability report other information about the network, such as the number of connected components (using the MATLAB command *conncomp(G)*) and the spectrum of the graph. For the most part, the graph spectra were not investigated because calculating the spectrum of a large graph is extremely computationally intense and time consuming. The next section will explore the results obtained using this algorithm.

# 6 Results

## 6.1 Disease Progress

The main purpose of the proposed evolving network algorithm is to simulate the spread of infection on a social network in order to be of epidemiological use. Therefore before examining the properties of the networks generated by the algorithm, the disease dynamics will be briefly discussed and compared to the infection dynamics of traditional compartmental models. We considered three different models that can be accommodated by this algorithm: one in which individuals do not recover, one in which individuals recover and attain life-long immunity, and one in which individuals recover and become once again susceptible. Since the latter two cases are commonly modeled by SIR and SIS compartmental models, respectively, for the remainder of this chapter, these three cases will be referred to as "SI" (or No Recovery), "SIR", and "SIS" models for convenience. However, it is important to acknowledge that these models are *not* the same as the traditional compartmental SI, SIR, and SIS models; they are just similar in the way that individuals progress from one disease state to another. Furthermore, while these are the only three cases that were considered for this paper, the algorithm could be modified relatively easily to accommodate diseases that are usually modeled using more complex compartmental models, such as SIRS or SEIS, by simply incorporating additional disease states.

The two input parameters that affect the infection progress are the transmission probability, $\beta$, and the recovery probability, $r$. The transmission probability has to do with how easily a disease is transmitted from an infected individual to a susceptible individual, and is related to the likelihood that a susceptible node becomes infected by $1 - (1 - \beta)^N$ where $N$ is the number of infected contacts of that node. Thus, as $\beta$ increases, the rate at which the disease spreads through the network is expected to increase as well. An example of this is shown in Figure 6.1. The recovery probability affects how likely it is that infected nodes will recover, and thus as the recovery probability is increased, each infected node has a higher chance of recovering, and so the overall trend of the infection can change drastically. Figure 6.2 demonstrates an example of the effect of $r$ on disease dynamics. Generally, the infection dynamics are determined by the balance between transmission probability and recovery probability, as well as the specific network properties that constrain the spread of the infection.

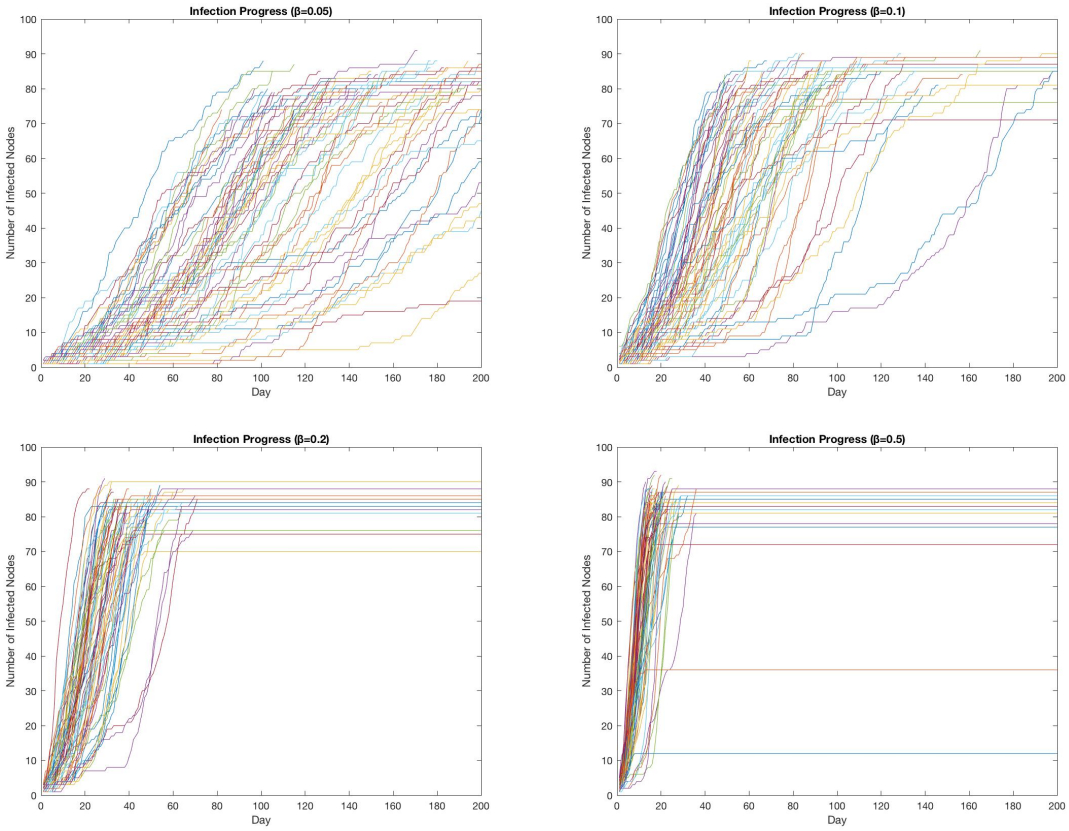



Figure 6.1: Comparison of infection progress for different transmission probabilities, with a recovery probability of 0 (*i.e.* no recovery). All four graphs represent 100 runs of the No Recovery simulation on a population of 100 nodes with a clustering probability of 0.2. Moving from left to right, top to bottom, the simulations have increasing transmission probabilities. From top left to bottom right, the transmission probabilities are 0.05, 0.1, 0.2, and 0.5.

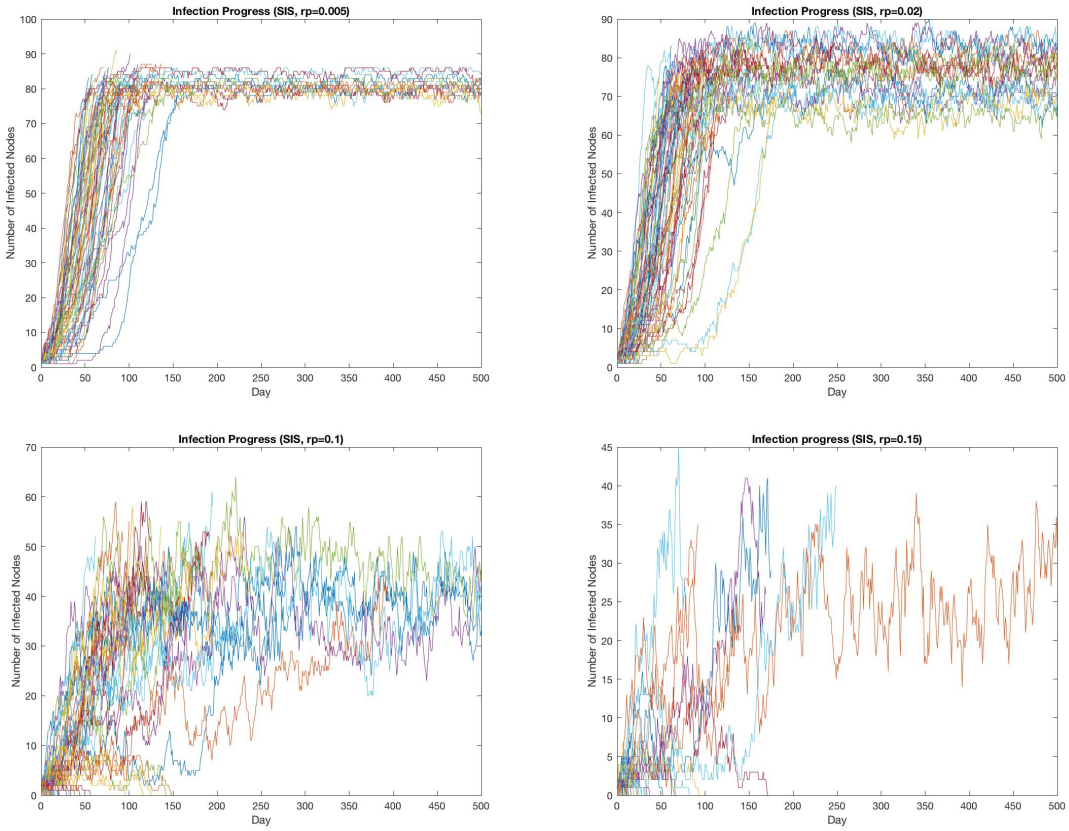



Figure 6.2: Comparison of infection progress in SIS model with different recovery probabilities. All four graphs represent 100 runs of the SIS simulation on a population of 100 nodes, with a clustering probability of 0.2 and a transmission rate of 0.1. Moving from left to right, top to bottom, the simulations have increasing recovery probabilities. From top left to bottom right, the recovery probabilities are 0.005, 0.02, 0.1, and 0.15.

When considering infection progress, there are two important measures to discuss. To begin with, it is important to look at the cumulative number of infections at each timestep in the simulation. In other words, this is the cumulative incidence of the disease over the time range spanning from the start of the simulation to the current timestep. In general, it was demonstrated that when using the same input parameters for each model, the cumulative number of infections was highest for the model with no recoveries, followed by the SIS model, and lowest for the SIR model. This is in agreement with patterns seen in comparing traditional compartmental SIR and SIS models, and can be explained by the fact that when a node recovers, it is no longer able to transmit the disease to other nodes in the network, and therefore the total number of infections in the network decreases. However, if nodes are able to become reinfected, this increases the chance that more transmission events will occur. Thus, it is to be expected that the average in cumulative number of

infections in the three different models would follow this trend. Some results are shown in Table 6.1 that demonstrate this pattern, and Figure 6.3 illustrates the same results. Using the same line of reasoning, it is easy to understand why given the same recovery probability, more cumulative recoveries are expected to occur in the SIS model than in the SIR model, since nodes are able to recover more than once.

Table 6.1: Average number of cumulative infections and cumulative recoveries over 100 runs of each model, excluding the runs where the total number of infections was less than 100, with population size 1000, clustering probability 0.2, transmission rate 0.1, and recovery probability 0.005.

| **Model** | **Cumulative Infections** | **Cumulative Recoveries** |
|---|---|---|
| No Recovery | 858.0 | 0 |
| SIR | 846.0 | 294.7 |
| SIS | 854.5 | 307.4 |

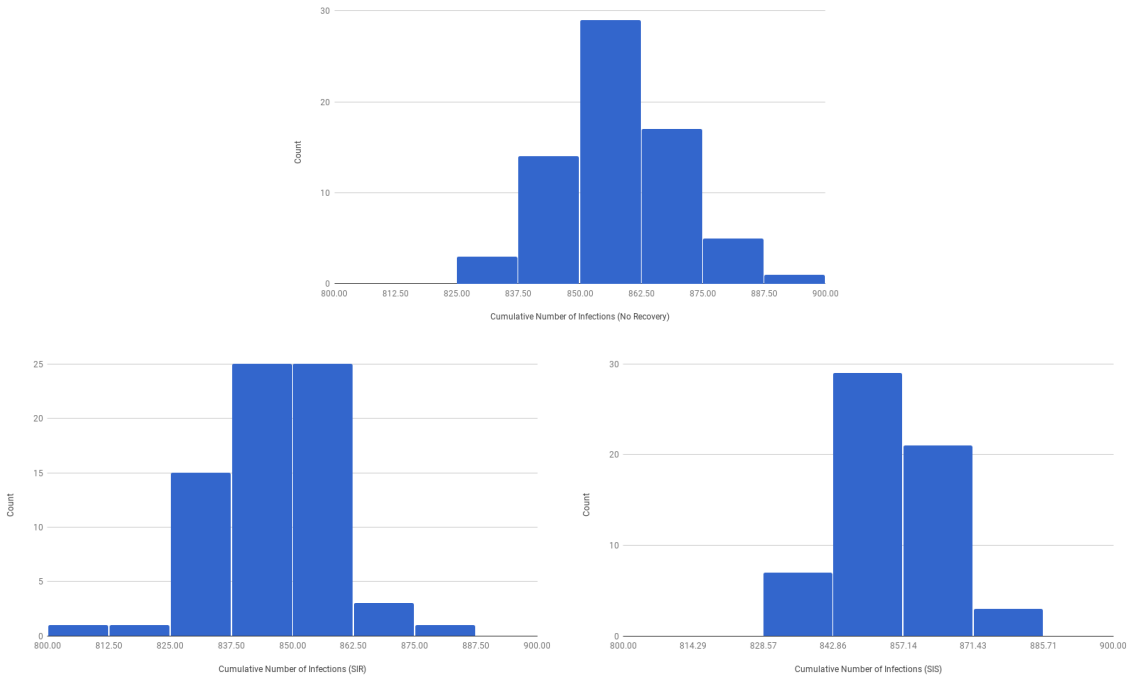



Figure 6.3: Histograms of the number of cumulative infections over 100 runs of each model, excluding the runs where the total number of infections was less than 100, with population size 1000, clustering probability 0.2, transmission rate 0.1, and recovery probability 0.005.

In addition to the cumulative number of infections, it is informative to consider the number of infectious nodes at each time step in each of the three models, or the prevalence of the disease. Figure 6.4 shows a comparison of the number of infected nodes at each timestep in each of the three models, using the same input parameters for each model. As is evident from the figure, after an initial period during which no

recovery occurs, on average there are the most infected nodes at any timestep in the model with no recovery. However, the rate at which nodes become infected is similar in all three models, with the highest number of infections occurring at approximately day 100. Table 6.2 presents some results that demonstrate this trend. In addition, using the same recovery probability of 0.005 for both the SIR and SIS models, it is clear that after a certain point, in most of the SIR runs, the number of infected individuals begins to decrease. In most of the SIS runs, by contrast, after a certain point the number of infected individuals begins to level off and reach an equilibrium, suggesting that the rate of infection is about equal to the rate of recovery.[3] However, this equilibrium is not always reached in SIS simulations. Figure 6.2 shows the way in which the recovery probability affects the infection progress. In the case where recovered nodes are once again susceptible, increasing the recover probability increases the variation in the infection progress for the different runs. Furthermore, once the recovery probability gets high enough, many of the runs end in complete recovery of the population rather than an equilibrium at a certain number of infected nodes.

Table 6.2: Average number of infected and recovered nodes at the end of the simulation over 100 runs of each model, excluding the runs where the total number of infections was less than 100, with population size 1000, clustering probability 0.2, transmission rate 0.1, and recovery probability 0.005.

| **Model** | **Number of Infected Nodes** | **Number of Recovered Nodes** |
|---|---|---|
| No Recovery | 858.0 | 0 |
| SIR | 367.3 | 478.7 |
| SIS | 833.1 | 21.8 |

In general, for a set recovery probability, the total number of recoveries was found to be higher for the SIS model than the SIR model when the simulations were run for long enough, which is a result of the fact that nodes can become reinfected in this model. However, for the same reason, the fraction of the population that is recovered, but not yet reinfected, at any timestep is larger for the SIR model than for the SIS model, since in the SIR model, once nodes become recovered, they cannot become susceptible or reinfected once again.

