NETWORKSIR AND ENVIRONMENTALSIR: TWO SIMPLE DISTRIBUTED MECHANISMS FOR MODELING EPIDEMICS

by

Madison Pickering

This thesis is dedicated to my partner

Rowan Quilty,

I would not be where or who

I am without you.

NETWORKSIR AND ENVIRONMENTALSIR: TWO SIMPLE DISTRIBUTED MECHANISMS FOR MODELING EPIDEMICS

by

MADISON PICKERING, BS

THESIS

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2021

ACKNOWLEDGMENTS

I would like to foremostly acknowledge my advisor, Dr. Venkatesan. I began puttering about your lab during the senior year of my bachelor's degree and simply never left; thank you for always having the time to answer my questions and provide feedback. Your constant belief in my abilities as well as focused, rigorous approach to things has truly shaped my life both as a researcher and person.

I would also like to thank everyone who showed an interest in this project and provided feedback, no matter the hour: Rowan Quilty, Anukriti Dey, Syra Abraham, and Stephanie Batch. This thesis would be significantly more incoherent, and likely unfinished, without your patient comments and support.

Finally, a thank you to Vitali and Sahar Loseau, for your guidance from my elementary school days to graduate school. I would have been lost without you.

November 2020

NETWORKSIR AND ENVIRONMENTALSIR: TWO SIMPLE DISTRIBUTED MECHANISMS FOR MODELING EPIDEMICS

Madison Pickering, MS
The University of Texas at Dallas, 2021

Supervising Professor: Subbarayan Venkatesan, Chair

The recent COVID-19 pandemic and its management has highlighted a need for new and innovative approaches to modeling epidemics. This thesis seeks to address some of those needs by providing a new framework for modeling disease using a distributed, network based approach, dubbed NetworkSIR. It then illustrates the potentials of this model by extending it to allow for disease to spread through the environment rather than exclusively person-to-person contact. This is of note as all known network-based epidemic models consider only the case in which infection spreads through direct human-to-human contact, or host-to-human contact. As “superspreader” events often result from disease spreading through poor environmental conditions, EnvironmentalSIR, while simple, delivers a powerful advantage over traditional modeling techniques by being able to capture these events.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

BACKGROUND

This chapter is divided into three main sections as the required background for epidemic modeling is fairly long. We maintain that a model is only useful if it is capable of capturing phenomena of interest. Therefore an understanding of what exactly *is* of interest is required for any discussion on model development or behavior.

1.1 Disease Spread

Prior to examining and developing a model that accurately captures the spread of disease in a population, it would be wise to examine how disease spreads at all. This will allow us to better evaluate the necessity of any assumptions made when modeling. Note however that this is by no means a complete description of how disease spreads. Epidemiology, the study of health and disease in a population, has been explored and refined since the 1800s; capturing all the nuance involved is fully out of the scope of this paper. [1] To narrow the scope, we will concern ourselves only with disease spread in humans, and only with factors which are typically examined when modeling disease spread.

1.1.1 Disease Spread in Individual Humans

Disease spreads to humans through some disease causing agent (a pathogen) which is acquired either from another human, from the environment, or from some intermediate host. [2] It has been noted that models used for examining human-to-human transmission of disease still hold when the method of transmission is from an intermediate host to a human. [3] As such, we will consider there to be two classes of disease spread: from an organism to a human, or from the environment to a human.

Once a human has been exposed to an infectious pathogen, there are a host of factors which determine if that individual actually becomes infected. Some factors are dependent on the infectious agent received, such as the quantity of the infectious material received (also called load, or dose), and the unique pathology of the disease (the mechanism by which it spreads and develops into an illness). Other factors are related to the individual in question, such as if they are immunocompromised, if they have received a vaccine, their immune memory, and the health of the organ that the infectious material effects are all factors. [2]

1.1.2 Disease Spread in Populations

Fortunately, many of the individual factors that effect the spread of disease (such as an individual's immune memory) tend to "average out" when one examines a population as a whole. [2], [11] Unfortunately, this does not make modeling of populations "easier", as interactions *between* humans now becomes an issue. That is, behavioral patterns of individual humans may cause certain sections of a population to be more likely to contract an illness than others. [1] For example, disease tends to spread more rapidly in schools and hospitals due to the high number of daily individual contacts each person has when they frequent those places.

These behavioral patterns elucidate one of the key requirements for an epidemic to occur: high population density. For an epidemic to occur, disease must spread within a population. That is, each individual must, on average, infect at least one other person. This value is typically noted as R(t), or the effective reproduction number, and states on average how many people can be infected by a single infectious person within a population at some point in time, t. A value of R(t) greater than one indicates that a disease is spreading, as individuals on average infect more than one person, while a R(t) value less than one indicates that the disease is "dying out" within the population.

[1] Because of this, the structure of a population is particularly of interest; networks depicting the topology of a given population are a valuable tool when modeling epidemics.

Naturally, the more individual contacts a person has, the more likely it is that they will infect someone. And, since a higher population density lends itself to a higher number of contacts per person, the higher the population density, the more likely it is that an epidemic—a rapid spreading of disease within a given population—will occur. In fact, as long as a population density remains below some threshold value, disease cannot spread quickly enough to become an epidemic. [2]-[4]

Because some areas of the population are more densely populated than others, identifying these areas is especially of interest to epidemiologists and public health agencies. That is, if one is able to identify densely populated areas, they will also identify areas which are particularly susceptible to an outbreak. From there, preventative measures can be taken, like ensuring adequate immunization rates within a population. The concept of achieving a specified immunization rate coupled with previous infection such that disease cannot spread (due to the very small number of susceptible contacts an individual has) is known as herd immunity. [4]

The environment of a population plays a large role in the spread of disease as well. Some diseases are unable to spread without certain environmental fixtures, which can be living or inanimate. For example, malaria requires the living presence of mosquitoes to transmit the disease to humans. In order to have a mosquito population, an area requires the inorganic features of standing water and warm temperature. Human made environments can similarly lead to opportune mechanisms of spreading disease. In the case of viruses spread by respiratory droplets like Severe Acute Respiratory Syndrome-CoronaVirus-2 (SARS-CoV-2) and Influenza, current research suggests poor ventilation systems may contribute to outbreaks. [22], [26], [27]

A key environmental factor to note is that many diseases can spread from poorly sanitized surfaces. For example, if infectious respiratory droplets come into contact with an inanimate surface, and a person touches that surface and followed by their mouth, that person may become infected. The ability of this indirect transmission to spread disease is based on the time the infectious agent or pathogen is able to survive outside of a host, and for the contact patterns of individuals themselves. [26] This follows naturally from the fact that an individual must come into contact with a contaminated surface before they can possibly become infected. Thus, the behavioral patterns of humans as they relate to visiting common environmental locations, like supermarket card readers, is especially of interest.

Lastly, movement between populations themselves is an issue. That is, it is entirely possible for people to migrate in and out of certain populations during the course of an epidemic. However, this factor tends to be ignored in practice as the effects of immigration and emigration tend to be unnoticeable as they relate to the speed at which a disease spreads within a population. Similarly, birth and death rates are often ignored for the same reason. [2] Despite these simplifying assumptions that never truly hold in practice, models which ignore migration patterns tend to have good performance regardless. [4], [11], [19], [21], [24]

1.2 Current Models

The most popular models tend to be compartmental in nature; that is, they designate members of a population as being within certain groups, or compartments. The SIR model, originally proposed by Kermack and McKendrick in 1927, tends to be the most widely used. [3] The model divides a population into three groups: those **s**usceptible, **i**nfected, and **r**ecovered or **r**emoved. For the remainder of this thesis, we will use ’S’ to denote the susceptible state, ’I’ to denote the infected state, and ’R’ to denote the removed state. The term recovered/removed includes those who died from the illness and is used as an encompassing compartment to denote those who are no longer able to spread the disease. Differential equations are then used to show how members of a population pass between those groups, moving from susceptible, to infected, to recovered/removed in that order. Note that once a node transitions to removed, there are no further transitions are possible.

Some key assumptions include the following: that it takes relatively little time for a disease to progress in a human compared to a human's lifespan (from which we derive the assumption that immigration and emigration from births and deaths can be safely ignored); that a population is held constant in size; and that humans have homogeneous mixing patterns. The term "homogeneous mixing patterns" refers to the fact that the model assumes that each human contacts other humans in the population randomly and uniformly. From section 1.1.2, we can see that the first two assumptions are relatively reasonable, while the latter is less so, but yields more tractable analysis.

The SIR model has found wide usage due to its simplicity and relatively high accuracy. It has inspired countless variations, such as the SIS, SEIS, and SIRD models (among others) which take into account those who are **e**xposed before becoming infected, and those who **d**ie in lieu of being recovered/removed. The similarity of these models to automata has inspired a host of cellular automata based models, such as [9]. In these models, each "cell" or pixel in a 2D grid is considered to be its own population. Each population then uses its physical neighbors as input (along with its own state) to determine its next state. These models extend the SIR model by explicitly modeling the SIR transitions through finite state machines and by including a very limited amount of non-random mixing at the population level. They have attracted limited attention however, perhaps due to their limited ability to model heterogeneous mixing.

Contact network based models, conversely, allow for heterogeneous mixing at the individual level [2], [10], [11], [14], [19], [21], [24] Humans are modeled as nodes and an edge exists between two nodes (u,v) if disease has spread from u to v or from v to u. Popular variations allow for nodes to represent people and links between them represent the *ability* for disease to spread between two nodes. The SIR model (and/or its variants) is then applied to the nodes themselves, and simulations are performed to examine the dynamics of disease spread at that individual level.

Much of the work developing contact network based models has been motivated by a desire to understand the effect of individual behavior, particularly because the early stages of epidemics are often the result of individual "superspreading" events. [11] Superspreading events refer to an individual infecting a statistically improbable number of other persons, typically leading to a sharp spike in cases. An example of such an event is that of "Patient 31", an individual who spread COVID-19 to approximately 70 others in South Korea; the outbreak was considered to have been under control prior to Patient 31's excursions, however the situation became difficult to control after the resultant cases. [12]

CHAPTER 2
CHALLENGES OF NETWORK BASED EPIDEMIC MODELING

There are, unfortunately, a host of challenges with contact network based epidemic modeling. The most fundamental is the lack of a unified network model which in turn makes comparing different network models intractable. For non-network models, the SIR model is most widely used and adapted, serving as a sort of common DNA between models and allowing for easy comparison between them. As such, most research agrees that a compartmental model like SIR should be used for network based modeling. However, there is very little overlap among research when it comes to how exactly SIR should be implemented in a network.

Despite variances in SIR implementation as they relate to networks, virtually all network based models rely on simulation to determine statistics of interest. Usually, a single infected node is introduced to a population of some fixed size and structure which results in an outbreak that is then measured. The obvious problem with this is that any simulation is highly dependent on the assumptions made when creating it. This includes assumptions about the structure of the population itself, as that will determine the rate at which individuals are infected, as well as the assumptions made when modeling the spread of disease.

2.1 Infection Dynamics Assumptions

The main issue with modeling the infection dynamics of the SIR model is that there is no standard way of representing those dynamics in a network. The SIR model itself is defined in terms of differential equations, however, differential equations are not so easily adapted to a network. Despite this, there are a few design choices which remain relatively ubiquitous: it is generally assumed that all nodes start as being susceptible and then transition based on the states of the nodes around them. Because all nodes start as being susceptible we only need to concern ourselves with defining the transitions from susceptible to infected as well as from infected to recovered.

1. Transition from S $\rightarrow$ I (Infection): Infection from one node to another is usually defined probabilistically. Each infected node has some probability to infect each of its neighbors while it is infected. However, the methods and equations that drive infection probability tend to vary from one study to another. For example, [21] only allows for individuals to become infected at a certain time (weekdays).

2. Transition from I $\rightarrow$ R (Recovery): The state transition from infected to recovery is equally problematic - most studies agree that it should be probabilistic, but the equations driving this probabilistic recovery also vary from study to study. Additionally, many researchers apply time thresholds on recovery. That is, they force nodes to recover after a period of time has elapsed, while other researchers do not impose this bound. Consequently, some models may allow an infected node to persist for arbitrarily long periods of time (albeit, with extremely low probability) while other models do not allow this behavior.

Finally, we note two other common sources of variance. Firstly, that most network based models choose to define compartments beyond the SIR model. They typically allow for at least a category for those who die from the disease, but frequently include significantly more. Each transition for these model-defined compartments tends to be defined uniquely, much as the transitions for infection and recovery. Secondly, since models are simulation based, they tend to execute their simulation with respect to some time unit. Typically, they allow for an individual to become infected with respect to some probability distribution once per time unit. However, there is significant variation in the length of these time units between models, resulting in a higher chance that an individual will become infected as the length of those time units decreases. This further exacerbates the difficulty of comparing between network models.

2.2 Population Structure Assumptions

Recall that network-based models reject the homogenous mixing assumption. Since this is motivated by a desire to adhere closely to the more realistic assumption of a non-infinite number of contacts for each person (which the SIR model does not make), one must then decide how many contacts each person within the simulation could and should realistically have. In short, this decision needs to be informed by reality. A realistic contact network, a type of graph which indicates who in a population is in contact with who, is the obvious choice. However, there are many such networks and the choice of what contact network to validate a model with is complex.

Ideally, the contact network data would capture exactly those interactions between humans that would be likely to cause infection. These are typically dubbed close proximity interactions (CPI) and represent interactions wherein individuals are usually 1-3 meters apart. This distance is usually sufficient for disease transmission. [27] However, real world contact network data that captures close proximity interactions (CPI) did not exist up until fairly recently. This is primarily because most CPI based contact networks tend to be captured using Bluetooth technology, which did not exist in the consumer market until 1999. It took additional time after that for Bluetooth to become popular and cheap enough to deploy on a large scale for studies. As a result, much of the preliminary work on network based models was created and evaluated using contact networks generated based on mathematical probability distributions (such as poisson and exponential random graphs), the Erdős–Rényi model, and scale-free networks. [11] While these graphs have interesting properties, there is no indication that they reflect realistic human CPIs.

That is not to say that realistic CPI networks are without their limitations. The Hawthorne effect describes the phenomena that humans behave differently when they know that they are being observed. Thus, these CPI networks are unlikely to capture true human behavior because all the participants know that they are in a study, and are likely to modify their behavior as a result. However, the alternative is that participants are unknowingly observed instead. This raises privacy and human rights issues, rendering covert surveillance an unviable option. We thus consider CPI-based contact networks to still be one of the best indicators of human behavior (and therefore ideal for modeling) despite this limitation.

Finally, note that the use of CPI based contact networks are highly vulnerable to the application domain: the same contact network should not be used to inform decisions about the spread of different diseases within the same population. This is because while CPIs record close-proximity interactions, the exact qualities of what constitutes a CPI is usually defined by whomever designs the experiment to capture the CPI data. That is, some CPI's may or may not be included based on how the experiment itself is designed. Consequently, one must pay close attention to the exact methods by which data were collected and ensure that it aligns with whatever one wants to model.

For example, consider a CPI network that models a student's social group in a high school. Data are recorded in this case by a student being within two meters of another student for some length of time, say, five minutes. In such close quarters, the school network may record a mean of 30-100 interactions per person. This network would be appropriate for modeling Influenza, but it would give an incredibly skewed perspective on how the Herpes Simplex Virus would spread in that same population. This is because the mean number of CPIs for a sexually transmitted infection (STI) in the same network is likely to be significantly smaller due to differences in definition of a CPI.

2.3 Issues with Validation and Comparison

Naturally, some criteria is needed to evaluate the performance of these models. A common metric used is the ability of the model to accurately predict the size of an epidemic. However, there exist a multitude of estimates for the final size of an epidemic. Furthermore, many of these statistics are accurate only for certain populations due to global variances in population density. For example, consider a statistic citing that roughly 10% of a population contracted a disease. That same disease may effect close to 30% of a population in a city, but only 5% of a population in a more rural location. As a result of these wide variances in values of the same statistic, it is possible to (misleadingly) claim that any model is accurate simply because it adheres to a valid statistic.

Being unable to clearly evaluate models based on epidemic size prediction accuracy necessitates some other metric of comparison. However, as mentioned previously, network SIR models tend to fully define their own methods of running simulations at virtually every step, making comparison between different models and different design choices near impossible. This is especially unfortunate because as engineers and scientists, it is imperative to have justifications for why certain design choices are being made. Understanding of design choices then leads to being able to improve upon methods and algorithms, driving innovation. We thus find that the creation of a common, extensible model is needed to further research in this field, at least for the sake of being able to evaluate different methodologies.

There is however one common design criteria inherent to all models: computational efficiency. As all these models are computer simulation-based, standard asymptotic analysis of time complexity is theoretically possible. However, there does exist a plethora of barriers that make it infeasible to do so. Perhaps due to the inherently interdisciplinary nature of model creation, rigorous algorithmic specification and analysis is usually omitted from model specification; exact descriptions of model implementation tend to be extremely scarce - source code in particular appears to be virtually absent. [11], [19], [21], [24], [25] Thus, an interested researcher must derive the time complexity of the best-case implementation of a model based on a small number of rules defined in the model's description. Furthermore, figuring out what the best-case implementation of a model is can be difficult to elucidate in and of itself. All of these features compound to make model comparison by asymptotic time complexity an impractibly time consuming affair.

2.4 Defining Common Model Criteria

As highlighted above, there are a host of issues with creating, evaluating, and comparing different models. It is our belief that creating better models must be done through careful and justified design choices. Thus, there needs to be some common criteria by which we can evaluate design choices, and thus analyze the performance of the models themselves. We propose the following criteria, ordered by importance:

1. Simple

   It is a common mantra that "all models are wrong", typically interpreted to mean that all models make simplifying assumptions that do not hold in reality. Many approaches to defining models attempt to reject as many simplifying assumptions as possible in order to adhere more closely with reality. However, this methodology is counterproductive. The real world is incredibly complex, which necessitates models; models provide an overly-simplistic, but useful, approximation of phenomena which is too complex to understand as it appears in nature. As such, we argue that a model should be as simple as possible.

2. Reasonably Accurate

   The simplicity of a model, while important, should not be secondary to accuracy. Rather, simplicity and accuracy should coexist as much as possible. After all, a model which is beautifully simplistic yet wildly inaccurate is useless because it does not provide a useful approximation of a complex phenomena.

3. Captures phenomena of interest

   A model must capture some phenomena of interest for it to be useful. If a model is optimized to collect a statistic with no use, then it simply doesn't matter how well the rest of the model is defined. We consider a statistic to be of no use if it is already known, or if it simply cannot be used for anything (like for example, the probability that a planet has useful resources given that there is no conceivable way of reaching it.)

   For another more nuanced example, consider an SIR model extended to an SIRD model, such that it differentiates between those who recover and those who die. Let it be known a priori that given a certain number of people who transition out of being infected, a certain percentage of them will die rather than recover. In this case, we consider the SIRD model as having failed to capture phenomena of interest since it defines an already known statistic: the number of people who will die rather than recover. Note that if we remove our prior knowledge, the SIRD model captures phenomena of interest. As such, this criteria is highly dependent on prior knowledge.

4. Efficient

   A model's implementation needs to be reasonably tractable in order for it to be used. As such, the parameters it depends on need to be available in some way, ideally with minor cost. Furthermore, any computation required by the model must be reasonably efficient. For example, we reject models which require multiple days or specialized equipment (e.g., supercomputers) to operate.

5. Available

   Ideally, a model's specification should be easily available and understandable. In the case of equation based models, a well written paper is usually sufficient. However, for algorithmic models, the algorithms used should be specified somewhere. Avoiding clear descriptions of model specification makes further work and verification by other researchers difficult to impossible.

CHAPTER 3

NETWORKSIR

Driven by the previously established need for a common, simple, and extensible network interpretation of the SIR model, we have constructed NetworkSIR. The general idea is to capture the state transitions of individuals through the movement of infectious particles themselves. The model is implemented as a distributed system. This lends itself to increases in computational efficiency as it eliminates the requirement that the model needs to be run on a single system. In the case that it is run on a single system, each process can be simulated using a discrete thread of execution, allowing one to take full advantage of multicore, multithreaded CPUs.

Furthermore, it is natural to describe the system in a distributed manner. A key feature of distributed systems is that information about the system is decentralized. As such, by modeling with a distributed system, we are able to explicitly highlight the fact that individuals are only able to make choices based on what they "see" around them, or what they "hear" from others. That is, an individual can only gain information by passing messages with their neighbors.

We begin with a more formal description of synchronous distributed systems.

3.1 Background: Synchronous Distributed Systems

The synchronous distributed system model is both elegant and intuitive. We describe it here using terminology more specific to the work of Lynch and Dolev [6]-[8], however the terms are general enough that anyone familiar with distributed systems can understand it easily.