Finally, in order to fully assess the success of the algorithm in replicating the accuracy associated with agent-based network models, it is important to verify that the simultaneous network construction process does not interfere with the normal infection trajectory. In other words, we must demonstrate that the infection

[3]In the SIS differential equations model, this steady state can be predicted analytically. For a given infection probability, $\beta$, and recovery probability, $\gamma$, the number of infected individuals at equilibrium is given by $I = (1 - \frac{\gamma}{\beta})N$ where $N$ is the total population size.[25] For the set of simulations described here, this means that from a compartmental SIS simulation we would expect there to be $(1 - \frac{0.005}{0.1})1000 = 950$ infected individuals at this steady state. From Table 6.2 we can see that the number of infected individuals in the evolving network model is actually closer to 830. This discrepancy is due to the fact that the compartmental SIS model relies on the assumption of homogeneous mixing within the population, which is not assumed in the evolving network model.

progress on the evolving network appears to be the same as the more traditional method of allowing an infection spread on a full network. It was verified that this is the case, with the peak prevalence occurring at approximately the same day in the simulation, given the same input parameters (see Appendix 9.1).

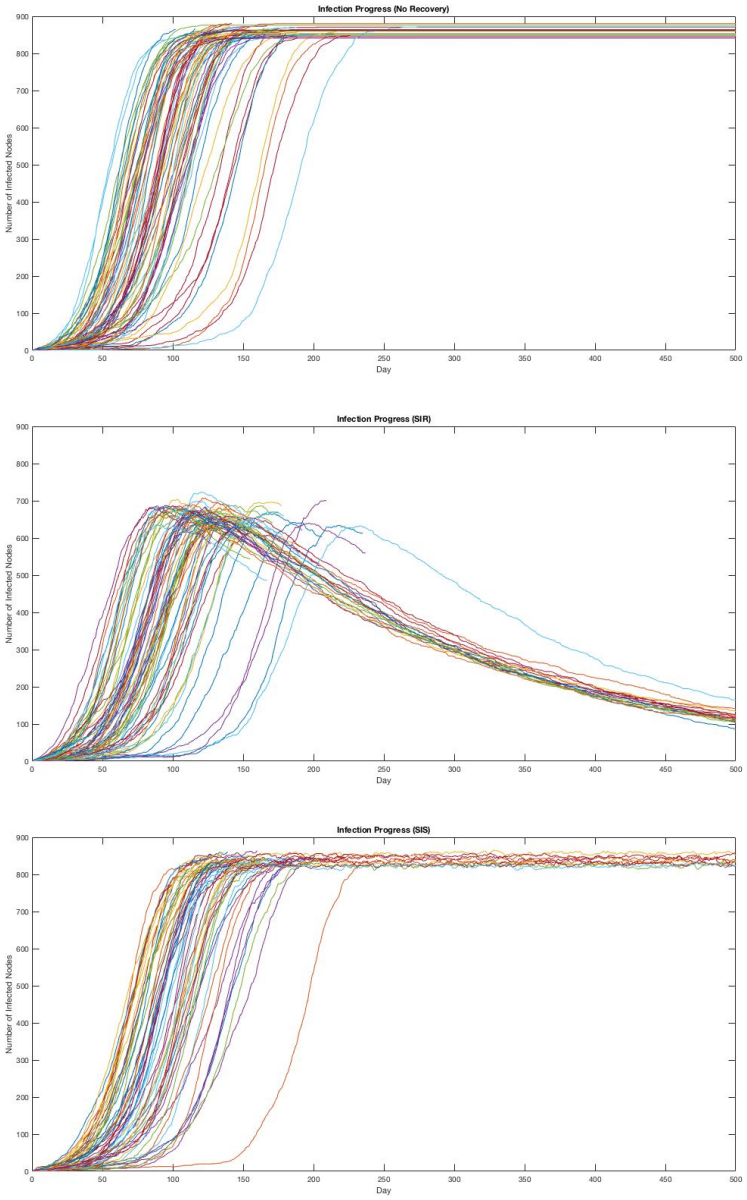



Figure 6.4: Plots of the infection progress in the first 500 days of each version of the simulation. Each plot represents 100 runs, where each line is the number of infectious individuals at each timestep in a single run. All of the simulations were run with a population of 1000, transmission probability of 0.1, and a clustering probability of 0.2. The models that include recovery were run with a recovery probability of 0.005.

## 6.2 Network Metrics

In this evolving network algorithm, the inputs that determine the network properties are a degree distribution and a clustering probability. As a result, in order to assess the accuracy and precision of the algorithm, it is important to examine the consistency of the average degree and average clustering coefficient of the generated networks across runs of the simulation, in addition to how well these metrics match the input parameters. Furthermore, the three models (SI, SIS, and SIR) differ only in their infection properties, so one would also expect the properties of the generated networks to be roughly the same on average for the three models, given the same input parameters.

Because there is an element of stochasticity in the network generation process, it is unlikely that the networks generated by any two runs will be exactly the same, even if they have the same input parameters. Therefore, in considering the consistency of average degree and average clustering coefficient, it will be important to look at the mean and standard deviation in these properties over a number of runs. Furthermore, we are also interested in assessing the ability of the algorithm to match other network properties. We will specifically consider graph density, average path length, and diameter, as these are some commonly considered graph metrics.

Additionally, before discussing these specific properties, there is one more element of the algorithm to consider. As a result of the nature of the network generation process, the default version of this algorithm only generates a single connected component of a graph. This is because disease cannot spread to disconnected components, and thus for a single outbreak, an infection is always confined to a single connected component. As we are mainly interested in modeling the contact networks of infected nodes, it is not necessary to generate the other components of the graph. Therefore when using this algorithm to generate graphs, some of the runs create small connected components within which the infection is contained. For the purposes of measuring graph metrics, these small components are likely to have properties that differ from those in larger components, simply because they contain only a handful of nodes and therefore are unlikely to accurately represent the degree distribution. Thus, in taking averages of graph metrics over many runs, the small connected components (in which the number of nodes is less than 10% of the entire population) have been excluded. However, we *are* interested in the proportion of runs that generate small connected components, as this could be related to the fraction of nodes that are in the largest connected component in a target network. Thus, after discussing each of the network metrics of interest, we will also discuss the fraction of runs that find small connected components rather than the large connected component, as well as the size

of the large connected component when it is found.

When the algorithm does not find the large connected component, the rest of the network can still be uncovered. This is achieved by initiating a new random infection elsewhere in the network. While this is not necessarily representative of the way in which epidemics spread in real networks, it can be repeated until the entire network, including isolated nodes, becomes infected, thereby uncovering all of the nodes in the network. This approach is more useful is certain circumstances because it reveals properties of the entire network, rather than just of the largest connected component. However, many network properties make more sense in the context of a single component rather than the entire network. For example, the a diameter of a network that has more than one component is technically infinite, but the diameter of the largest connected component is a more useful metric. Therefore this section will mainly discuss the properties of the largest connected component of the networks generated.

### 6.2.1 Average Degree and Average Clustering Coefficient

As previously mentioned, the algorithm takes a degree distribution as an input, and uses this distribution to assign the degree of each node in the network. As a result, one would expect the networks constructed to have an average degree that is close to the average of the distribution. The degree distribution used for the runs in this section has a mean of 2.39, which includes all of the nodes with degree zero. Table 6.3 shows the results of the average degree and average clustering coefficient of the largest connected component over 100 runs of each model with a population of 1000 nodes (excluding runs where fewer than 100 infections occurred). Clearly, the average degree is very close for the networks generated by each of the three models. However, this value of approximately 2.78 is larger than the average of the input degree distribution. This can be accounted for by the fact that the networks generated by the algorithm do not include nodes of degree zero, since they only generate the largest connected component. Thus, if the distribution is adjusted to exclude degree zero, but keep the relative probabilities of all of the other possible degrees the same, the average of this new degree distribution is 2.779, which is very close to the calculated average of the average degrees of the generated networks.

Table 6.3: Mean average degree and average clustering coefficient for the largest connected component in a population of 1000 nodes over 100 runs of each simulation, excluding those in which the network size was less than 100. (This includes 69 networks for No Recovery, 71 for SIR, and 60 for SIS). For these runs, the transmission probability was 0.1, recovery probability 0.005, and clustering probability 0.2.

| **Model** | **Average Degree** | **Average Clustering Coefficient** |
|---|---|---|
| No Recovery | $2.784 \pm 0.05980$ | $0.1497 \pm 0.01249$ |
| SIR | $2.777 \pm 0.04919$ | $0.1515 \pm 0.01330$ |
| SIS | $2.777 \pm 0.05280$ | $0.1581 \pm 0.01368$ |

Unlike other algorithms for generating networks with a specific degree distribution and level of clustering (see [28]), this algorithm does not take a specific average clustering coefficient as an input. Instead, it uses a "clustering probability" in the construction of the network, which is related to the likelihood that nodes are friends with their neighbors' friends. The relationship between the clustering probability and average clustering coefficient will be discussed further in Section 6.3. For now, the consistency of the measured average clustering coefficient of the constructed networks will be considered. Table 6.3 shows the average clustering coefficient results for a clustering probability of 0.2 in each model.[4] While the 100 runs considered in this table result in similar average clustering coefficients in all three models, approximately 0.15, from this table it appears that the average clustering coefficient is, on average, slightly higher in the networks generated by the SIS model than the SIR model, and slightly higher for the SIR model than for the model with no recovery. Given the standard deviations, however, these differences can be considered unimportant, and in fact other runs of the simulations have not shown the same trend. Thus, it can be concluded that these differences are due to the stochasticity in the models and the relatively small sample size of 100 runs of each simulation, rather than true differences in the properties of the networks generated by each model. Figure 6.5 shows histograms of the average degree and average clustering coefficient of the networks generated by each model, and demonstrates the relatively small spread in each metric. This suggests that despite the stochasticity in the network construction process, the average degree and average clustering coefficient of the networks generated can be controlled with a great deal of precision.

The average degree and average clustering coefficient of the full networks, rather than just the largest connected component, can also be calculated by allowing the infection to spread to the entire population. In this case, using the model with no recovery, the average degree over 100 runs using the same input parameters was 2.383, which is very close to the average of the degree distribution. The average clustering coefficient was 0.143.

[4]While the clustering probability does have an effect on the average clustering coefficient of the network, these two parameters are not the same. Their relationship will be discussed further in Section 6.3.

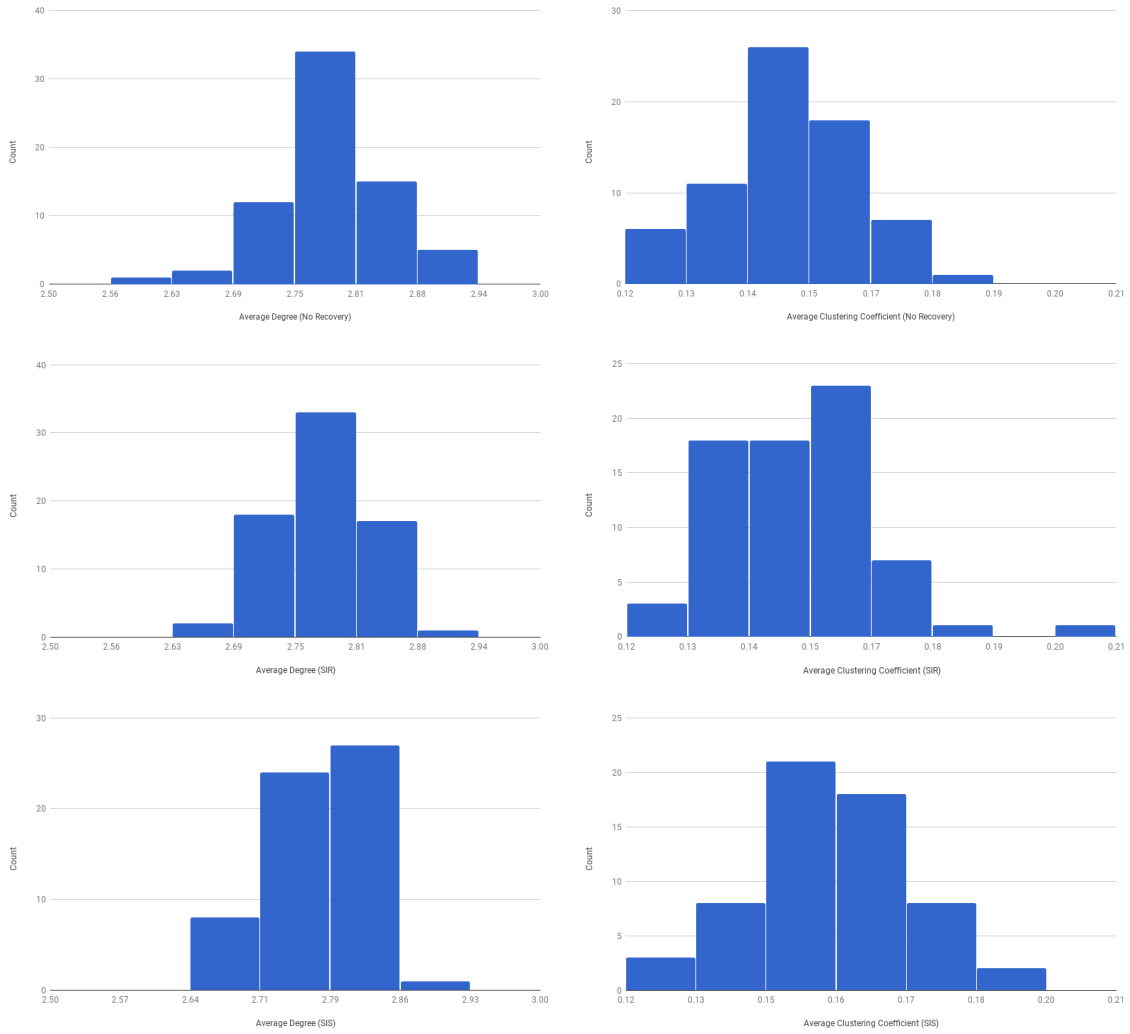



Figure 6.5: Histograms of the average degree and average clustering coefficient of the largest connected components of networks with a total population of 1000, generated by 100 runs of each model. Averages and standard deviations of the average degree and clustering coefficient for these networks are shown in Table 6.3.