A synchronous distributed system is modeled as a digraph, G = (V, E) in which each vertex $v \in V$ is a process. Each edge (u, v) represents a communication link (also called a channel or message buffer) between two processes. Each edge can contain at most one message, which may be of arbitrary size. The message alphabet M is finite in size, and we let M = M ∪ {null} such that we denote the absence of a message as null.

Associated with each process i, P_i, is the following:

1. $outneighbor_i$, the set of edges from which there is a link originating from i

2. $inneighbor_i$, the set of edges from which there is a link ending at i

3. $states_i$, a set of states which may or may not be finite

4. $start_i$, the set of possible initial states $msgs_i$, the message-generation function which maps $states_i$ X $outneighbor_i$ to the elements of M. That is, given i's current state, $msgs_i$ determines which messages should be generated and what links on which they should be placed

5. $trans_i$, a state transition function which maps $states_i$ in conjunction with the messages received from $inneighbor_i$ to arrive at the next state $s \in states_i$

Note that each process is implicitly modeled as a finite state automaton. As such, we allow halting states. When a process has arrived at a halting state, no further messages are generated and the only state transition is a self loop. However, in contrast to traditional automata, the concept of an accept state need not exist. That is, if a process halts, it may or may not be because it has reached an "ideal" state.

Execution of the system occurs in lockstep; each process P_i repeatedly performs the following two steps. We call the execution of these two steps a round.

1. Apply $msgs_i$ to the current state. Place all messages generated this way in the appropriate outgoing channels.

2. Remove all messages from incoming channels. Apply $trans_i$ to arrive at the next state

A configuration of a system c consists of the state of each process and all the messages in links at that time. The system starts in an initial configuration c_0 and moves to its next configuration c_1 after a single round r_1, $c_0 \rightarrow r_1 \rightarrow c_1$. An execution e of a system is a series of configurations and the rounds used to achieve them: $e = \{c_0, r_1, c_1, r_2, c_2, ...\}$. Viewing a system as a series of configurations and/or executions often provides valuable insights about the system, particularly as they relate to correctness, impossibility, and/or the ability of the system to recover from faults.

3.2 Description

As mentioned previously, the NetworkSIR model is a type of synchronous distributed system. The central mechanism driving infection is motivated by the agent model of rumor spreading proposed by Giakkoupis et al. [5] We make some changes to the model primarily to the effect of making rumor spreading applicable to epidemiology. That is, we introduce preferred links for agents to travel with such that all nodes do not have equal probability of being infected. We also allow agents to represent the pathogenic material, such that the exchange of information equates to infection, and information is exchanged when an agent meets with a node: the visit-exchange model of Giakkoupis et al. Thus, only infected nodes are allowed to generate and transmit agents.

Specifically, we propose the following:

Let the model be represented as a synchronous distributed system G = (V, E). All configurations of the system consist of processes that will engage in the simulation, a leader process, and a number of agents ≥ 0. The leader process is designed to simply receive statistics from nodes and log them to a file, as well as perform minimal analysis of the statistics received in order to determine if further infection in the system is possible. If no further infection is possible, the leader waits for all nodes to recover, and then notifies the system that it can terminate the simulation. Because the leader process is concerned with communication it must be connected to all nodes. Aside from that restriction however, the graph may be arbitrarily constructed.[1]

As mentioned earlier, information/infection is spread through agents. Agents are tokens containing information which pass from process to process, utilizing communication links to do so. In a single round, an agent is allowed to move or stay in their current process by utilizing an outgoing edge from the process they are in. When an agent meets with a node, information is exchanged between the agent and that node. In other words, the node becomes infected if possible; recovered nodes cannot become infected again.

Recall too that we make the modification that each communication link has a probability associated with it. This associated probability determines the probability that an agent will use that communication link to travel, and therefore equates directly to the probability that a node will infect one of its neighbors. We use a Uniform Random distribution in concert with a certain equation to generate probabilities, for reasons discussed in subsection "Mechanism by which Nodes Become Infected". [2]

As the leader process does not directly engage in the simulation, all links associated with the leader process have probability 0, and thus agents cannot travel using any edges connected to the leader. Note too that while we refer to agents being allowed to move, it is the agent's *host* process that actually performs the computations about where they will go. Consequently, the movement of agents can be thought of as an instance of applying the message generation function $msgs_i$.

[1]A simple modification is possible to allow for completely arbitrary graph topologies: let each node log its round statistics to its own personal file, then compile the statistics for the total simulation based on the results of those logs. This modification is not discussed further for the sake of keeping the discussion (and algorithms) simple.

[2]The mechanism behind edge-weighting may vary; other probability distributions (or MLE or MAP estimates) can be chosen instead for models extending this one. However, we impose a requirement upon them such as not to violate the second axiom of probability. Since each link is the probability that an agent will use that link to travel, the sum of the probabilities of all out-edges of a node and the probability that an agent will stay within that node (a self loop) must be equal to one.

As is customary, we model each process as a state machine. Each process has a state value associated with it which determines its behavior, including if it is allowed to exchange information with visiting agents. While the current run is active, the processes follow the following behavior as described by these state machines:

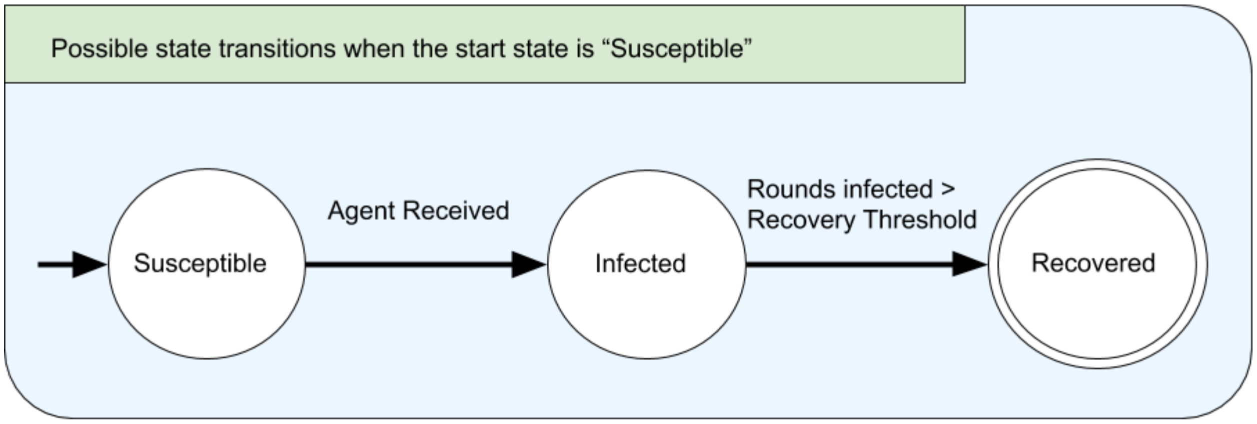
Possible state transitions when the start state is "Susceptible"
Susceptible
Agent Received
Infected
Rounds infected > Recovery Threshold
Recovered

Figure 3.1. State Diagram for Susceptible Nodes

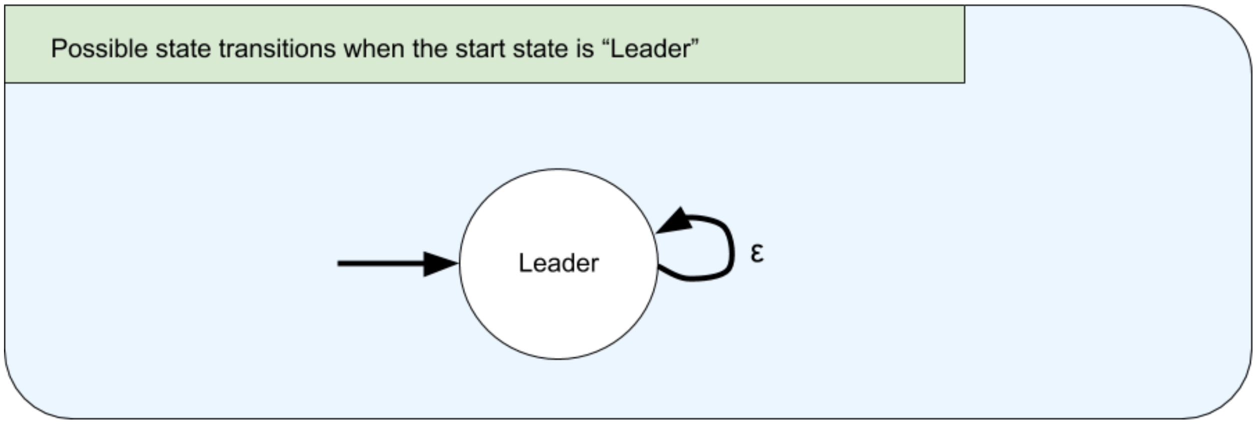

Figure 3.2. State Diagram for Leader Nodes

All processes either begin in one of the above states. If a process is susceptible, its state is allowed to progress to being infected upon receiving an agent. For the duration of being infected, if the process does not contain an agent, it generates an agent once per round. After a finite series of rounds - the "Recovery Threshold" - the process transitions to the Recovered state. If a Recovered process receives an agent, it deletes it and does not send it. Note that while the Recovered state is denoted as an accept state, the automata is still reporting statistics to the leader process each round. It is simply denoted as an accept state because at this point it is "removed" from the simulation; it does not forward any agents, and cannot transition to any other state.

In contrast, when the start state is Leader, the only state transition is to itself (a self loop). Here, the epsilon is used to denote the empty string. While the self loop is technically not needed (as all automata have such implicit self loops) it is shown here for emphasis. We additionally do not include a final state in the diagram, as Leader nodes never progress to a state in which their behavior changes. That is, they will always perform the same duties regardless of what messages they may receive or how long the run progresses for.

In this section we have described a number of design choices without regards to their motivation; that is remedied in the following section. Readers wishing to skip this may move to "Implementation", which gives pseudocode, analysis, and other implementation details of the NetworkSIR model.

3.3 Design Choices and their Motivation

There are two transitions in the SIR model: the transition from Susceptible to Infected, and the transition from Infected to Recovered. As such, we discuss the two mechanisms driving these transitions.

3.3.1 Mechanism by which Nodes Become Infected

There are two design choices to note which govern an infected node's behavior: the mechanism by which edges are weighted (and thus agents are sent), and the decision that an infected node should generate an agent once per round if it does not currently contain an agent.

Generating an Agent

The motivation for this choice is fairly simple. A node which is infected should always be able to infect other nodes while it is infected. Since infection is caused by the passing of agents, an infected node consequently always needs to have an agent to infect with in order to maintain that property. Thus, while a node is infected, it should always generate an agent if it does not already have one.

Weighting Edges

Ideally, a model should adhere to real life phenomena while avoiding unneeded complexity as much as possible. As such, we make the following simplifying assumptions:

1. Assumption 1 (A1): All nodes have the same behavior when it comes to infecting those around them. That is, each node behaves identically irrespective of the virus in question. Thus, a weighting mechanism that describes one node's behavior is suitable to describe all nodes behavior.

2. Assumption 2 (A2): The probability of infection correlates linearly and positively with the duration of a CPI with an infected individual. In other words,

   $$\text{P(infection)} = \text{k(CPI duration)}$$

   where k is some constant < 0. Intuitively, this equates to the assumption that for each second that two individuals are in contact, there is the same probability that one individual will infect another.

From the above assumptions, we conclude that if we are able to find some measure that governs the probability of the occurrence of a CPI of some set length we will similarly be able to determine the probability that one individual will infect another. Therefore, the problem of edge-weighting is reduced to the problem of determining the mechanism that governs CPI behavior. We approach the problem in two ways: by performing regression analysis on the data, and by performing a goodness-of-fit test to see if the data follows a probability distribution.

We use the contact network collected from a U.S. high school for our analysis. [19] The data set consists of over 700 individuals and the duration of all the CPIs they had in a day. A CPI within the data set is defined as an interaction between two individuals in which the individuals are within three meters (9 feet) of each other for a continuous period of time $\geq$ 20 seconds.

We begin by graphing[3] the data to help visualize the relationship between probability and CPI duration. The graph is listed below as Figure 3.3. More precisely, Figure 3.3 displays the probability (**per individual**, for every individual in the data set) that that individual will have an CPI of a given duration, given that they will have a CPI. Mathematically, the graph plots

$$P_i(\text{CPI with Duration X} \mid \text{CPI}) \quad \forall i$$

We then perform regression analysis. A Power Series appears to be an adequate measure of capturing the duration of a CPI, as indicated by the R^2 value on the below graph. While not shown, linear, logarithmic, exponential, and polynomial regression were also performed and yielded smaller R^2 values. However, there is still the question of if there exists a probability distribution which has a better fit. That is, can we do better?

[3]See the appendix for information on how to access the source code used to parse these data sets and generate the graphed statistics

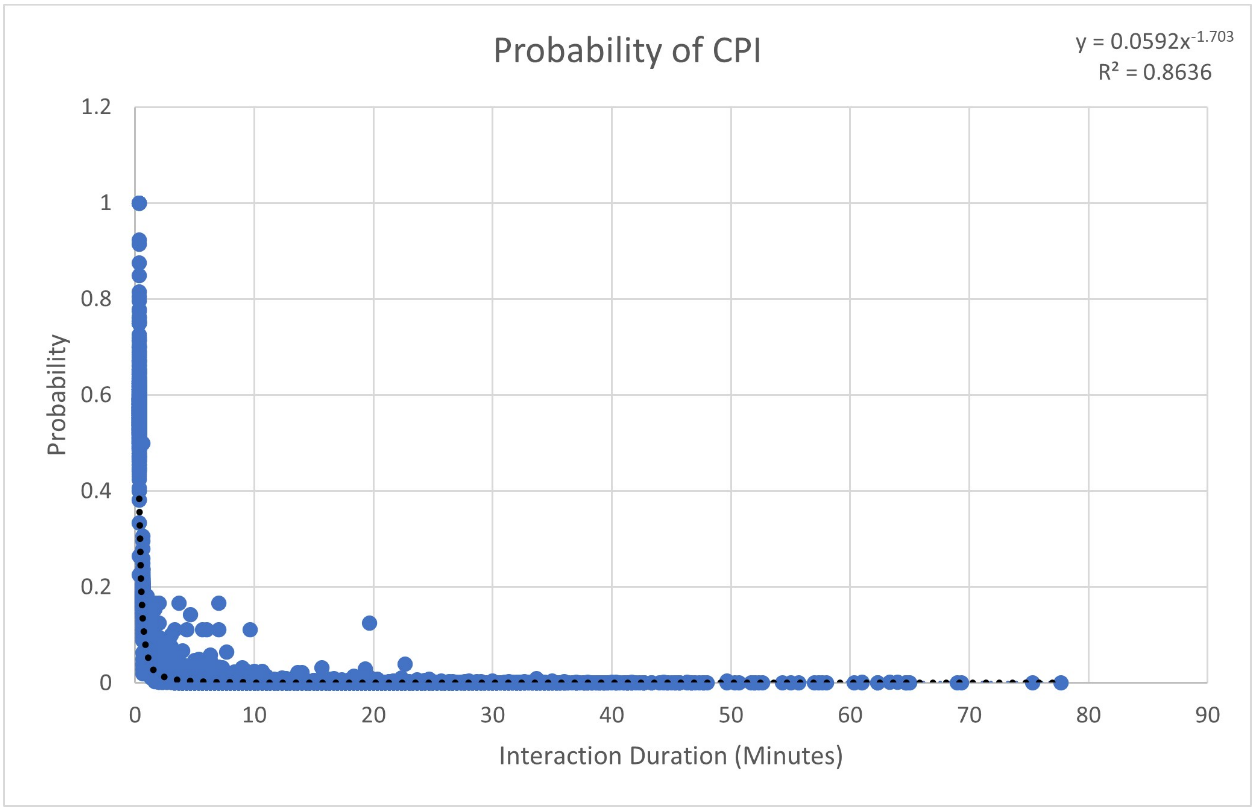

Figure 3.3. Probability of Individuals Having a CPI of a Given Duration

From the above graph, it is easy to see that humans do not have uniform CPI durations; humans are less likely to have long CPIs. That is, a given human is less likely to have a very long CPI (such as one that would span hours) versus a short one which might be obtained by passing by someone in a street. Thus, the probability distribution that governs CPI duration is not a uniform probability distribution. The shape of the above graph indicates P(CPI of duration x | CPI) most closely appears to fit a geometric distribution, and most certainly does not fit a Gaussian/Normal, Poisson, or Binomial distribution. For additional clarity, we include a graph of a single individual's interactions (below) to indicate that the interactions of an individual follow the shape of the graph for all individuals. While the choice to graph Individual 3's CPIs is chosen arbitrarily, this pattern appears to hold for all individuals. Thus, we can begin a test for goodness-of-fit.

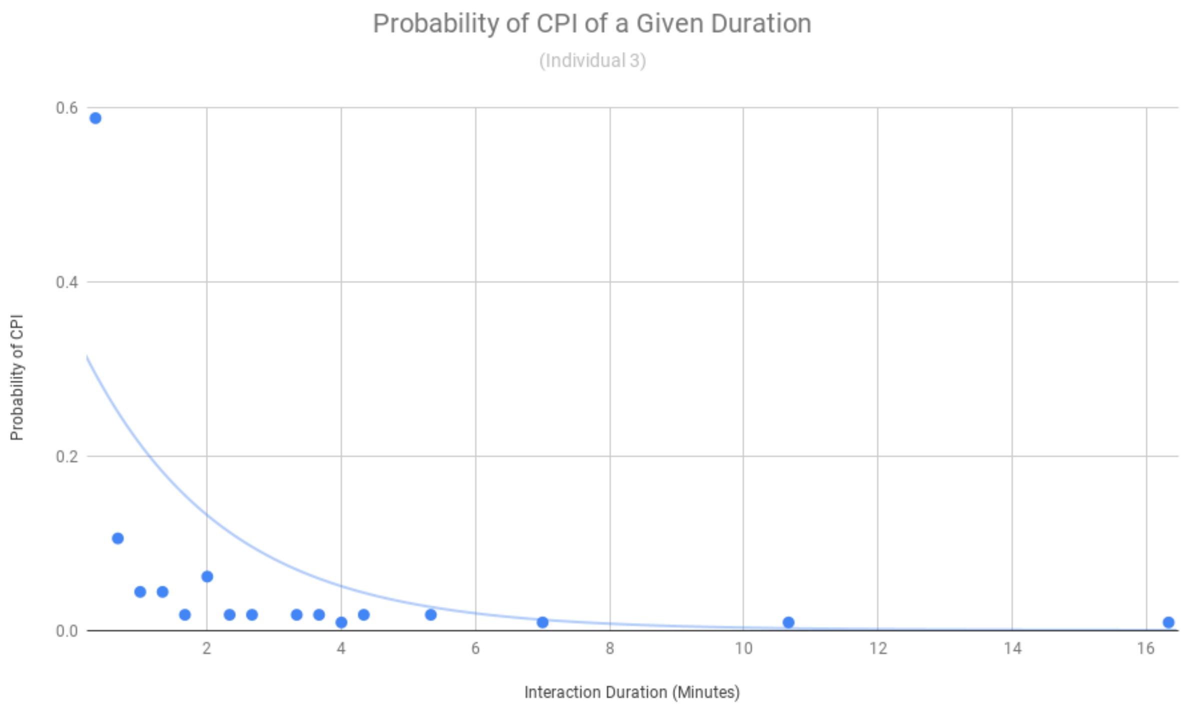

Figure 3.4. Probability of a Single Individual Having a CPI of a Given Duration

In order to determine which probability distribution is likely to fit this data, we use Pearson's Chi Squared Test [31] to determine goodness of fit. As above, we let each individual's list of CPIs be a random variable, and perform the test for each individual. Since Pearson's Chi Squared Test determines how well a distribution conforms to a theoretical distribution, we require a probable theoretical distribution. We choose the Geometric distribution due to the characteristics of the data. Our justification is as follows:

Geom(P): A Geometric random variable gives the probability of a certain outcome occurring given the fact that the outcome did not occur the previous k times. A geometric random variable requires the two following assumptions to hold, both of which are roughly satisfied:

1. Each result can be represented by a Bernoulli trial

   This can be done very neatly; the probability space only allows for two values, 0 or 1. We can map 0 to the occurrence of two people having a CPI, and 1 to the occurrence of two people not having a CPI. Thus, a CPI of duration n is equal to n results of 0, followed by a result of 1. Ergo, a CPI of duration n has n+1 Bernoulli trials.

2. The probability of a given result is independent of the previous results [4]

   This equates to the assumption that a person's decision to continue to be in proximity with another isn't influenced by the fact that they were in proximity with that same person for the past 20 seconds. This is likely to hold if the majority of a person's CPI are with strangers (and as such the two are likely to not socialize for an extended period of time) or if individuals are usually in proximity with each other by coincidence. In a crowded high school, this appears to hold.