While the degrees of the nodes and the level of clustering in the networks are somewhat explicitly entered into the algorithm as inputs, and thus are the most obvious parameters to consider, other graph metrics can be extracted from the networks that are generated as well. The next few sections will consider the consistency of a few other network properties across runs of the simulations.

### 6.2.2 Graph Density

As discussed previously, graph density is a measure of the number of edges in the graph relative to the maximal possible number of edges. This metric can range in value from 0 to 1, where a higher value describes a more dense graph. While graph density is not an explicit input to the algorithm, it is closely related to the degree distribution and thus would be expected to be relatively consistent across the different networks generated by the simulation. Table 6.4 shows the average of the density of the largest component of 100 networks generated by each model (excluding the runs where the total number of nodes in the network was fewer than 100). For the particular degree distribution and clustering probability used to generate these networks, the density was found to be about 0.0024 for all three models. As with the average degree and average clustering coefficient, the slight differences between the models is likely due to the random variation in the network construction process, and overall the density is consistent across all three models. Histograms of the density of the networks generated by each model are shown in Figure 6.6, in which the density has been multiplied by $10^3$ in order to better illustrate the variation across runs. However, both the standard deviations in Table 6.4 and the narrow spread of the histograms demonstrate the relative consistency of the density in the different networks.

Table 6.4: Average density for a the largest connected component in a population of 1000 nodes over 100 runs of each simulation, excluding those in which the network size was less than 100. (This includes 69 networks for No Recovery, 71 for SIR, and 60 for SIS). For these runs, the same degree distribution was used as earlier in this section, and the transmission probability was 0.1, recovery probability 0.005, and clustering probability 0.2.

| **Model** | **Graph Density** |
|---|---|
| No Recovery | $0.002400 \pm 0.00006181$ |
| SIR | $0.002389 \pm 0.00005194$ |
| SIS | $0.002384 \pm 0.00005305$ |

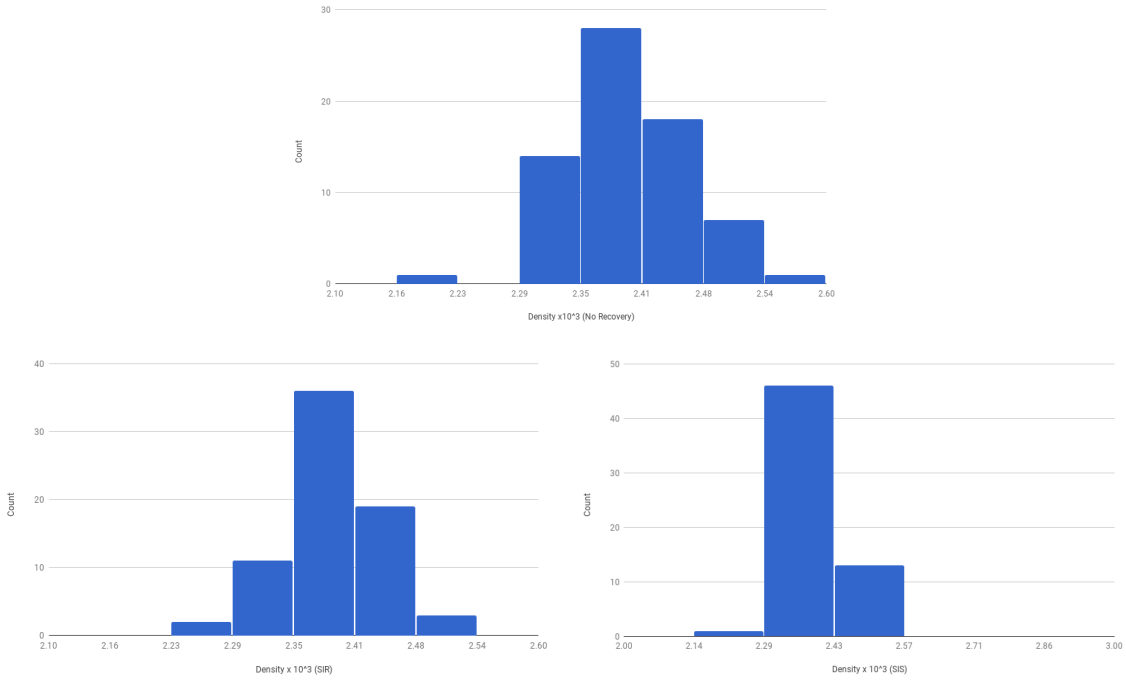



Figure 6.6: Histograms of density of the networks generated by 100 runs of each model, multiplied by $10^3$. Averages and standard deviations of the density of these networks are shown in Table 6.4.

### 6.2.3 Average Path Length and Diameter

Unlike density, average path length and diameter are not as closely related to the degree distribution of a graph. As a result, slightly more variation in these properties is expected between different networks that are generated with the same input parameters. However, as seen in Table 6.5, these properties were still very consistent across the different models. For the specific degree distribution and a clustering probability of 0.2, the average path length of the largest connected component over 100 runs of each simulation (excluding those in which the network generated had fewer than 100 nodes) was approximately 8.2, and the diameter was about 19. The variation between runs *was* a bit larger for these metrics than for those discussed previously, but the average path length was between 7.5 and 9.5 for all generated networks with more than 100 nodes, and the diameter was between 15 and 28. The histograms in Figure 6.7 depict the distribution of average path length and diameter across the networks generated using each model.

Table 6.5: Mean average path length and diameter of the largest connected component for a population of 1000 nodes over 100 runs of each simulation, excluding those in which the network size was less than 100. (This includes 69 networks for No Recovery, 71 for SIR, and 60 for SIS). For these runs, the transmission probability was 0.1, recovery probability 0.005, and clustering probability 0.2.

| **Model** | **Average Path Length** | **Diameter** |
|---|---|---|
| No Recovery | $8.216 \pm 0.3192$ | $19.30 \pm 2.171$ |
| SIR | $8.220 \pm 0.3112$ | $19.37 \pm 2.133$ |
| SIS | $8.350 \pm 0.3571$ | $19.98 \pm 2.332$ |

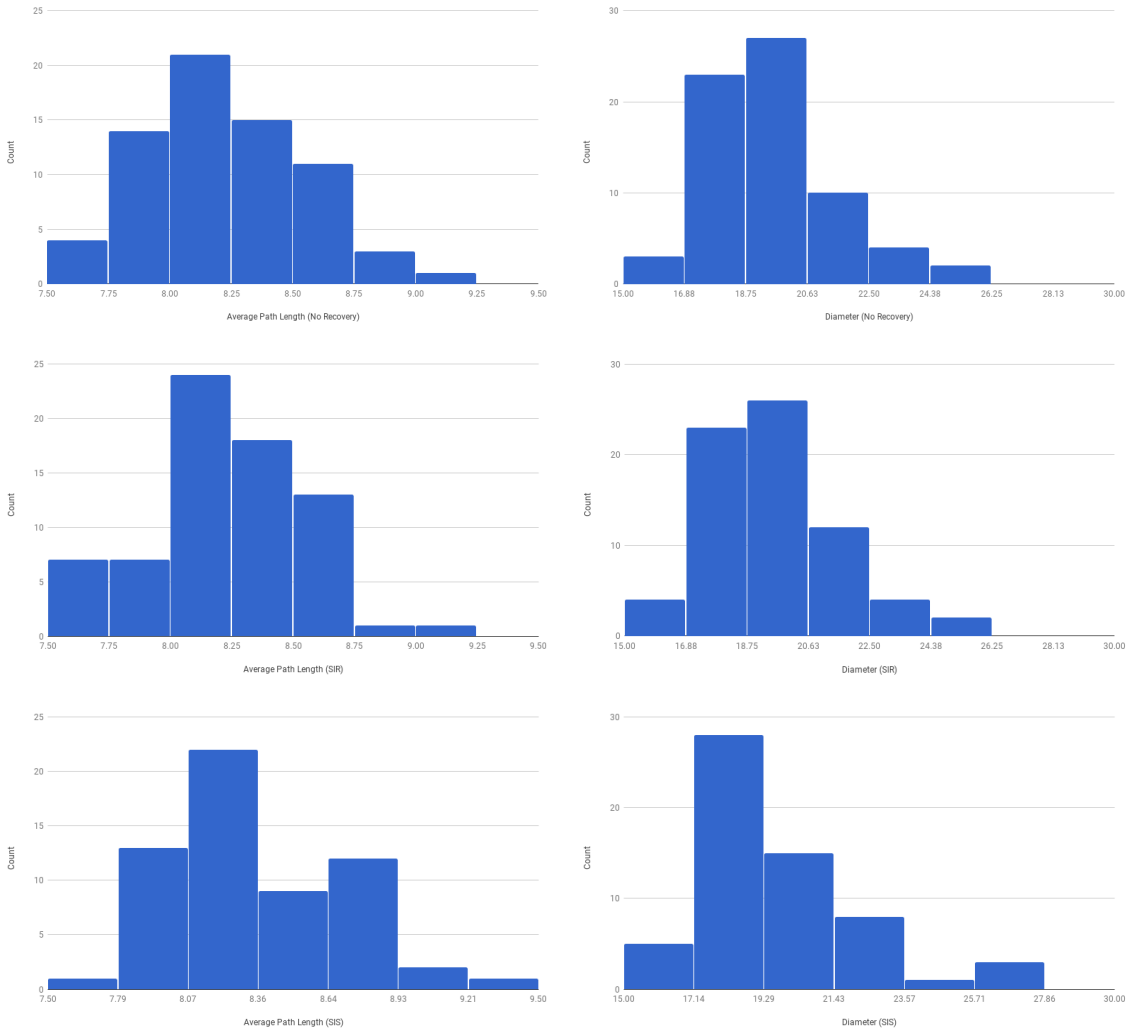



Figure 6.7: Histograms of the average path length and diameter of the largest connected component of networks generated by 100 runs of each model. Averages and standard deviations of these metrics for the networks are shown in Table 6.5.

### 6.2.4 Connected Components

In addition to the network metrics discussed, another important network property is the size of the largest connected component. As already explained, each run of this algorithm only generates a single connected component. Since the larger connected components are more representative of the network properties than smaller components, this section will focus on the runs that generate a "large" connected component, meaning a connected component that contains more than 10% of the nodes in the network. The size of the connected component is related to, but not exactly the same as the number of infected nodes. For the model with No Recovery, all of the nodes in the connected component will eventually become infected, and thus the number of infected nodes and size of the connected component are almost the same at the end of the simulation. There is, however, a stopping condition that would cause the simulation to end before all of the nodes become infected, so there may be slightly fewer infected nodes than the total number of nodes in the network.

For the models that do allow for recovery, clearly the number of infected nodes cannot be used to measure the size of the connected component. As already discussed, the number of infected nodes *and* the cumulative number of infections are generally larger in the model with no recovery than in either of the other models. However, in Table 6.6 it can be seen that the size of the large connected component does not notably differ between models. This means that the infection dynamics are not affecting this particular network property. Furthermore, the standard deviation in the size of the large connected component is relatively small for all three models. This suggests that although the size of the large connected component is not an input to the algorithm, the networks generated are relatively consistent with respect to this property. Figure 6.8 illustrates the distribution in the size of the large connected component across 100 runs of each model.

Table 6.6: Average size of large connected component for a population of 1000 nodes over 100 runs of each simulation, excluding those in which the network size was less than 100. The "frequency" refers to the percentage of runs that generated a large connected component (containing at least 10% of the nodes in the network), rather than a smaller component.

| **Model** | **Frequency** | **Average Size of Connected Component** |
|---|---|---|
| No Recovery | 73% | $860.45 \pm 10.26$ |
| SIR | 67% | $859.03 \pm 11.62$ |
| SIS | 66% | $858.53 \pm 11.95$ |

Finally, Table 6.6 also includes the frequency of the large connected component. This number refers to the percentage of runs that find a large connected component, as opposed to a smaller sub-network. For all three models using this particular degree distribution, this frequency was found to be about 70%. When the code is run with multiple infections in order to generate the entire network, often the largest

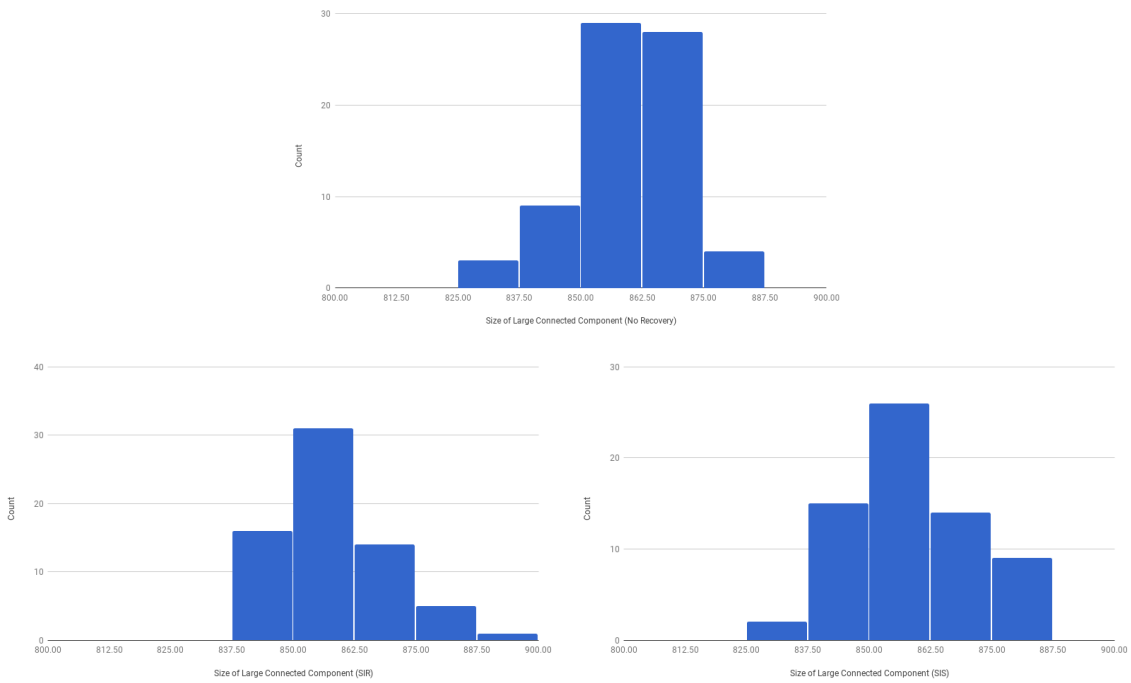



Figure 6.8: Histograms of the size of the large connected component generated by 100 runs of each model (excluding runs where this component included fewer than 100 nodes). Averages and standard deviations of the density of these networks are shown in Table 6.6.

component is the only component that contains more than one node, or in other words the network that is generated is composed of one large component and then a number of isolated nodes. Sometimes, a second, smaller connected component is found, and occasionally even more are discovered. The number of connected components in the network is dependent on the degree distribution used, but in general the algorithm tends to favor the formation of one giant component over multiple smaller components. The reason for this is not fully understood, but is likely due to the algorithm's efforts to match the input clustering probability. However, in many cases this network structure (one large component with relatively few smaller components and isolated nodes) is representative of real social networks, which will be discussed further in Section 6.4. For the degree distribution used in this section, the average number of connected components over the 100 networks generated was $1.32 \pm 0.53$ (excluding isolated nodes), reflecting that while 71% of the networks had only one connected component, 26% had 2 components, and 3% had 3.