We thus formulate our null hypothesis (H0) and hypothesis (H1) for individual i to be

$$\text{H1} = \text{individual i's CPI durations are not Geometric Random distributed}$$

$$\text{H0} = \text{individual i's CPI durations are Geometric Random distributed}$$

Since Chi Squared tests are vulnerable to mispredictions when the number of occurrences of an outcome are small, we adopt the common practice of not counting occurrences with incredibly small frequency. [28], [29] We use frequency > 3 as our threshold value for this. We

[4]While this may not actually hold, many discrete probability distributions (such as the Poisson and Binomial distributions) require independence. The Gaussian/Normal distribution similarly, if obtained by applying a formulation of the Central Limit Theorem, requires independence as well. As such, we may have to "settle" for adopting this assumption.

then test to ensure that our sample size is significantly large. We avoid testing individuals who have less than 250 interactions recorded after purging those data points which have frequency < threshold frequency. There are 622 such individuals in the data set after purging.

The above gives us our observed frequencies. To generate the ideal frequencies, we use Maximum Likelihood Estimation to generate the requisite parameters. We only need to generate a single parameter, as Geom(p) only takes a single parameter. That parameter, p, is estimated (denoted $\hat{p}$) as follows:

$$\hat{p} = \frac{n}{\sum_{i=1}^{n} m(i)}$$

Where m is the number of Bernoulli trials for the data point and n is the number of CPIs. Intuitively, $\hat{p}$ is equal to the proportion of successful trials over all trials performed.

Next, we must determine the degrees of freedom. The degrees of freedom are given as k - j - 1, where: k is given as the number of CPIs of some duration of which there are at least four instances (as given by the threshold value) of individual i having a CPI of that duration; j is given as the number of parameters of the hypothesized distribution estimated using sample statistics. Thus, there are k - 1 - 1, or k-2, degrees of freedom as only one parameter is estimated using sample statistics.

We then calculate χ^2 using the observed and ideal/expected frequencies. From there, we use a lookup table of ideal χ^2 values based on the P value and degrees of freedom to determine if we can accept or reject the null hypothesis H0. [30] We tested with P value = 0.001. All tested individuals rejected the null hypothesis by a very wide margin: the calculated chi squared value was typically 50-100 units over the ideal. We thus conclude that P(CPI of duration x | CPI) does not fit a Geometric distribution.

Since we found that a Geometric distribution fits the best based on the characteristics of the data, it is unlikely that another probability distribution would fit the data well. We then turn to regression analysis for our method of weighting edges. Specifically, we propose the following:

1. Generate some probability using a Uniform Random distribution.
2. Use the equation given by regression analysis to find the associated CPI for that probability. The equation given was $y = 0.0592x^{-1.703}$, where y is the probability and x is the CPI duration. Rounding to the hundredths place and solving for x yields $x = 0.06y^{\frac{1}{1.7}}$
3. Weight the edge using the duration of the CPI calculated in step 2
4. Normalize edge weights

Such a method generates edge weights in proportion to CPI duration, where CPI durations are generated according to the probability of an individual having a CPI of that duration. From assumptions A1 and A2, we then argue that this method is a good approximation to the probability that an individual will infect another, and thus is an appropriate method of edge-weighting.

3.3.2 Mechanism by which Nodes Recover

We let nodes be infected for a certain and fixed number of rounds, which we denote the Recovery Threshold. This is motivated by the following:

1. Randomization is already present in the model by way of infection and graph structure, so adding an additional function to determine if a node recovers in a certain round may simply excessively increase the complexity of the model.[5]

2. Using a Recovery Threshold prevents infected nodes from persisting for arbitrarily long lengths of time.

3. The parameter of "Recovery Threshold" is naturally captured. It is common practice to monitor symptoms of individuals when they are sick, and consequently the average duration that individuals are sick for (the Recovery Threshold) is similarly captured. Therefore, there is no need for the model user to calculate this value as it is provided by hospital staff.

Lastly, we also require that Recovered nodes remove all agents they receive. This is to ensure that recovered nodes do not send agents (infect other nodes).

3.4 Implementation

The simulation as a whole requires the following parameters:

1. G = (V, E): the structure of the graph, specified as an adjacency list.

2. P(Stay): The probability that an infected node will ***not*** infect someone once per round

[5]This is because variations in recovery time are largely dependent on the individual. That is, a good recovery function would likely have to take into account many parameters, such as age, past medical history, and quality of healthcare available.

3. Recovery Threshold: The number of rounds a node will stay infected for

Parameter 1 is a function of the population, while parameters 2 and 3 are a function of the disease being modeled. Parameter 3 should be provided by healthcare workers as an epidemic develops, and Parameter 2 can be estimated from R0[6] values, or determined by trial and error based on the final size of the epidemic and what is known about the disease. Note that the parameters which govern the behavior of how infection is spread are functions of the disease in question, not the contact network. Thus, once the parameters 2 and 3 are known, no further computations are needed to run the simulation with different graph topologies – a difference from many traditional network based models.

The simulation then proceeds as follows:

1. A graph is constructed using the edges and nodes specified in an adjacency list

2. A leader node is connected to all vertices

3. Each node's edges are weighted/assigned probabilities

4. All nodes begin computation

Steps 1-4 all together takes linear time: Step one takes O(N + E), Step 2 takes O(N), Step 3 takes O(E), and Step 4 takes O(N) time. Thus, simulation initialization takes time O(N + E).

[6]Where R0 is the basic reproduction number.

The weighting of edges proceeds as follows:

Algorithm 1: Weighting Edges

```
for Each node $v \in V$ do
    outEdges = v.outEdges;
    /* Set edge weights */
    double totalWeight = 0;
    for Each outEdge $e \in outEdges$ do
        y = some probability generated using a Uniform Random distribution;
        /* Find the associated CPI 'w' for that probability */
        w = $0.06 * y^{\frac{1}{1.7}}$;
        /* Weight the edge */
        e = w;
        totalWeight += w;
    end
    /* Normalize Edge Weights */
    for Each outEdge $e \in outEdges$ do
        e /= (totalWeight);
        e *= (1 - P(Stay));
    end
end
```

To make message passing easier, we then assign each edge a discrete probability interval, which corresponds to the weight of the edge. Thus, if there are two out-edges with normalized probabilities 0.3 and 0.1, and P(stay) = 0.6, the out-edges would be associated with intervals (0.6, 0.9] and (0.9, 1] respectively. This takes O(E) time overall.

As specified in the synchronous distributed system model, in each round a process receives messages, then transitions to its next state. The code for a process P_i reflects this:

Algorithm 2: Code for Process P_i

```
Boolean continueSimulation = true;
while continueSimulation do
    receiveMessages();
    transition();
    reportStats();
    thisRound++;
    Message m = receiveMessageFromLeader();
    continueSimulation = m.continueSimulation;
end
```

There are the following methods to expand upon: receiveMessages(), transition(), and reportStats(). Method receiveMessageFromLeader() simply receives a message from the leader process, and therefore takes time O(1).

1. **receiveMessages()**

The node simply waits until it has received messages from all of its neighbors for the current round. If the message is non-null, it places the message in a special queue, denoted "agents", which is a list of all the agents currently residing in this node. This takes time O(E'), where E' is the indegree of a node.

2. **transition():** The node transitions as follows:

Algorithm 3: transition()

```
/* check to see if we have received an agent - if so, change our state accordingly
   */
if (myState == SUSCEPTIBLE) then
    if (agents.isEmpty()) then
        return;
    else
        myState = INFECTED;
    end
end
/* By definition, recovered nodes discard all agents they receive                 */
if (myState == RECOVERED) then
    agents = new Queue();
    return;
end
/* else, we are infected.  Check to see if we are going to recover this round    */
sickCounter++;
if (sickCounter > RECOVERY_THRESHOLD) then
    state = NodeState.RECOVERED;
    return;
end
/* if we are infected and have no agents, generate one                            */
else if agents.isEmpty() then
    Agent agent = new Agent();
    agents.add(agent);
end
sendMessages();
```

This takes time O(1) + O(sendMessages())

Algorithm 4: sendMessages()

```
/* if we have no outgoing edges, each agent we receive stays...            */
if (outEdges.size() == 0) then
    return;
/* associate each agent with a probability, and sort                        */
for (int i = 0; i < agents.size(); i++) do
    agents.get(i).setProbability(Random.nextUniformRandom());
agents.sort();
/* determine which agent will use which link                                */
List<Agent> staying = new List<>(); int agentPtr = linkPtr = 0;
while (agentPtr < agents.size()) do
    Agent thisAgent = agents.get(agentPtr); Link thisLink = outEdges.get(linkPtr);
    double probability = thisAgent.getProbablity();
    if (probability > thisLink.getProbHigh()) then
        linkPtr++;
        /* if no more links, the remaining agents must be staying           */
        if (linkPtr ≥ outEdges.size()) then
            for (int i = agentPtr; i < agents.size(); i++) do
                staying.add(agents.get(i));
    else
        /* (probability ≤ thisLink.getProbHigh())                           */
        agentPtr++; msgsSent++;;
        thisLink.sendMessage(thisAgent);
/* the current list of agents should be only those that are staying        */
agents = staying;
```

The above code takes time O(AlogA + A), where A is the number of agents a node has received. This begs the question, how large can A get?

The total number of agents in the system, A', can increase if an infected node P_i sends all of its agents to another node P_j (or nodes) where the destination of those agents is not Recovered or will not change its state to Recovered in the next round. Since the total number of agents will increase the most if all infected nodes send their agents to a single node, we consider only the situation where P_j is a single process. We consider the case that P_j is susceptible, and P_j is infected separately for ease of calculation.

Case 1: P_i sends an agent to P_j, where P_j is susceptible: In this case, P_i will generate a single agent. In the worst case, all the infected nodes are able to generate an agent by sending all of their agents to the same susceptible individual. Thus, the number of infectious individuals (per round) similarly increases by one per round, and A' increases by I for each round that there exists a susceptible individual. This yields the equation

$$\text{A'}(P_j == \text{Sus}) = \sum_{i=1}^{S}(I+i)$$

Where I is the number of Infected nodes and S is the number of Susceptible nodes.