## 6.3 Effects of Input Parameters on Network Properties

Now that it has been demonstrated that the models with different infection dynamics lead to the generation of networks with essentially the same properties, it is clear that two inputs that *do* play a role in determining the network properties are the degree distribution and clustering probability. The degree distribution can have many possible forms, including but not limited to common distributions such as Poisson or power-law. In addition, the degree distribution can be drawn from real-world data in order to closely replicate the properties of an actual social network. Section 6.4 will examine the use of actual datasets in choosing the degree distribution for the algorithm. The clustering probability is a value between 0 and 1 that roughly describes the likelihood that, when assigning a node's friends, they are friends of that node's other friends. This property is positively correlated with the average clustering coefficient, as a higher clustering probability makes the presence of triangles in the network more likely, and thus leads to a higher average clustering coefficient. However, the relationship between these two parameters is not straightforward. In this section, the effect of clustering probability on the properties of the networks generated will be considered, including average clustering coefficient, average degree, average path length, diameter, density, and number of connected components. In addition, we will discuss the effect of changing population size on the average clustering coefficient of and number of connected components in the generated networks. All of the simulations considered in the rest of this section were run using the model with no recovery. In addition, the properties of the entire networks (rather than just the largest connected component) will be considered.

To begin with, Table 6.7 illustrates the effects of clustering probability and population size on the average

clustering coefficient of the network. It is clear that as clustering probability ($c$) increases, the average clustering coefficient also increases. However, this relationship does not appear to be linear and, furthermore, there is a lot of variation in the average clustering coefficient of networks for a given clustering probability. For example, the average over 100 runs of the average clustering coefficient for a $c$ of 0.4 and a population size of 100 was 0.275, but the standard deviation was 0.062. This standard deviation is more than 10% of the average, which suggests that there is a substantial amount of variation around the mean. As the population size increases, however, the variation does decrease. In addition, as the population size increases for a given clustering probability, in some cases the average clustering coefficient of the network seems to decrease slightly. This decrease is small, and in some cases entirely absent, so it could be do to random variation; however this trend was observed consistently, and therefore could pose an obstacle in trying to generate networks whose average clustering coefficient matches that of a large target network.

In order to address this issue, as well as to try to generate networks with a higher average clustering coefficient, a different version of the algorithm was written in which the clustering probability is more strictly enforced. This was achieved by necessarily giving a node a neighbor that is a friend of a friend when indicated by the clustering probability (see Section 5), even if doing so requires changing the assigned degree of some of the nodes. (In this case, the assigned degree of such a node is exchanged with the assigned degree of another node in the network in order to avoid skewing the degree distribution). However, even with this modification, the effects of clustering probability and population size on average clustering coefficient were indistinguishable from those presented in Table 6.7, and so this other version of the algorithm was not considered to be an improvement. Nevertheless, it is clear that increasing the clustering probability has the desired effect on average clustering coefficient, and we will see in the following sections that this feature can be used to match the network properties of many target networks.

Table 6.7: Relationship between clustering probability ($c$) and average clustering coefficient. The left column represents the population size, and the other columns are the average and standard deviation over 100 runs of the network average clustering coefficient for different values of $c$.

| **n** | **c=0.2** | **c=0.4** | **c=0.6** | **c=0.8** |
|---|---|---|---|---|
| 100 | $0.191 \pm 0.056$ | $0.275 \pm 0.062$ | $0.364 \pm 0.067$ | $0.470 \pm 0.053$ |
| 500 | $0.147 \pm 0.021$ | $0.253 \pm 0.025$ | $0.354 \pm 0.030$ | $0.465 \pm 0.023$ |
| 1000 | $0.140 \pm 0.016$ | $0.247 \pm 0.019$ | $0.354 \pm 0.024$ | $0.465 \pm 0.021$ |

Table 6.8 shows the effects of $c$ and population size on average degree. Since the degrees of the nodes are assigned prior to the network construction process according to the degree distribution, the clustering probability should not affect the average degree of the networks. The results in Table 6.8 suggest that this

is in fact the case. The degree distribution used for these simulations has an average of 2.39, which is very close to the average of this metric over 100 runs for each set of input parameters. There is slight variation in some of the averages, but consistent pattern is apparent, and all of the values fall within one standard deviation of each other. In addition, it is evident that the standard deviation in average degree over the 100 runs decreases as the population size increases. This is simply because a larger population size requires that more random numbers be drawn from the degree distribution in order to assign the degree of all of the nodes. By the Law of Large Numbers, this means that the average of these degrees will be closer to the true mean of the distribution, and therefore these averages will be closer to each other, leading to a smaller standard deviation. The proximity of all of these measurements to one another suggests that changing the clustering probability and population size does not affect the average degree of the generated networks.

Table 6.8: Relationship between clustering probability ($c$) and average degree. The left column represents the population size, and the other columns are the average and standard deviation over 100 runs of the network average degree for different values of $c$.

| **n** | **c=0.2** | **c=0.4** | **c=0.6** | **c=0.8** |
|---|---|---|---|---|
| 100 | $2.354 \pm 0.192$ | $2.356 \pm 0.204$ | $2.350 \pm 0.175$ | $2.370 \pm 0.175$ |
| 500 | $2.385 \pm 0.081$ | $2.381 \pm 0.078$ | $2.376 \pm 0.081$ | $2.408 \pm 0.076$ |
| 1000 | $2.385 \pm 0.052$ | $2.383 \pm 0.063$ | $2.384 \pm 0.057$ | $2.380 \pm 0.052$ |

Table 6.9 shows the effect of $c$ on the average path length, diameter, and density of the networks. It appears clear that changing $c$ has very little effect on graph density. This makes sense because density is a measure of the number of edges and thus is directly related to the average degree of the network; so if the average degree does not change with changing $c$, the density would not be expected to either. By contrast, both the average path length and diameter appear to increase as $c$ increases. This is consistent with results found in [4], where the introduction of clustering (*i.e.* higher proportion of triangles) led to an increase in average path length of the generated networks. According to the authors of this paper, this is likely because the increase in clustering causes the network to become divided into more separate subgroups, such that the nodes in different subgroups require more edges to travel between them. In addition, as would be expected, increasing the population size causes a decrease in the density of the network, as well as an increase in the average path length and diameter. The decrease in density is directly proportional to the increase in population size (e.g. doubling the population size causes the density to be halved). The positive correlation between population size and average path length/diameter in random graphs has been well documented, and thus these results are consistent with previous results from random graph theory.[4][21]

Table 6.9: Effect of clustering probability ($c$) on average path length, diameter, and density for a population size of $n = 1000$. Each value represents an average and standard deviation over 100 runs of the network.

| $c$ | **Average Path Length** | **Diameter** | **Density** |
|---|---|---|---|
| 0.2 | $8.35 \pm 0.35$ | $21.32 \pm 2.40$ | $0.00239 \pm 0.00005$ |
| 0.4 | $9.91 \pm 0.69$ | $27.84 \pm 3.55$ | $0.00239 \pm 0.00006$ |
| 0.6 | $13.49 \pm 2.12$ | $40.72 \pm 8.58$ | $0.00239 \pm 0.00006$ |
| 0.8 | $20.67 \pm 4.79$ | $61.45 \pm 15.15$ | $0.00238 \pm 0.00005$ |

Finally, Table 6.10 demonstrates the effect of clustering probability and population size on the number of connected components in the graph. In general, increasing the clustering probability increases the number of components, on average, in the generated networks. The reason for this is likely related to the explanation given for the effect of clustering on average path length. As the clustering probability increases, the nodes in the networks begin to form more, smaller subnetworks with high internal clustering. During the network construction process, if all of the nodes in one of these sub-networks have formed connections to the number of neighbors that they have been assigned, then no new connections can be built and the infection will terminate in that smaller component. Therefore, in the evolution of networks with high levels of clustering, multiple smaller components are formed, rather than one large component. Two examples of this are shown in Figure 6.9, where the clustering probability used was 0.8. Furthermore, increasing the population size appears to increase the number of connected components as well.

Table 6.10: Relationship between clustering probability ($c$) and number of connected components. The left column represents the population size, and the other columns are the average and standard deviation over 100 runs of the number of components in the network for different values of $c$.

| **n** | **c=0.2** | **c=0.4** | **c=0.6** | **c=0.8** |
|---|---|---|---|---|
| 100 | $1.28 \pm 0.60$ | $1.75 \pm 0.78$ | $2.19 \pm 1.33$ | $3.61 \pm 1.86$ |
| 500 | $1.34 \pm 0.59$ | $1.81 \pm 1.06$ | $2.75 \pm 1.78$ | $7.67 \pm 3.29$ |
| 1000 | $1.39 \pm 0.65$ | $1.71 \pm 1.00$ | $3.20 \pm 1.75$ | $13.96 \pm 5.99$ |

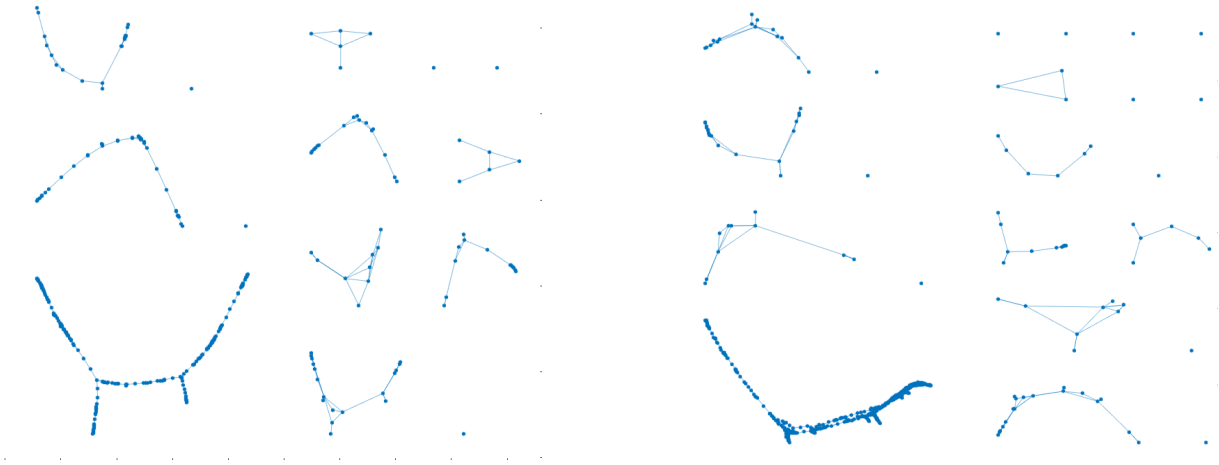


Figure 6.9: Two examples of networks generated on 1000 nodes using a clustering probability of 0.8 (excluding some of the isolated nodes). The network on the left has 9 connected components, and the one on the right has 10.

## 6.4 Application to Real Datasets

In this section, the application of the algorithm to five different open-source datasets will be discussed. These datasets will be referred to by the names shown in the table below.

| **Dataset Name** | **Source** |
|---|---|
| Norwegian Boards | Seierstad, C., and Opsahl, T. (2011) [24] |
| Journal Authors | Blagus, N., Bajec, M. (2015) [6] |
| Friendship Network | Mastrandrea, R. et al. (2015) [18] |
| Wiki-Vote | Rossi, R., Ahmed, N. Network Repository. (2013) [23] |
| Infectious Network | Isella, L. et al (2011) [14] |

In each case, the dataset is considered as a "target network" for the simulations, and the ability of the simulations to match the properties of the target network is assessed. In order to achieve this, the degree distribution was extracted from each dataset and used as an input to the algorithm. To determine the necessary clustering probability in order to best match the properties of the target network, different clustering probabilities were tested until one was found that generated a network with an average clustering coefficient that was very close to the average clustering coefficient of the target network. However, as was already demonstrated, the effect of $c$ on average clustering coefficient is also dependent on population size, and therefore a clustering probability that generates a network of one size with the desired average clustering coefficient may not work for a larger network. Therefore, for the purposes of this section, the goal was for

the generated networks of a set population size (generally 1000 nodes, although in some cases a smaller population size was considered) to have properties that matched those in the target networks.

In all of the target networks, the nodes represent people, and the edges represent different types of connections between them. Since the evolving network algorithm generates simple, undirected graphs, most of the selected target networks were simple and undirected, and those that were not were modified accordingly; in networks that contained multiple edges, any duplicate edges were removed, and directed networks were made undirected by replacing any directed edge with an undirected edge, and then removing any resulting duplicate edges. The target networks are very different from each other in many ways, including size, average degree, level of clustering, and number of components. As a result, the algorithm was able to match the properties of some target networks better than others. In identifying which properties of each target network the algorithm was and wasn't able to replicate, its strengths and weaknesses are revealed. The target networks will be considered one at a time in order to achieve this goal.

### 6.4.1 Norwegian Boards

This dataset represents an interpersonal network among the directors of 384 different public limited companies in Norway during the month of May 2002. There are 5758 nodes in the network, and it is relatively sparse. The network properties are summarized in Table 6.11, and some images of the network structure are shown in Figure 6.10.

Table 6.11: Properties of the Norwegian Boards target network, where "ACC" refers to the average clustering coefficient of the network, and the diameter refers to the maximum diameter over all connected components.

| **Nodes** | **Average Degree** | **ACC** | **Density** | **Average Path Length** | **Diameter** |
|---|---|---|---|---|---|
| 5758 | 1.108 | 0.96 | 0.00019 | 3.07 | 8 |

This network is composed of 117 connected components and 4745 isolated nodes. The largest connected component contains 154 nodes, which is only 2.7% of the entire network population. Furthermore, its extremely high clustering coefficient suggests that many of the components are nearly complete graphs, which can be seen in Figure 6.10. With only a single initial infection, the spread of a disease on a network composed of many small components, such as this one, would die out very quickly.

As a result of the extremely high clustering coefficient and large number of components in this target network, the evolving network algorithm was not able to match its properties very accurately. Even with a clustering probability of 0.999, the average clustering coefficient of the networks generated using this degree distribution was only about 0.72, which is nowhere near as high as the average clustering coefficient of the

target network, 0.96. Table 6.12 shows the average results of 100 runs of the simulation on a population of 1000 nodes using a $c$ of 0.999, where the entire network was generated. In addition to the average clustering coefficient being considerably lower than in the target network, the average degree is lower as well. It is not completely understood why this is the case, since the degree of each of the nodes in the network is drawn from the degree distribution of the target network. The most likely explanation is that, while the average of the degrees assigned to the nodes does match the average degree of the target network, in the construction process the algorithm is unable to find enough suitable neighbors for each node in order to allow the node to have its full assigned degree. Generating sufficiently large networks could address this problem. However, this would be unlikely to address the disparity in average clustering coefficient between target network and simulated networks, since increasing the population size while maintaining the same clustering probability generally leads to a decrease in average clustering coefficient of the generated networks.

Table 6.12: Average properties over 100 runs of the simulation on a population of 1000 nodes, using the degree distribution from the Norwegian Boards target network and a clustering probability of 0.999.

| **Average Degree** | **ACC** | **Density** | **Average Path Length** | **Diameter** |
|---|---|---|---|---|
| $0.0355 \pm 0.0234$ | $0.724 \pm 0.092$ | $0.0000355 \pm 0.0000234$ | $1.60 \pm 0.26$ | $2.63 \pm 0.76$ |

Furthermore, from Tables 6.11 and 6.12 it is evident that on average, the target network is far less sparse than the simulated networks. This is likely a direct result of the lower average degree in the simulated networks, since average degree and density are directly proportional to each other. In addition, the target network has a longer average path length and diameter than the simulated networks. As with the average degree, this might simply be due to the population size, and running the simulation on a population of 5000 (approximately the size of the entire target network) could resolve this inconsistency. However, even if using a larger population size were able to resolve this issue, the inability of the algorithm to replicate the properties of the target network for a smaller population size is still a concern.

In short, the algorithm was not able to replicate the properties of the Norwegian Boards network very accurately. The most likely cause of this is the extremely high average clustering coefficient of this target network. As the clustering probability is increased in order to increase the average clustering coefficient of the generated networks, this also increases the likelihood that the algorithm will not be able to find suitable neighbors for the nodes, and therefore fewer edges are added to the network. This would lead to all of the trends that were observed in the simulated networks. However, an average clustering coefficient as high as 0.96 would be very unlikely in true social networks, and in Sections 6.4.2-6.4.5, we will see some examples of target networks whose properties are better matched by the proposed evolving network algorithm.

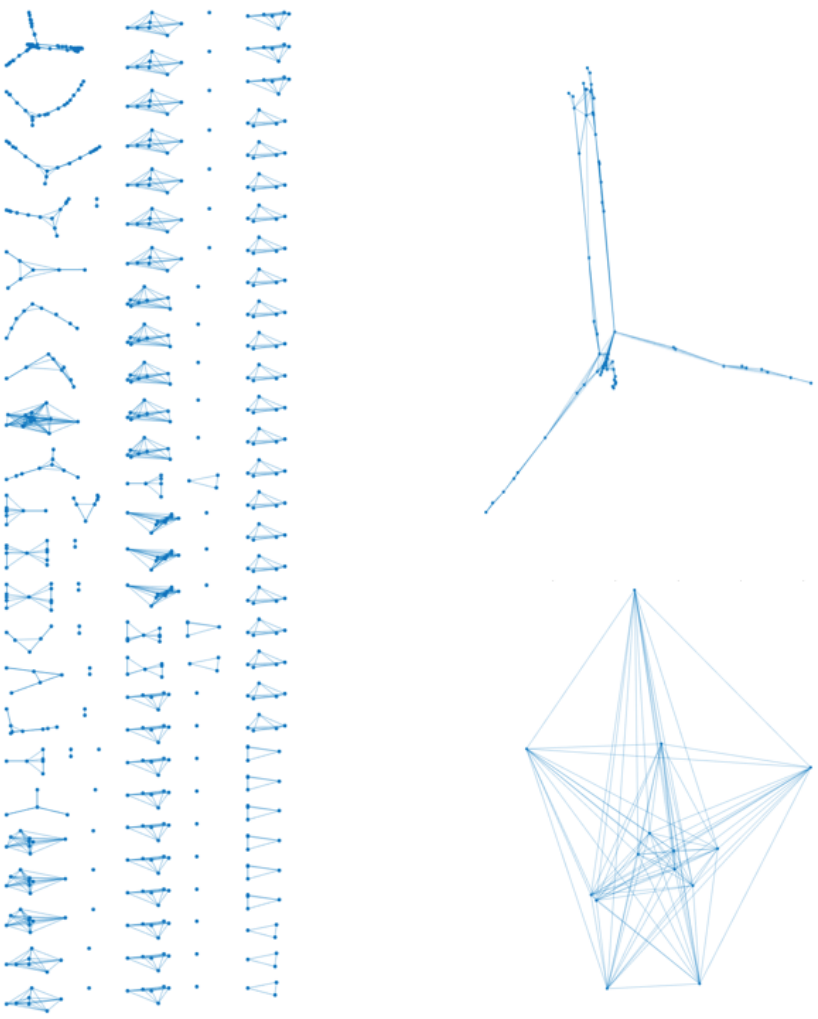


Figure 6.10: Left: Most of the Norwegian Boards network excluding isolated nodes and some connected components, each of which is a complete graph on three vertices. Right: Two different connected components from the Norwegian Boards network, the bottom of which is a nearly complete graph on 14 vertices.

### 6.4.2 Journal Authors

This dataset represents a collaboration network between authors in the Slovenian journals *Informatika* and *Uporabna Informatika*, where each node represents an author, and an edge between two authors signifies a paper that they collaborated on. This network is larger than the last, and contains 12820 nodes, 952 of which are isolated. There are 777 components, the largest of which is composed of 9663 nodes. The properties of this network are shown in Table 6.13, and images of the graph of the network are shown in Figure 6.11.

Table 6.13: Properties of the Journal Authors target network, where "ACC" refers ot the average clustering coefficient of the network, and the diameter represents the diameter of the largest connected component.

| **Nodes** | **Average Degree** | **ACC** | **Density** | **Average Path Length** | **Diameter** |
|---|---|---|---|---|---|
| 12820 | 4.15 | 0.63 | 0.00030 | 6.717 | 19 |

Like the Norwegian Boards network, this network is composed of many small components. However, one important distinction is that the largest connected component of this network contains about 75% of the total population, which makes this network very different from the Norwegian Boards network. The average properties over 100 runs of the simulation on 1000 nodes, using the degree distribution from the Journal Authors Network and a $c$ of 0.97, are shown in Table 6.14. This clustering probability was chosen in order to best match the average clustering coefficient of the target network. Comparing these results to Table 6.13, it is clear that the average degree and clustering coefficients match very closely. However, the density of the simulated networks is, on average, an order of magnitude larger than that of the target network. This seems strange, considering that the average degree is about the same, but it reflects the fact that the full target network contains 12820 nodes, while the simulated networks only contain 1000 nodes. Thus, the maximal possible number of edges is much larger in the much more extensive target network, so its density is much smaller. This discrepancy would likely be addressed by simply increasing the population size in the simulations.

Table 6.14: Average properties of 100 runs of the evolving network algorithm using the degree distribution from the Journal Authors target network. The population size in each of the simulations was 1000, and the clustering probability was 0.97.

| **Average Degree** | **ACC** | **Density** | **Average Path Length** | **Diameter** |
|---|---|---|---|---|
| $4.09 \pm 0.15$ | $0.64 \pm 0.02$ | $0.0029 \pm 0.001$ | $12.45 \pm 4.44$ | $35.39 \pm 13.89$ |

Finally, the average path length and diameter in the networks generated by the evolving network algorithm were, on average, about twice those of the target network. This suggests that the components in the networks generated by the evolving network algorithm are generally slightly larger than the components in the target network. This is consistent with results that are generally observed with the algorithm, in that it often generates more, larger components, instead of fewer smaller ones. Two examples of graphs generated using the algorithm and the Journal Authors degree distribution are shown in Figure 6.11, along with images of the entire target network and its largest connected component. These two particular examples graphs have 3 and 4 components, respectively, which is far fewer than the 777 components in the target network, so it makes sense that the average path length and diameter of the simulated networks would be longer than those

for the target network, despite the smaller size of the simulated networks. However, in general the evolving network algorithm does a good job of replicating the properties of the Journal Authors target network. In addition, as demonstrated in the previous section, increasing the population size causes an increase in the number of components in the network, as well as a slight decrease in the average clustering coefficient. Thus, an increase in the population size would also require an increase in the clustering probability, which further promotes the formation of more, smaller components. As a result, perhaps increasing the population size for the simulations would resolve the differences in properties between the target network and the simulated networks.

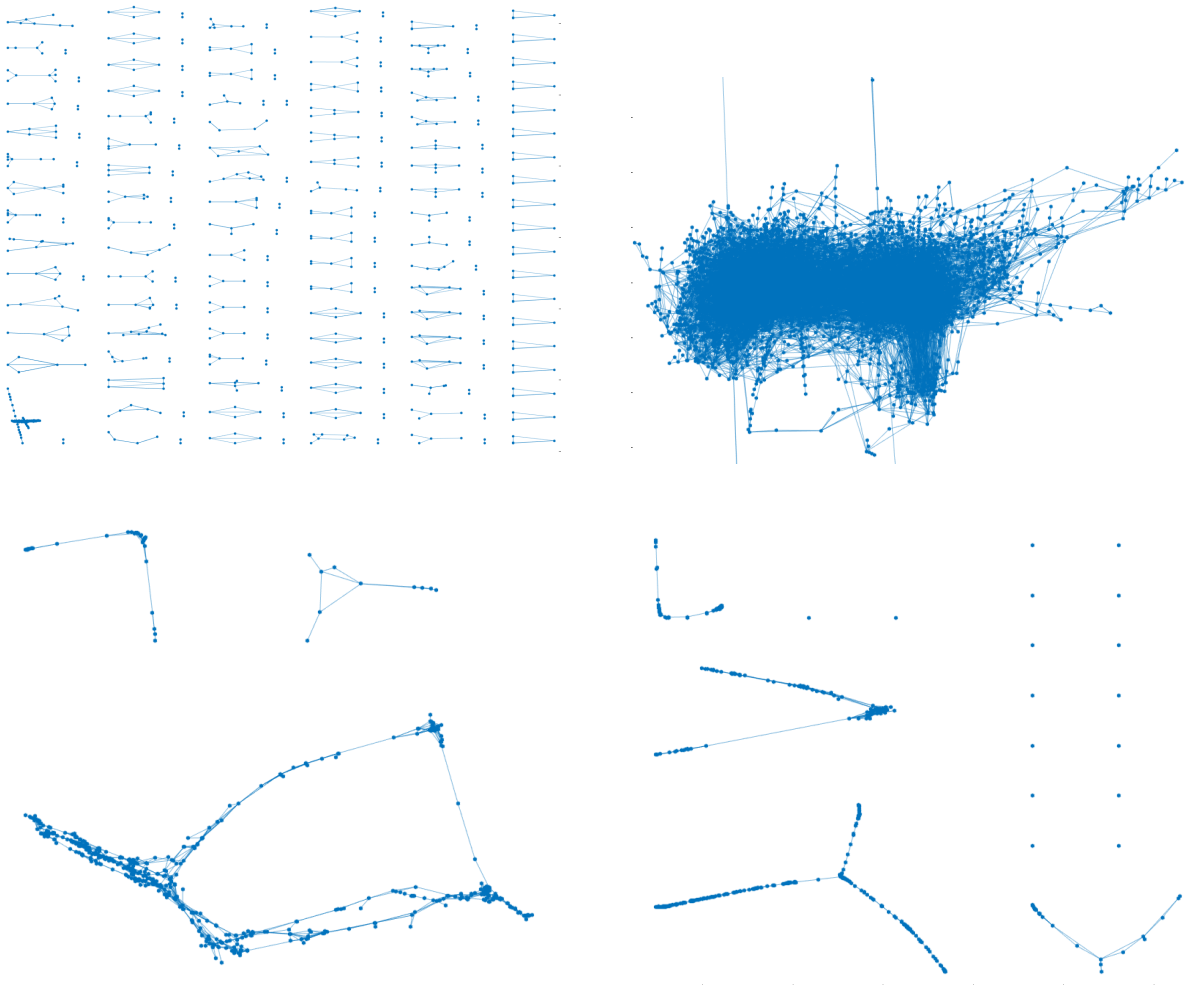


Figure 6.11: Top: Graph of the most of the Journal Authors Network excluding many isolated nodes and some small components (left), and a closeup of the largest connected component (right). Bottom: Two examples of simulated networks using the degree distribution from the Journal Authors network on 1000 nodes with a $c$ of 0.97.

### 6.4.3 Friendship Network

This network represents the self-reported friendships between students in a high school in Marseilles, France, in December 2013. While the original network is directed, for the purposes of this project it was treated as an undirected network, where an edge in either or both directions was considered as an undirected edge. There are 1828 nodes in the network altogether, 1694 of which are isolated. The rest of the nodes are divided between 3 components, the largest of which is composed of 128 nodes, or about 7% of the network (see Figure 6.12). The properties of the network are described in Table 6.15.

Table 6.15: Properties of the Friendship target network, where "ACC" refers ot the average clustering coefficient of the network, and the diameter represents the diameter of the largest connected component.

| **Nodes** | **Average Degree** | **ACC** | **Density** | **Average Path Length** | **Diameter** |
|---|---|---|---|---|---|
| 1828 | 0.36 | 0.56 | 0.00024 | 4.02 | 10 |

Because so many of the nodes in this network are isolated, the average degree of the network is very low (0.36), as is the density (0.00024). However, the network has a relatively large average clustering coefficient of 0.56. Table 6.16 shows the average results of 100 runs of the evolving network algorithm using the Friendship Network degree distribution and a $c$ of 0.6. Comparing Tables 6.15 and 6.16, the average degrees are very similar, which would be expected since the degrees of the nodes in the simulated networks were drawn from the degree distribution of the graph whose properties are shown in Table 6.15. The average clustering coefficients are also quite close (the true average clustering coefficient of the target network is within one standard deviation of the average of that for the simulations), and could potentially be fine-tuned even more closely by decreasing the clustering probability slightly. The average density of the generated networks is relatively close to that of the target network, although it is slightly larger. The reason for this difference is not completely understood, especially because the average degree of the simulated networks is actually slightly smaller than that of the target network. Nonetheless, since the density is such a small number, these values of 0.00024 and 0.00035 are relatively close and the difference can be considered unimportant.