In the worst case, there is only one infected node initially, and all the rest of the nodes are susceptible. This then reduces to:

$$\text{A'}(P_j == \text{Sus}) = \sum_{i=1}^{N-1}(1+i)$$

Case 2: P_i sends an agent to P_j, where P_j is susceptible: Since, from above, the number of infected individuals increase by one per round, they will similarly decrease by one per round after the earliest infected node passes its recovery threshold. Before then, A' will increase by I - 1 each round. (The minus one is included because an infected node containing agents will never produce an agent). Recall too that in the worst case, all nodes at this point are infected, making the term equal to N-1 per round. Since the recovery threshold is constant, this term similarly reduces to kN, where k is the number rounds after all nodes are infected and before the first infected node recovers.

Recall from Case 1 that in the worst case the number of infected individuals increases by one per round. Because the recovery threshold is held constant, the number of infected individuals similarly must decrease by one per round. This is equivalent to

$$\text{A}(P_j == \text{Inf}) = \sum_{i=1}^{I}(I-1)$$

Or,

$$\text{A}(P_j == \text{Inf}) = \sum_{i=1}^{N}(N-1)$$

Combining Case 1 and Case 2, we then get a worst-case maximum size for A' (and thus for A) of O(N^2). It is worthwhile to note, however, that the worst case is only possible under certain graph topologies. That is, this is only possible given a completely connected graph. Furthermore, even if there is a complete graph provided, the statistical chance of agents being sent in this highly coordinated manner is approximately 0.

It is worthwhile then to consider a typical case. From our evaluation, we found that the average number of agents persisting in the system at any point in time t (A'(t)) is considerably lower than N^2. Running NetworkSIR 500 times on a contact network of 789 nodes taken from a U.S. highschool, with reasonable parameters approximated from influenza statistics yielded a maximum (average) A'(t) of 16, which is much, much less than even N.

Thus, the worst case time complexity for sendMessages is O($N^2 log N^2$), however it is highly improbable that the worst case would be encountered, both probabilistically speaking and with regard to realistic contact network graph topoglogies. So as to generalize time complexity appropriately, we then choose to denote the worst case time complexity as O(A'logA')

3. reportStats()

This takes time O(1), since it involves sending a single message to the leader node containing information about the node's activities in the last round, such as if it changed its state or not.

Overall Complexity

The overall time complexity for a process P_i is O(A'logA' * R) where R is the number of rounds executed.The largest value that R can take, R', is equal to (Recovery Threshold - 1) * N rounds, in the case that the simulation proceeds as a single chain and each process sends an agent on the round right before it recovers. Assuming the graph is completely connected and the simulation proceeds this way, we can obtain overall time complexity of O(N^2LogN^2*R'), which is approximately O(N^3logN^2).

The overall message complexity of the system is equal to O(E*R). This is because in the worst case, each node is infected and sends an agent on a different link. More specifically, the worst case message complexity is equal to O((A' + N) * R), where A' is the total number of agents in the system. The N term comes from the fact that the leader process receives a message from every node each round.

3.5 Evaluation

We evaluate our performance on three metrics: ability to capture SIR dynamics despite having the simulation take place using a network, ability to accurately predict the size of epidemics, and evaluation on other criteria as proposed in “Defining Common Model Criteria”.

3.5.1 Capturing SIR Dynamics

A key purpose of developing NetworkSIR was to be able to capture SIR dynamics at the individual level in a network. We use two contact networks of varying sizes to indicate that these dynamics are captured, regardless of network size: Zachs Karate Club, consisting of 34 individuals, and a CPI network taken from a U.S. high school containing 789 individuals. [15], [19] For the US highschool, we let the starting infection size be five individuals, chosen randomly, and P(stay) = 0.8. We let the Recovery Threshold be 30 rounds for the sake of capturing asymptotic behavior. The resulting graph is as follows:

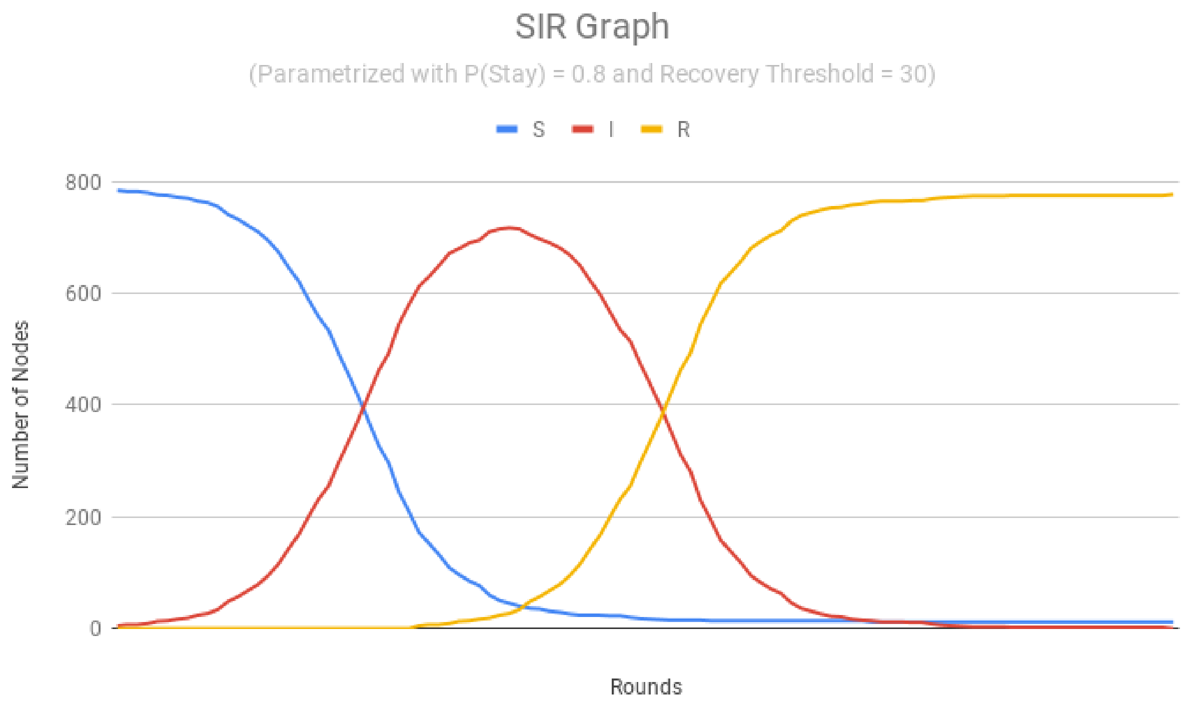

Figure 3.5. Generated "Ideal" SIR Graph

The above graph is the result of an arbitrary but representative run. The graph appears to be roughly identical to that of an ideal SIR graph, with the minor change that the infected curve is symmetric. In an ideal SIR graph, the infected curve tapers off much more slowly. This implies that (in an ideal SIR graph) as time passes, people remain infected for longer periods of time. This assumption does not hold in reality without outside influence; holding all else constant, there is no reason why infected individuals would recover at slower rates as time passes. As such, we feel comfortable relaxing the requirement that the infected curve be asymmetric. Furthermore, the graph appears to capture the asymptotic behavior of an SIR graph, the behavior of the graph leading to the development of the infection curve, and maintains the important property that the number of people in the network at any point in time stays constant despite transitions between states. We therefore claim that NetworkSIR is able to provide a good approximation of an ideal SIR graph despite the difference in infected curves.

We then follow by running NetworkSIR on Zach's Karate Club contact network. The parameters P(stay) and Recovery Threshold are again chosen arbitrarily, but such that the asymptotic behavior can be captured. We make the change that only one individual is initially infectious, in order to ensure that there isn't too much difference proportionally between the high school contact network and Zach's Karate Club. The resulting graph is as follows:

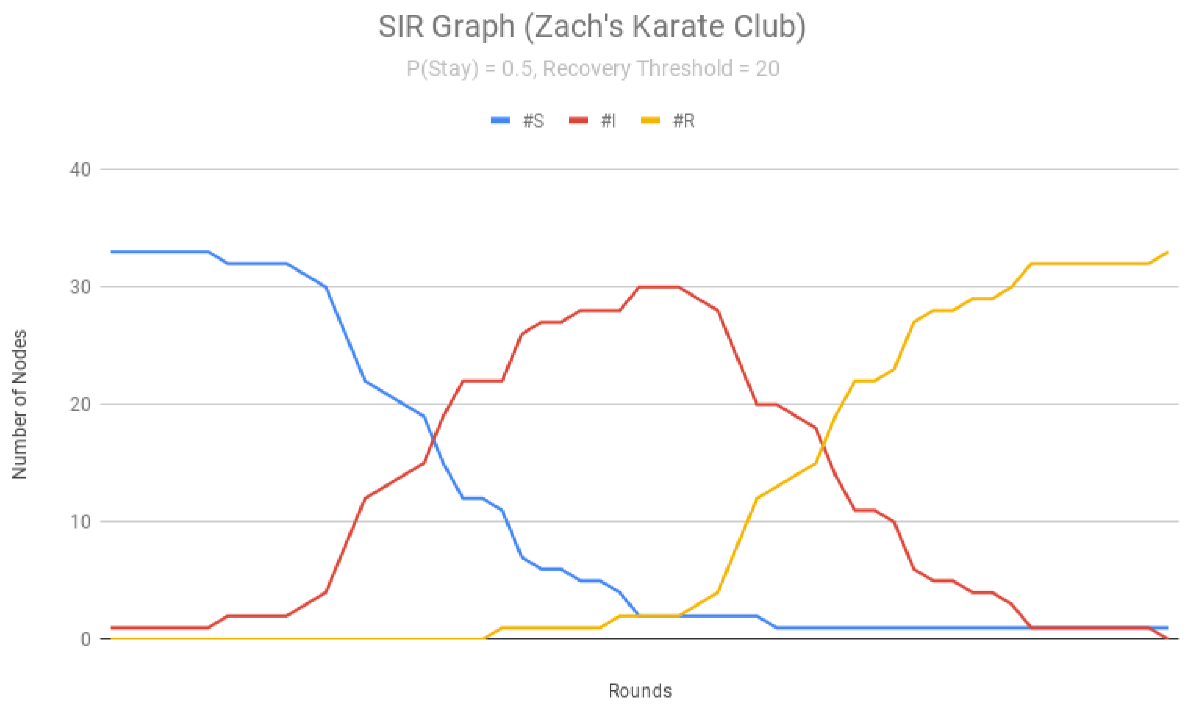

Figure 3.6. Generated "Ideal" SIR Graph on a Small Sample

While the above graph is clearly more "blocky" in appearance due to the small number of nodes, we maintain that SIR behavior is still captured. Thus, we conclude that NetworkSIR provides a good approximation of SIR behavior at the individual level, regardless of sample size.