Table 6.16: Average properties over 100 runs of the simulation on a population of 1000 nodes, using the degree distribution from the Friendship target network and a clustering probability of 0.6.

| **Average Degree** | **ACC** | **Density** | **Average Path Length** | **Diameter** |
|---|---|---|---|---|
| $0.345 \pm 0.048$ | $0.608 \pm 0.046$ | $0.00035 \pm 0.00005$ | $4.63 \pm 0.84$ | $11.69 \pm 2.65$ |

Finally, both the average path length and diameter for the networks generated by the algorithm are slightly larger than those of the target network. As with the Journal Authors network, this is likely due

to the larger number of components in the target network than in the networks generated by the evolving network algorithm. While the target network has 3 components, of the 100 networks generated, only 3 had 3 components, 9 had 2, and the remaining 88 had only 1 component. This discrepancy is not as dramatic as with the Journal Authors network, and therefore its effect on the average path length and diameter is not as noticeable. However, as with the Journal Authors simulations, it is likely that increasing the population size would increase the number of components and therefore could likely resolve the inconsistency. Nonetheless, this discrepancy is relatively slight, which might also be related to the fact that the size of the simulated networks is closer to the size of the entire target network (in contrast to the Journal Authors network, where the target network is more than 10 times the size of the networks generated by the algorithm).

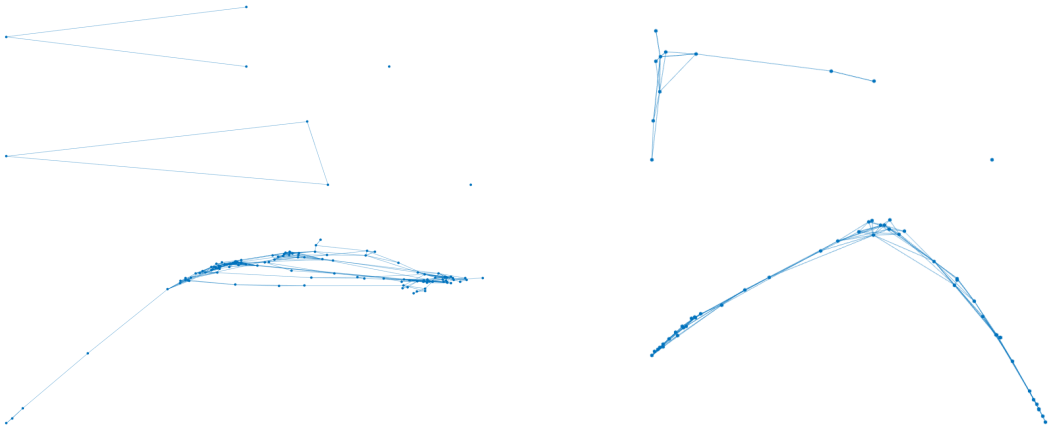


Figure 6.12: Left: Graph of the Friendship Network, excluding most of the isolated nodes. The largest component is composed of 128 nodes, while each of the smaller components include 3 nodes. Right: One example of a simulated network on 1000 nodes, excluding isolated nodes, generated using the Friendship Network degree distribution and a clustering probability of 0.6.

### 6.4.4 Wiki-Vote

The Wiki-Vote dataset is not exactly a network of social connections in the same way as the previous three networks; however, it still represents interactions between people, and thus can be used as a sample social network for disease simulation. Within Wikipedia, any user can contribute information, but a small portion of the Wikipedia contributors are administrators, who are have access to additional technical features. The Wikipedia community votes in order to decide which users to promote to adminship. Thus, a network can be constructed in which the nodes represent users, and the edges votes between them. The particular network contains all the Wikipedia voting data from the time that Wikipedia was first created until January 2008.[23]

Unlike the three networks already discussed, this network is connected. It contains 889 nodes, and its properties are summarized in Table 6.17. Since this network is composed of a single connected component, it is a good candidate for a target network for the simulations. In addition, it demonstrates a high degree of small-world character, as can be seen by the relatively small average path length given the population size. Since many social networks demonstrate high small-world character, this network is in many ways a good representation of a social network on which a disease could spread. This network is also much more dense than the networks previously considered, which in part reflects the fact that it does not contain any isolated nodes. Finally, its average clustering coefficient of 0.19 is lower compared to the target networks considered thus far, and as a result is more in the range of the level of clustering in organic networks.

Table 6.17: Properties of the Wiki-Vote target network, where "ACC" refers to the average clustering coefficient of the network.

| **Nodes** | **Average Degree** | **ACC** | **Density** | **Average Path Length** | **Diameter** |
|---|---|---|---|---|---|
| 889 | 6.56 | 0.19 | 0.0074 | 4.10 | 13 |

The results of the average of 100 simulations using the degree distribution from the Wiki-Vote Network and a clustering probability of 0.35 are shown in Table 6.18. As with the other target networks, this clustering probability was chosen in order to best match the average clustering coefficient of the target network. The average degree and average clustering coefficient of the target network both fall within one standard deviation of the mean of the simulations, suggesting that the simulations did a decent job of matching these properties of the target network. Furthermore, the average density of the graphs generated using the algorithm is relatively close to the density of the target network, with a percent error of 14.9%. The fact that the average degree and density are both slightly lower than in the target network likely reflects the fact that towards the end of the simulation, the algorithm was unable to find suitable neighbors for all of the nodes, which is a pattern seen consistently. However, since density is such a small value, this error is relatively inconsequential.

Table 6.18: Average properties of 100 runs of the evolving network algorithm using the degree distribution from the Wiki-Vote target network. The population size in each of the simulations was 1000, and the clustering probability was 0.35.

| **Average Degree** | **ACC** | **Density** | **Average Path Length** | **Diameter** |
|---|---|---|---|---|
| $6.33 \pm 0.26$ | $0.20 \pm 0.01$ | $0.0063 \pm 0.0003$ | $3.80 \pm 0.12$ | $8.72 \pm 1.21$ |

Finally, the average path length and diameter of the target network are slightly larger than those calculated from the simulated networks. This trend is the opposite from that seen in the networks previously considered, and can be understood by comparing the structure of the actual network with that of one of the

simulated networks. As can be seen in Figure 6.13, the graph of the Wiki-Vote network has a few nodes that are extended out at a relatively far distance from the rest of the network (in the top left and bottom right of the graph). This particular structural feature is difficult to replicate in the simulated networks without controlling the architecture of the networks very carefully. By contrast, the graph on the right, which is one example of a network constructed by the algorithm using the Wiki-Vote degree distribution and a $c$ of 0.35, does not appear to have the same kind of long extensions. Since this simulated network has almost exactly the same degree distribution as the target network, it does have just as many nodes of degree 1 and 2. However, these nodes must be contained in cycles, and thus are closer to the other nodes than in the target network. Nevertheless, in general the algorithm did a reasonably good job of generating networks with properties of the Wiki-Vote target network.

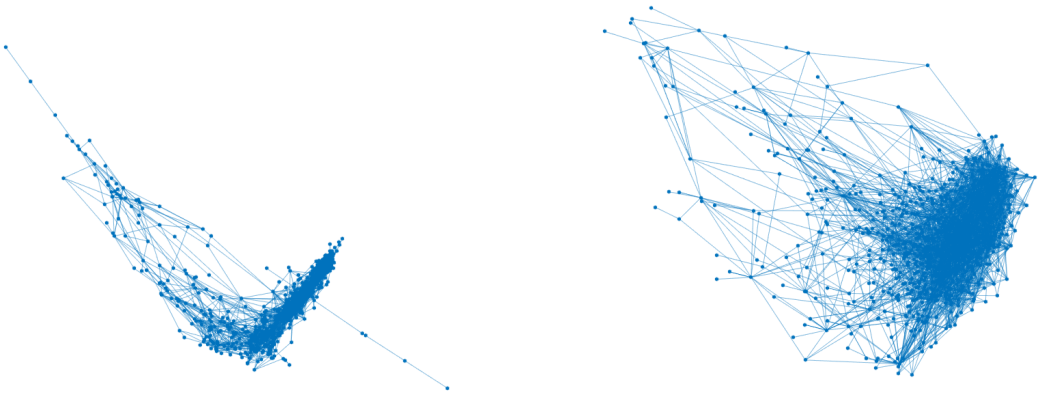


Figure 6.13: Graph of the actual Wiki-Vote Network (left) and an example of a simulated network with $c = 0.35$ (right).

### 6.4.5 Infectious Network

The last network that will be considered was describes the interactions between people at an exhibition in Dublin, Ireland in 2009. This free interactive exhibit explored the mechanisms of disease contagion and strategies of containment by warning visitors to "stay away", and tracking their interactions. The nodes represent people who visited the exhibit, and an edge between two nodes represents a face-to-face interaction that lasted for at least 20 seconds. The properties of this network are shown in Table 6.19. Like the Wiki-Vote network, this network is composed of a single connected component and has high small-world character, as demonstrated by its average path length of 3.63. It also has a relatively high average clustering coefficient of 0.47, and is denser than any of the other target networks considered. Finally, its smaller size distinguishes

it from the other networks considered, as it is composed of only 410 nodes. As a result, the simulations were run on only 400 nodes rather than 1000, since the target network itself contains fewer than 1000 nodes.

Table 6.19: Properties of the Infectious Network target network, where "ACC" refers ot the average clustering coefficient of the network.

| **Nodes** | **Average Degree** | **ACC** | **Density** | **Average Path Length** | **Diameter** |
|---|---|---|---|---|---|
| 410 | 13.49 | 0.47 | 0.033 | 3.63 | 9 |

The average results of 100 simulations using the degree distribution from the Infectious Network and a $c$ of 0.97 are shown in Table 6.20. The average degree, average clustering coefficient, and graph density all match those of the target network very closely, suggesting that the simulations accurately mimic the properties of the original graph. The average path length and diameter calculated from the simulations, while relatively consistent with those of the target network, were found to be slightly larger. The explanation for this trend is not the same as in the previous examples, since there is only one component. Instead, as with the Wiki-Vote simulations, this discrepancy is likely caused by some feature of the target network that cannot be perfectly accounted for in the degree distribution and level of clustering. Figure 6.14 shows a graph of the full target network as well as an example of a simulated network on 400 nodes. Visually, these networks appear relatively similar. However, based on the deviation in the average path length and diameter, the simulated networks must have some nodes that are extended further from the rest of the network than in the target network. This can be observed in the lower right-hand corner of the graph that was generated by the algorithm, where there are a few nodes extended further out from the rest of the network.

Table 6.20: Average properties of 100 runs of the evolving network algorithm using the degree distribution from the Infectious target network. The population size in each of the simulations was 400, and the clustering probability was 0.97.

| **Average Degree** | **ACC** | **Density** | **Average Path Length** | **Diameter** |
|---|---|---|---|---|
| $13.26 \pm 0.47$ | $0.46 \pm 0.02$ | $0.031 \pm 0.001$ | $4.08 \pm 0.27$ | $9.65 \pm 1.15$ |

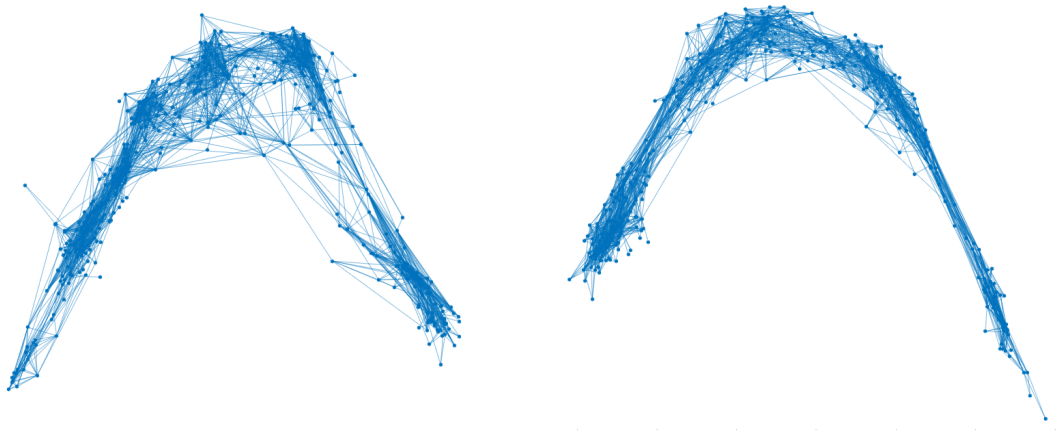


Figure 6.14: Graph of the actual Infectious Network (left) and an example of a simulated network on 400 nodes with $c = 0.97$ (right).

Nevertheless, the evolving network algorithm is overall very successful in replicating the properties of the Infectious target network. This success, as well as the success in replicating the properties of the Wiki-Vote network, suggests that connected networks with high small-world character can be modeled relatively accurately by the evolving network algorithm. In contrast the networks composed of many smaller, disconnected components were not modeled as accurately by the evolving network algorithm. While the network properties vary based on the epidemiological features of the disease being studied, many epidemiological networks are connected or composed of only a few components. Additionally, many *do* demonstrate high small world character.[2] As a result, it can be concluded that the algorithm has potential to be useful in simulating the spread of disease on epidemiological contact networks. While its inability to replicate the properties of all graphs is a concern, this limitation is unlikely to interfere with its utility for the purpose that it was created.

# 7 Conclusion

Accurately modeling the spread of infectious diseases is essential for disease prevention and treatment strategies. Using the tool of mathematical disease simulations, it is possible to develop prevention strategies for epidemiological events that will hopefully never occur. The two most common approaches to disease modeling, compartmental and agent-based models, offer a trade-off between the accuracy of the predictions they make and the efficiency with which they can provide information. Compartmental disease models are very well-established and can make excellent predictions for certain highly infectious diseases under the assumption of homogeneous mixing, especially when using a large number of specialized disease compartments. However, for some diseases it is not reasonable to assume the presence of homogeneous mixing because of their transmission mechanisms. For example, Ebola is much more likely to spread between close contacts than casual contacts, and as a result compartmental models cannot make accurate infection predictions for this disease. In cases like this, agent-based models are able to provide more accurate predictions using information about the connections between the individuals in the population. However, while agent-based models provide valuable insight for certain types of diseases, they are very computationally intense, especially when used to model large populations. Furthermore, in cases where the epidemic begins with a single initial infection, it is unnecessary to model the structure of the entire network at all times, and by avoiding this, much computational time can be saved. This is the idea behind the evolving network model discussed in this paper.

The basic premise of the proposed evolving network model is that initially the entire population is modeled compartmentally, where all nodes are susceptible, and as nodes become infected their contact networks are built. This seemingly simple concept holds enormous potential, especially for newly emerging or reemerging diseases, like Ebola, that are low prevalence and spread only through close contact. In short, the algorithm takes a degree distribution and clustering probability as inputs, and constructs the network by first randomly assigning each node a degree from the distribution and then evolving the network through the addition of edges in order to fulfill the assigned degree of each node. The construction of such a model is poses an interesting, complex mathematical problem. The overarching goal is to find a way to grow an evolving network, such that each node that is added builds towards the global properties of the network, even though this addition is based only on local information. Achieving this goal requires extensive consideration of not only how to develop the algorithm for this network construction process, but also how to evaluate the networks produced by this algorithm.

Despite the stochasticity of this process, the results were shown to be relatively consistent for each set of input parameters. By changing the transmission probability, $\beta$, and recovery probability, $r$, the disease progress can be controlled, and these infection dynamics show consistency across runs and closely replicate patterns seen in more traditional disease simulations. Future work could aim to further explore the consistency between the dynamics of the spread of disease on the evolving network and the spread of disease on the corresponding full target network, which is considered briefly in Appendix 9.1. Furthermore, while this paper only explored the use of this algorithm in modeling three sets of disease dynamics (susceptible-infected, susceptible-infected-recovered, and susceptible-infected-susceptible), in future work the evolving network model could be expanded to other, more complicated disease scenarios, such as susceptible-infected-recovered-susceptible (SIRS) or susceptible-exposed-infected-recovered (SEIS); the latter of which could be used to more accurately simulate the spread of a disease like Ebola, which has an incubation period before exposed individuals become infected. Finally, through the incorporation of multiple contact types, like in the Ebola model proposed by Kiskowski [17], the accuracy of the predictions made by the algorithm could be further improved.

In addition to the disease dynamics, this paper explored the properties of the contact networks generated using concepts drawn from graph theory. In particular, the graphs were compared and assessed based on five main metrics: average degree, average clustering coefficient, density, average path length, and diameter. While the average degree and average clustering coefficient are constrained relatively precisely by the inputs to the algorithm, the other three properties are not specifically controlled for by the algorithm. However, the results demonstrated that the networks generated by different runs of the evolving network algorithm using the same input parameters are still relatively consistent in their density, average path length, and diameter. While the input "clustering probability" has a notable effect on the average clustering coefficient of the networks, it does not appear to have a large effect on the average degree or density of the networks, which supports that the $c$ is serving its desired purpose of increasing the level of clustering in the networks without significantly altering their structures otherwise. It does, however, appear to impact the average path length, diameter, and number of components of the graphs generated, which is consistent with results found previously in a paper from 2009.[4] While these correlations could pose limitations in the network construction process, they also help to explain the ability of the algorithm to generate networks whose properties match those of various target networks, beyond just the average degree and clustering coefficient.

Finally, the ability of the algorithm to generate graphs whose properties match those of five different real network datasets was assessed. Generally, the algorithm was able to match their properties relatively well

(as assessed through all of the metrics just discussed), although it did appear to have certain limitations. For example, the properties of networks consisting of a single connected component were matched much more easily than those containing many small components. Many social networks are connected, in which case this limitation would not pose a problem. However, it is still important to consider ways that this challenge could be overcome to optimize the usefulness of the algorithm. In addition, the algorithm was unable to replicate the properties of networks with extremely high average clustering coefficients, and in some cases was not able to perfectly replicate certain structural features of networks that could not be captured in the degree distribution or level of clustering. While this latter issue could be resolved by the incorporation into the algorithm of additional input parameters which further specify the properties of the network, doing so could potentially lessen the utility of this algorithm for epidemiological implementation. In order for the evolving network algorithm to be used successfully for epidemiological studies, the input parameters must be drawn from information that can be extracted from real populations. While information such as the degree distribution and average clustering coefficient have been estimated for real populations by sampling, it would be difficult to estimate global properties, such as average path length and diameter, without having access to detailed information about the social network which is being modeled. Thus, in only requiring two input parameters in order to replicate network properties relatively accurately, the algorithm is easier to use for epidemiological applications. However, further investigation is necessary in order to assess whether the slight differences between the target networks and those generated by the algorithm are significant enough to warrant the incorporation of additional control in the network construction process.

It would also be informative to consider the properties of the evolving networks at an earlier stage in the simulation, rather than just once the infection has spread to most of the population. In doing this, we could see whether the properties of these smaller sub-networks match the properties of the larger networks, as well as the properties of the target networks. Furthermore, we would also like to investigate the implementation of infection control strategies, such as vaccination and quarantine measures, using the evolving network algorithm. This could be achieved through relatively simple modifications to the code, and would provide further insight into the potential usefulness of the evolving network model for epidemiological purposes. Finally, and perhaps most critically, we would like to find a way to sample the parameter that we are calling "clustering probability" from the target networks. Although this property is used in the construction of the networks, it is not completely understood what it represents. Efforts to sample $c$ from target networks by looking at the fraction of closed triangles were unsuccessful; the $c$ sampled from the target networks was consistently lower than that required to generate networks with properties that matched those of the target

networks. Furthermore, even when a certain clustering probability was used to generate a network, and then the $c$ of that generated network was calculated, there was still disagreement between the values. As a result, in order to decide what value of $c$ to use in generating the networks, a range of values were tested to see which one allowed for the generation of networks whose average clustering coefficient best matched that of the target network. While this worked reasonably well for most of the target networks considered, it would not be feasible in a situation where a full target network is not available from which to calculate the average clustering coefficient. Therefore, it is desirable to find a way to sample the clustering probability from target networks and be able to use this as an input to the algorithm. Future work will focus on this goal.

# 8 References


[1] G. Alanis-Lobato. *CNM: A Matlab toolbox for the construction of artificial complex networks*. 2014. URL: https://www.mathworks.com/matlabcentral/fileexchange/45734-cnm?focused=3810104&tab=function.

[2] R. Albert and A.-L. Barabasi. "Statistical mechanics of complex networks". In: *Rev. Mod. Phys.* 74 (2002), pp. 47–97. DOI: https://doi.org/10.1103/RevModPhys.74.47.

[3] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. London: Springer, 2000.

[4] S. Bansal, S. Khandelwal, and L. A. Meyers. "Exploring biological network structure with clustered random networks". In: *BMC Bioinformatics* 10.405 (2009). DOI: https://doi.org/10.1186/1471-2105-10-405.

[5] E. A. Bender and S. G. Williamson. "Ordinary Generating Functions". In: *Foundations of Combinatorics with Applications*. 2006.

[6] N. Blagus and M. Bajec. "The network of collaboration: Informatica and Uporabna Informatika". In: *Uporabna Informatika* 23.1 (2015), pp. 22–31.

[7] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. New York: Elsevier Science Publishing Co., 1995.

[8] A. Brouwer and W. H. Haemers. *Spectra of Graphs*. New York: Springer, 2012.

[9] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic Complexity Theory*. Berlin, Germany: Springer Verlag, 1997.

[10] C. Gopalappa. *Novel Multi-dimensional Evolving Contact Network (MECN) algorithm*. NIH MIDAS Grant Proposal. 2016.

[11] C. M. Grinstead and J. L. Snell. "Markov Chains". In: *Introduction to Probability*. Gainesville, FL: University Press of Florida, 2009, pp. 405–470.

[12] E. Halloran. "Communicable Diseases and Data Analysis". In: *Biometrics - Volume II*. Oxford, UK: EOLSS Publishers, 2009, pp. 148–161.

[13] A. Haveman-Nies et al., eds. *Epidemiology in Public Health Practice*. The Netherlands: Wageningen Academic Publishers, 2010.

[14] L. Isella et al. “What’s in a crowd? analysis of face-to-face behavioral networks”. In: *J. of Theoretical Biology* 271.1 (2011), pp. 166–180.

[15] M. J. Keeling and K. T. D. Eames. “Networks and Epidemic Models”. In: *Journal of the Royal Society Interface* 2.4 (2005). DOI: https://doi.org/10.1098/rsif.2005.0051.

[16] M. Kiskowski and G. Chowell. “Modeling household and community transmission of Ebola virus disease: epidemic growth, spatial dynamics and insights for epidemic control”. In: *Virulence* 7.2 (2015), pp. 163–173. DOI: https://doi.org/10.1080/21505594.2015.1076613.

[17] M.A. Kiskowski. “A Three Scale Network Model for the Early Growth Dynamics of 2014 West Africa Ebola Epidemic”. In: *PLOS Currents Outbreaks* (2014). DOI: https://doi.org/10.1371/currents.outbreaks.c6efe8274dc55274f05cbcb62bbe6070.

[18] R. Mastrandrea, J. Fournet, and A. Barrat. “Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys”. In: *PLoS ONE* 9 (2015).

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

[20] M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics*. Oxford: Oxford University Press, 1999.

[21] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. “Random graphs with arbitrary degree distribution and their applications”. In: *Phys. Rev. E* 64.026118 (2001). DOI: https://doi.org/10.1103/PhysRevE.64.026118.

[22] B. J. Prettejohn, M. J. Berryman, and M. D. McDonnell. “Methods for generating complex networks with selected structural properties for simulations: A review and tutorial for neuroscientists”. In: *Frontiers in Computational Neuroscience* 5.11 (2011). DOI: https://doi.org/10.3389/fncom.2011.00011.

[23] R. Rossi and N. Ahmed. *Network Repository*. 2013. URL: http://networkrepository.com.

[24] C. Seierstad and T. Opsahl. “For the few not the many? The effects of affirmative action on presence, prominence, and social capital of female directors in Norway”. In: *Scandinavian Journal of Management* 27.1 (2011), pp. 44–45.

[25] *SI and SIS Models - Generic Model Documentation*. Intellectual Ventures Management, LLC, 2018. URL: https://institutefordiseasemodeling.github.io/Documentation/general/model-si.html#sis.

[26] *SIR and SIRS models*. Institute for Disease Modeling. URL: https://institutefordiseasemodeling.github.io/Documentation/general/model-sir.html#sir-and-sirs-models.

[27] S. Tsironis, M. Sozio, and M. Vazirgiannis. "Accurate Spectral Clustering for Community Detection in MapReduce". In: (2013).

[28] E. Volz. "Random networks with tunable degree distribution and clustering". In: *Phys. Rev. E* 70.056115 (2004). DOI: https://doi.org/10.1103/PhysRevE.70.056115.

[29] D. J. Watts and S. H. Strogatz. "Collective dynamics of small-world networks". In: *Nature* 393 (1998), pp. 440–442. DOI: https://doi.org/10.1038/30918.

[30] E. W. Weisstein. *Simple Graph*. MathWorld–A Wolfram Web Resource. URL: http://mathworld.wolfram.com/SimpleGraph.html.

[31] *WHO Ebola virus disease*. World Health Organization. URL: http://www.who.int/mediacentre/factsheets/fs103/en/.

# 9 Appendix

## 9.1 Additional Infection Progress Plots

As alluded to in the results section, one additional way to assess the success of the model is to compare the infection progress on the evolving network to that on a static, agent based network. The next few figures were generated by taking a single network, and recording the infection progress as the disease spreads on this network, beginning with a different random infection each time. The networks used were generated by the proposed evolving network algorithm, and therefore share the same degree distribution and level of clustering. In particular, the two examples shown in Figures 9.1 and 9.2 used two different networks that were generated using the same degree distribution, clustering probability (0.2), and population size (100) as in Section 6.1 in order to allow for direct comparison.

While there are clearly some differences between the two sets of plots in Figures 9.1 and 9.2, as well as between these plots and those in Figure 6.1, the overall shapes of all three sets of plots are the same and the peak infection prevalence occurs at approximately the same day in the simulation. There is considerably less variation in the different infection progress curves within these plots than within those shown in Figure 6.1, which reflects the fact that in these plots, the infection is spreading on exactly the same network each time. However, based on these plots, it appears that at least for the scenario where no recovery is allowed, the evolving network algorithm does a relatively good job of generate networks on which the disease spreads similarly to on a static, agent based network.

Finally, Figure 9.3 is analogous to Figure 6.4 in that it shows plots of the infection progress on a network with a population size of 1000 using each of the three disease models. All of the input parameters are the same as in Figure 6.4, with a clustering probability of 0.2, transmission probability of 0.1, and recovery probability of 0.005 for the models that allow recovery. These three graphs closely resemble the corresponding ones generated using the proposed evolving network algorithm. The peak prevalence occurs at approximately the same day in the simulation (slightly after day 50), and the shapes of all three curves are very similar. As in Figures 9.1 and 9.2, there is considerably less variation in these curves than in those shown in Figure 6.4, which is simply due to the fact that each run is using the same network.

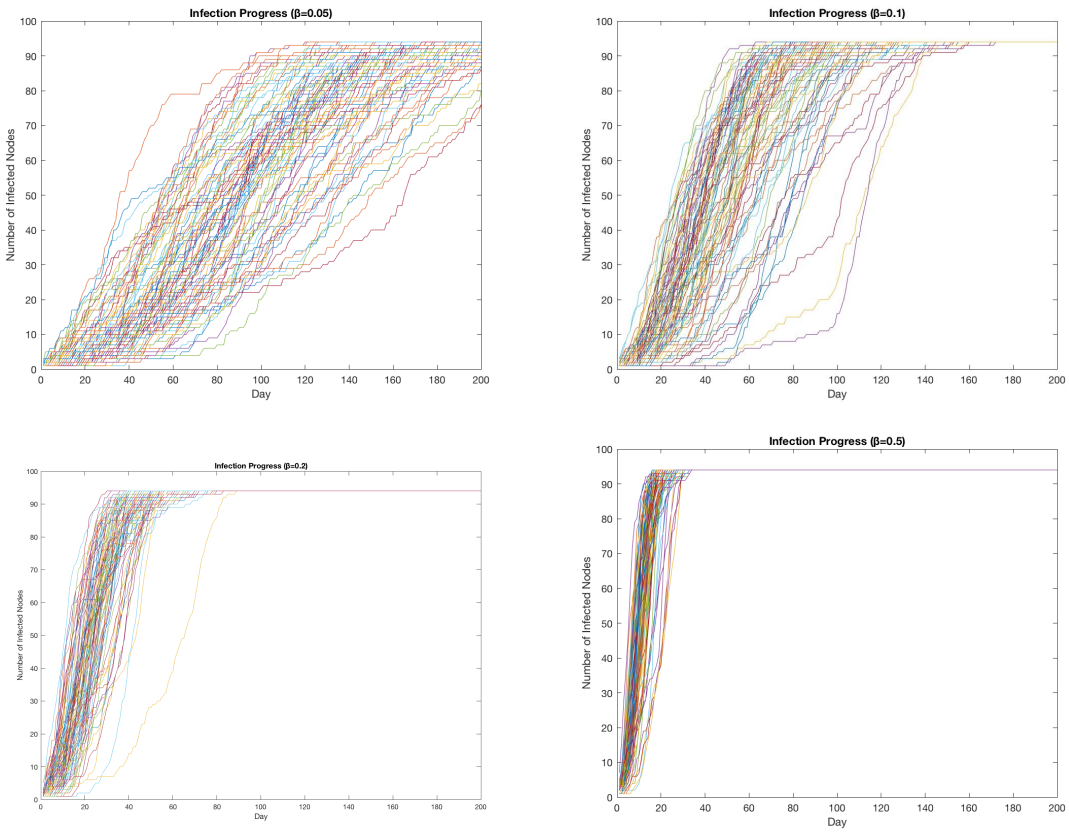



Figure 9.1: Plots of the infection progress on the full network for four different transmission probabilities (clockwise from top left, 0.05, 0.1, 0.2, and 0.5) with no recovery, using the same network each time, but different random initial infections. The network was composed of a single large component containing 94 nodes, and 6 isolated nodes. Compare to Figure 6.1.