3.5.2 Accurate Predictions

We performed our evaluation with the goal of producing data in-line with known statistics regarding seasonal (yearly) influenza. This has been done because of the wealth of information about influenza, and the fact that it is spread through respiratory droplets much like the current COVID-19 epidemic. It is our hope that good performance when modeling influenza will translate to good performance when modeling COVID-19.

Recall that the parameters needed to run a simulation include a graph topology, Recovery Threshold, and probability that an individual will not infect another person, per round (P(Stay)). Our graph topology has been parsed into an adjacency list from the contact network of [19]. Our Recovery Threshold has been parameterized by data from [16] and [17] by taking averages of the minimum and maximum duration to be sick for, and then averaging the min and max to yield an average recovery threshold of 5. We use P(stay) = 0.78. Running 500 simulations with these parameters yield an average epidemic size of 15.2%. This begs the question: is this an accurate prediction?

Data from the NIH indicates that the worldwide average epidemic size is approximately 9% of a given population, while in the US it tends closer to 20%. [18] Data from the WHO similarly gives an estimate of 5-15%. [17] Because schools are notoriously densely populated, and the data in question was taken from a U.S. high school, we consider an ideal epidemic size to be approximately 10-20%. We give such a wide range for an ideal epidemic size because of the wide variances in estimates. These estimates vary based on country, year (since we are modeling seasonal influenza), and study. NetworkSIR's prediction of 15.2% then falls into our proposed range quite nicely; we deem the simulation reasonably accurate.

While this simulation shows that NetworkSIR is able to make accurate predictions on a single contact network, it would be ideal to show that the model is versatile enough to make accurate predictions with the same parameters on a different network. As such, we run the model 500 times on Zach's Karate Club contact network, which consists of only 34 individuals. To ensure that the initial number of infected is proportionally similar, we allow for only a single individual to be initially infected.

Despite the small sample size, and the fact that the total number of edges is only 78 (as opposed to the high school network, where a single individual is likely to have over 78 out-edges), NetworkSIR is still able to achieve good performance. The average epidemic size is 11.7%, which is well within the given bounds. Furthermore, the fact that it decreased slightly from the high school network is ideal, as the population density of the club is significantly less than that of the high school network.

3.5.3 Evaluation According to Proposed Criteria

In "Defining Common Model Criteria", we highlighted some ideal metrics for comparing models. To round out our evaluation, we consider those metrics now.

1. Simple: Randomization in NetworkSIR is introduced at exactly one point (infection), keeping the simulation stochastic while minimizing complexity. Furthermore, the simulation itself takes minimal parameters: a graph topology, a recovery threshold, and the probability that an individual will not infect another (P(Stay)).

2. Reasonably Accurate: From the above two sections we conclude that the model appropriately captures the behavior of the SIR model and predicts the size of an epidemic with reasonable accuracy.

3. Captures phenomena of interest: By providing a model which captures population dynamics while similarly allowing for analysis of individuals, we conclude that NetworkSIR captures phenomena of interest. Furthermore, by explicitly modeling the movement of infectious material, NetworkSIR elucidates exactly how people become infected.

4. Efficient: We consider the algorithm itself, as discussed in Implementation, to be reasonably computationally efficient. We also consider the parameters to be relatively easy to obtain. There are many contact networks, and if there is a perceived absence of one, good approximations can be generated. [11] Recovery thresholds are relatively easy to obtain as well, since they tend to be captured as a natural part of symptom monitoring. Discovering P(Stay) is more difficult, but can be computed relatively easily if the R0 value and recovery threshold is known. If R0 is not known, but early statistics about the size of an epidemic are, P(Stay) can be approximated by running the simulation and minimizing the difference between the computed epidemic size and that indicated by early data.

5. Available: This thesis describes NetworkSIR in a wealth of detail, and is free to the public. All source code used to generate this data is similarly open-source. (Details describing access and use to source code is available in the appendix.) As such, we consider NetworkSIR to be appropriately available.

CHAPTER 4

ENVIRONMENTALSIR

A key advantage of modeling disease spread explicitly through agents is that it allows one to examine how commonly visited locations contribute to the spread of disease. That is, by making a small change to NetworkSIR, we can allow agents to reside in special "Environment" nodes. Sanitation can be simulated by removing agents after some amount of time has passed, denoted as the Sanitation Threshold. Thus, by increasing or decreasing the Sanitation Threshold, the effect of more regular sanitation can be measured. We begin with a more precise discussion of EnvironmentalSIR, then end with a discussion of the results.

4.1 Description

EnvironmentalSIR is identical to NetworkSIR except for the fact that we allow for an additional initial state for nodes, called "Environment". The state transitions of Environment nodes is as follows:

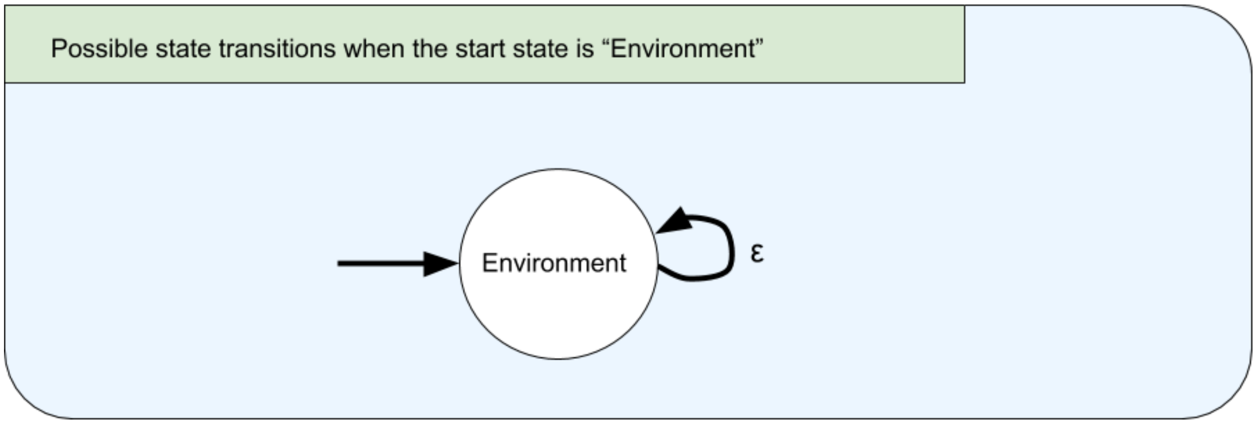
Possible state transitions when the start state is "Environment"
Environment
ε

Figure 4.1. State Diagram for Environment Nodes

From the above, it is clear that Environment nodes do not change their state throughout the course of any given run. While their state is "Environment" they send any agents they have received, and remove the agents within their node once their Sanitation Threshold has been reached. The code for an environment node is as follows:

Algorithm 5: Code for Process P_i, Where P_i.State == Environment

```
/* Maintain a list of received agents                          */
Agents = new list();
/* Maintain a counter, cleaningCounter, initially zero         */
int cleaningCounter = 0;
boolean continueSimulation = true;
while (continueSimulation) do
    receiveMessages();
    transition();
    reportStats();
    thisRound++;
    Message m = receiveMessageFromLeader();
    continueSimulation = m.continueSimulation;
```

Functions receiveMessages() and reportStats() are identical to that of NetworkSIR, and thus will not be elaborated on. The code for transition() is as follows:

Algorithm 6: transition() for Environment Nodes

```
if (cleaningCounter > SANITATION_THRESHOLD) then
    cleaningCounter = 0;
    Agents = new list();
sendMessages();
```

The overall time complexity and message complexity of EnvironmentalSIR is identical to that of NetworkSIR. This is because the code that is unique to EnvironmentalSIR all contributes a constant (O(1)) factor.

4.2 Evaluation

We avoid evaluation by the criteria proposed in “Defining Common Model Critera” because EnvironmentalSIR is so similar to NetworkSIR, there would be very little new information to discuss. Furthermore, evaluating the ability of EnvironmentalSIR to accurately predict epidemic size becomes difficult due to the fact that it is currently unknown to what effect commonly visited locations contribute to the spread of disease. As such, we concern ourselves mostly with attempting to identify trends between the number of Environment nodes, the Sanitation Frequency, and the overall size of the resulting epidemic.

We begin by comparing the results of running EnvironmentalSIR with the same starting parameters as NetworkSIR when predicting influenza epidemic size. That is, we run the simulation 500 times on the high school contact network, with P(Stay) = 0.78 and Recovery Threshold = 5. The initial number of infected individuals remains 5, but we now randomly set 10 nodes to be Environment nodes[1], with Sanitation Thresholds of 2 rounds. This equates to a final average epidemic size of 14.8%: a reduction of 0.4% from when there were no Environment nodes.

Increasing the Sanitation Threshold to 5 rounds and running the simulation an additional 500 times yields an identical average epidemic size. However, there were frequent cases in which agents were allowed to persist and cause small, secondary outbreaks with this increased Sanitation Threshold. The below graph was taken from one of these runs, and illustrates this:

[1]This implicitly assumes that humans visit environments with the same frequency/behavior that they visit humans. We avoid evaluating the veracity of this assumption due to scope of the experiment.