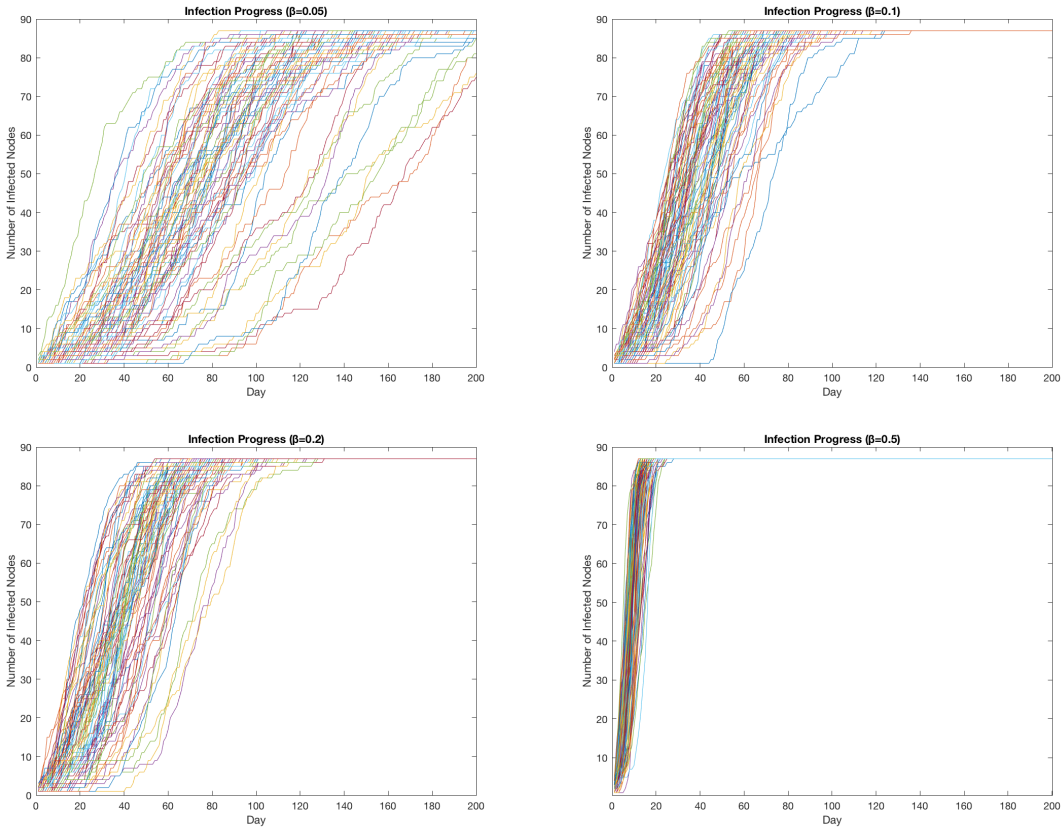



Figure 9.2: Plots of the infection progress on the full network for four different transmission probabilities (clockwise from top left, 0.05, 0.1, 0.2, and 0.5) with no recovery, using the same network each time, but different random initial infections. The network was composed of a single large component containing 87 nodes, and 13 isolated nodes. Compare to Figure 6.1.

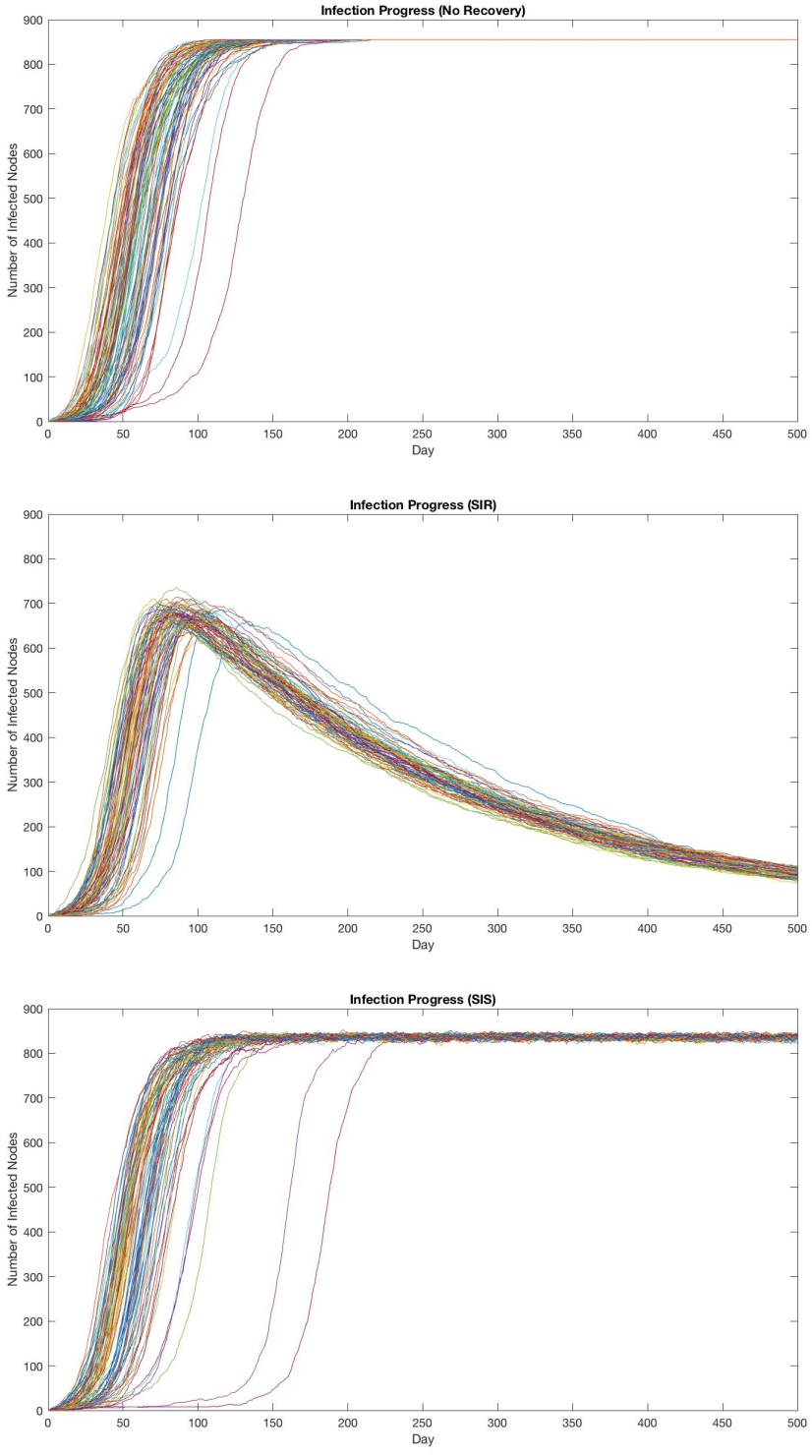



Figure 9.3: Plots of the first 500 days of infection progress on a static network, using each model. Each plot represents 100 runs, where each line is the number of infectious individuals at each timestep in a single run. All of the simulations were run with a population of 1000 and transmission probability of 0.1, and the models that include recovery were run with a recovery probability of 0.005. Compare to Figure 6.4.

## 9.2 Evolving Network Code

The following code is the simplest version of the evolving network model proposed in this paper. It does not include recovery. In order to incorporate recovery, a few slight modifications to this code are necessary. A description of what each section of this code achieves is provided in Section 5. The parameter *cp* is the clustering probability, which was called *c* throughout this paper.

```
k=0;
%change this to generate either one connected component (k=0) or the full network (k=1)

n=1000; %size of population
cp=0.2; %probability with which friends of friends are added first
beta=0.1; %transmission rate

cumulativedistribution=([possible degrees; cumulative distribution]);

A=zeros(n); %set up adjacency matrix

%set up the degree matrix and choose the degrees of all nodes
DegreeMatrix=zeros([n 2]);
for i=1:n
        DegreeMatrix(i,1)=i; %make the first column the index of the node
        p=rand; %generate a uniform random number between 0 and 1
        x=1; %start at the first column of the cumulative distribution
        while p > cumulativedistribution(2,x)
                x=x+1;
        end
        DegreeMatrix(i,2)=cumulativedistribution(1,x);
        %make the second column the degree of the node, drawn from the distribution
end
```

```
Infected=[]; %create an array to keep track of all infected nodes
NewInfected=[]; %nodes who became infected at the last timestep

NumInfectedContacts=zeros([n 2]); %number of infected contacts of each node
for i=1:n
        NumInfectedContacts(i,1)=i; %make the first column the index of the node
end

maxit=10000; %change this to alter how long the code runs for
it=0; %keeps track of how many times it tries to find a neighbor for a given node
day=0; %keeps track of how many ''days" the simulation runs for
days_without_new_infection=0; %how long the code runs without any new infections

InfectionProgress=[];

x=ceil(rand*n); %pick a radnom node x to infect
NewInfected=[x]; %add node x to NewInfected
Infected=[x]; %add node x to Infected

while length(Infected)<n && it<maxit && day<1000
        it=0;
        if ~isempty(NewInfected)
        %if there are any newly infected nodes, find their contacts
                days_without_new_infection=0;
                for j=1:length(NewInfected) %for each newly infected node
                        a=NewInfected(j);
                        FriendsOfFriends=[];
                        %create an array to keep track of friends of friends of a
                A2=A^2; %square the adjacency matrix
                for i=1:n
```

```
                    if A2(a,i)>0
                        FriendsOfFriends=[FriendsOfFriends,i];
                    end
                    FriendsOfFriends=FriendsOfFriends(find(FriendsOfFriends~=a));
                    %remove a
                end

                while sum(A(a,:))<DegreeMatrix(a,2) && it<maxit
                %while the degree of a is less than the assigned degree
                    if length(FriendsOfFriends)>0 && cp>rand
                    %with probability p, choose from friends of friends
                        x=ceil(rand*length(FriendsOfFriends));
                        b=FriendsOfFriends(1,x);
                        %choose a random node from the friends of friends list
                    else
                        b=ceil(rand*n);
                    end
                    if (b ~= a) && (A(a,b) == 0) && (sum(A(b,:))<DegreeMatrix(b,2))
                        A(a,b)=1;
                        A(b,a)=1;
                    end
                it=it+1;
                %keep track of how hard it's looking to find a suitable connection
                end
        end
end

%now each newly infected node has its connections, so it's time to spread the infection
NewInfected=[];
```

```
%calculate the number of infected contacts of each node and update the matrix
for i=1:n
        count=0; %count the number of infected contacts of i
        for j=1:n
                if A(i,j)==1 && ismember(j,Infected(:))
                %if i and j are neighbors and j is infected
                        count=count+1;
                end
        end
        NumInfectedContacts(i,2)=count;
end

%spread the infection
for i=1:n
        if ~ismember(i,Infected(:)) && 1-(1-beta)^(NumInfectedContacts(i,2)) > rand
        %if node i is not already infected and p > random number
                NewInfected=[NewInfected i];
                Infected=[Infected i];
        end
end

if days_without_new_infection>500 %if the infection is no longer spreading
        'Infection has terminated.'
        break
end

days_without_new_infection=days_without_new_infection+1;
day=day+1;
InfectionProgress=[InfectionProgress length(Infected)];
end
```

```
it2=0;


if k==1 %if we want to generate the full network
while length(Infected)<n && it2<10000 && day<6000
        it3=0;
        while( true ) && it3<maxit
                x=ceil(rand*n); %pick a new random node x to infect
                if sum(A(x,:))<DegreeMatrix(x,2)
                        break
                end
                it3=it3+1;
        end

        NewInfected=[x]; %add node x to NewInfected
        Infected=[Infected x]; %add node x to Infected

        %spread of infection
        while length(Infected)<n && it<maxit
                it=0;
                if ~isempty(NewInfected)
                        days_without_new_infection=0;
                        for j=1:length(NewInfected) %for each newly infected node
                        a=NewInfected(j);
                        FriendsOfFriends=[];
                        A2=A^2;
                        for i=1:n
                        if A2(a,i)>0
                                FriendsOfFriends=[FriendsOfFriends,i];
```

```
                end
                FriendsOfFriends=FriendsOfFriends(find(FriendsOfFriends~=a));
                end

                while sum(A(a,:))<DegreeMatrix(a,2) && it<maxit
                    if length(FriendsOfFriends)>0 && cp>rand
                        x=ceil(rand*length(FriendsOfFriends));
                        b=FriendsOfFriends(1,x);
                    else
                        b=ceil(rand*n);
                    end
                if (b ~= a) && (A(a,b) == 0) && (sum(A(b,:))<DegreeMatrix(b,2))
                    A(a,b)=1;
                    A(b,a)=1;
                end
                it=it+1;
                end
            end
        end

NewInfected=[];

        for i=1:n
            count=0;
            for j=1:n
                if A(i,j)==1 && ismember(j,Infected(:))
                    count=count+1;
                end
            end
            NumInfectedContacts(i,2)=count;
```

```
        end

        for i=1:n
        if ~ismember(i,Infected(:)) && 1-(1-beta)^(NumInfectedContacts(i,2)) > rand
                NewInfected=[NewInfected i];
                Infected=[Infected i];
        end
        end

        if days_without_new_infection>500
                'Infection has terminated.'
                break
        end

        days_without_new_infection=days_without_new_infection+1;
        day=day+1;
        InfectionProgress=[InfectionProgress length(Infected)];
        end
        end
end
```