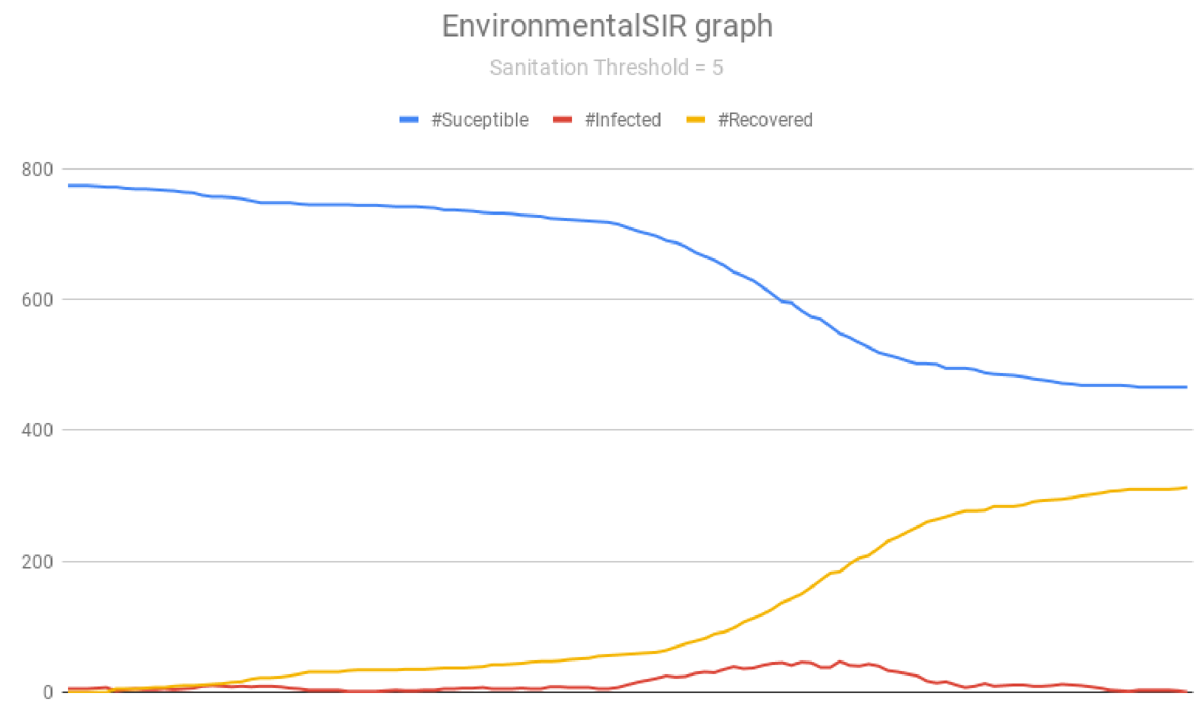

Figure 4.2. Secondary Outbreak When Sanitation Threshold = 5

The number of those infected approaches 0 early on in the run, yet a secondary outbreak occurs later on in the run which takes awhile to peter out. It is notable too that the infection curve is no longer symmetric, instead having an appearance closer to that of an ideal SIR Infected curve. Increasing the Sanitation Threshold to 10 yielded an average epidemic size of 14.6, but exacerbated the issue of multiple outbreaks. A graph depicting one of the runs with multiple outbreaks is as follows:

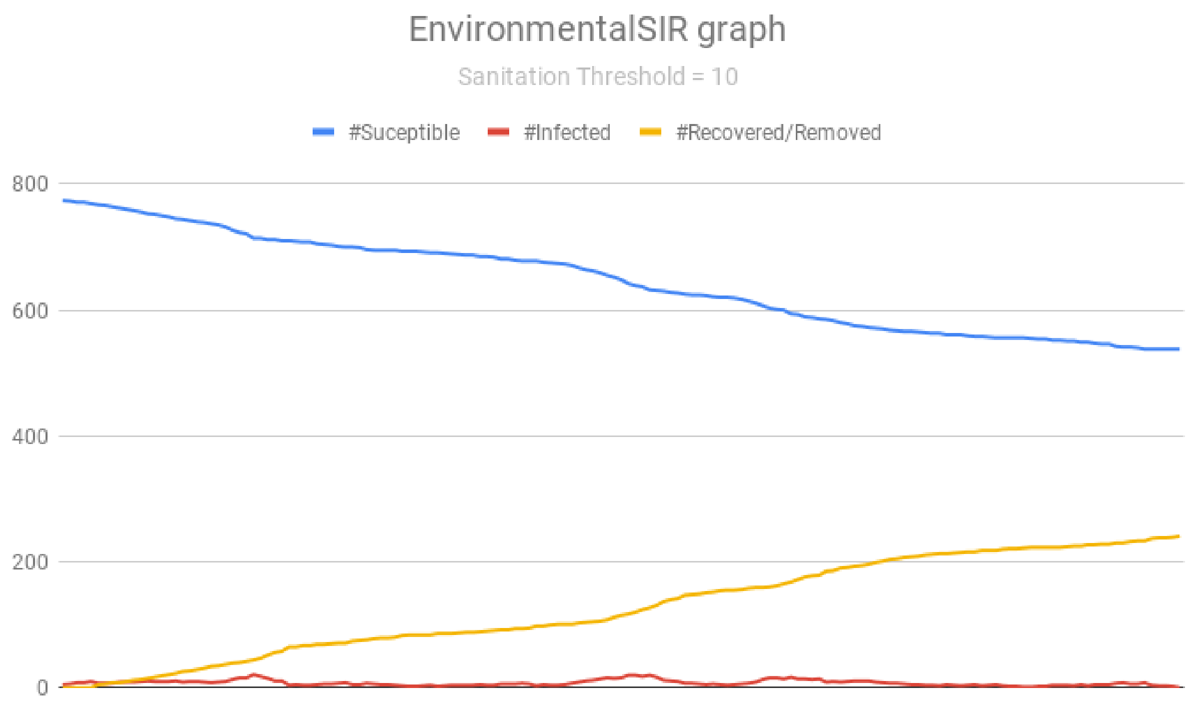

Figure 4.3. Secondary Outbreak When Sanitation Threshold = 10

Based on the above results, we conclude that changing the sanitation frequency primarily effects the duration of an epidemic. Decreasing the sanitation frequency results in the possibility of multiple outbreaks, while keeping the sanitation frequency short effectively prevents the occurrence of multiple outbreaks. Including environment nodes at all seems to make infection curves non-symmetric as well.

Going back to our series of runs with Sanitation Threshold = 5, we instead change the number of Environment nodes to 50. This results in an average epidemic size of 11.9%, indicating that increasing the number of environment nodes actually appears to reduce the overall magnitude of the epidemic. There is a good chance however that this is because increasing the number of Environment nodes effectively reduces the population density. That is, if a node is initialized to be an Environment node then it is done so in lieu of being initialized as a Susceptible individual. Furthermore, Environment nodes are incapable of directly generating agents, but can remove them. Thus, it makes sense that the introduction of Environment nodes (as opposed to an only human contact network) decreases the size of the overall epidemic.

CHAPTER 5
CONCLUSION

We began with an introduction of disease spread and common models, highlighting current gaps in research along the way. We then focused our discussion to network based models. From there, we highlighted the need for some metric by which to compare common network models as well as a need for a common base framework for network based models. This led to the proposal of NetworkSIR, which we found to adequately capture SIR behavior in a network. We also gave a strong argument for a uniform method of modeling infection in a network, which NetworkSIR uses.

We emphasized the extensibility of NetworkSIR by adapting it to EnvironmentalSIR. EnvironmentalSIR has the advantage of being able to capture the extent to which commonly visited environments contribute to the spread of disease. Our evaluation of it highlighted the following trends:

1. The introduction of Environment nodes allowed for non-symmetric infection curves

2. The introduction of Environment nodes allowed for multiple outbreaks, particularly as sanitation became more infrequent

3. Introducing more Environment nodes reduced the final average size of epidemics, perhaps by reducing the population density (as Environment nodes replaced individuals who would otherwise be initialized as Susceptible)

However, there is considerable future work that must be done to truly get an understanding of just how Environment nodes effect epidemic size. Tracing agent paths, for one, is an area ripe for exploration, and may provide insights to the field of contact tracing. Furthermore, the capturing of contact networks where commonly visited locations are recorded along with who visits them would be ideal to evaluate just how locations contribute to the spread of disease. The unavailability of such contact networks prevents us from verifying EnvironmentalSIR further, and similarly limits further analysis and verification of the above trends. It is our hope that by providing NetworkSIR and EnvironmentalSIR for public use we can inspire future researchers to perform work in this area.

APPENDIX

CODE REPOSITORIES

1. The source code for Network and EnvironmentalSIR is available at:

   https://github.com/madisonPickering/EnvironmentalSIR

2. The code used to generate statistics over multiple runs is available at:

   https://github.com/madisonPickering/EnvironmentalSIR _probabilityHelper

3. The code used to parse the CPIs from [19] is available at:

   https://github.com/madisonPickering/EnvSIR_HiResDatahelper

4. The contact networks used, specified as adjacency lists is available at:

   https://github.com/madisonPickering/EnvironmentalSIR_Datasets

REFERENCES

[1] Susser, E. and M. Bresnahan. ”Origins of Epidemiology”. *Annals of the New York Academy of Sciences*, Vol 954: pp 6-18. (2001).

[2] Grassly, N. and Fraser, C. ”Mathematical models of infectious disease transmission”. *Nature Reviews Microbiology*, 6: pp 477–487. (2008).

[3] Kermack, W. O., McKendrick A. G., and Walker, T. ”A contribution to the mathematical theory of epidemics”,*Royal Society London*: Vol 115, issue 772. pp 700–721 (1997; originally published 1927)

[4] Fine, P., Eames, K., and Heymann, D. ”“Herd immunity”: A rough guide”. *Clinical Infectious Diseases*: Vol. 52, No. 7, pp. 911-916 (2011)

[5] Giakkoupis, G., Mallmann-Trenn, F., and Saribekyan, H. ”How to Spread a Rumor: Call Your Neighbors or Take a Walk?”*ACM Symposium on Principles of Distributed Computing (PODC ’19)*, (2019)

[6] Lynch, N. Distributed Algorithms, Morgan Kaufmann Publishers Inc. (1996)

[7] Fischer, M., Lynch, N., Paterson, M. ”Impossibility of Distributed Consensus with One Faulty Process.” *Journal of the Association for Computing Machinery*, Vol. 32, No. 2, pp. 374-382. (1985)

[8] Dolev, S. Self Stabilization. MIT University Press. (2000)

[9] White, S. H., Rey, A. M., and Sánchez, G.R. ”Modeling epidemics using cellular automata” *Applied Mathematics and Computation*: Vol 186, Issue 1, pp 193-202, (2007)

[10] Robert, M., and Alun, L. ”Infection dynamics on scale-free networks”. *Physical review* E: Vol 64, 066112. (2002)

[11] Bansal S, Grenfell BT, Meyers LA. ”When individual behaviour matters: homogeneous and network models in epidemiology.” *J R Soc Interface* 4(16):879-91. (2007)

[12] Kasulis, K. ”‘Patient 31’ and South Korea’s sudden spike in coronavirus cases.” *Al Jazeera.* https://www.aljazeera.com/news /2020/03/31-south-korea-sudden-spike-coronavirus-cases- 200303065953841.html (2020)

[13] Chierichetti, F., Giakkoupis, G., Lattanzi, S., and Panconesi, A. Rumor Spreading and Conductance. Journal of the ACM: Vol. 65, No.4, Article 17. (2018)

[14] Dong, W., Pentland, A., and Heller, K. ”Graph-Coupled HMMs for Modeling the Spread of Infection”. *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence* (2012)

[15] Zachary, W. "An Information Flow Model for Conflict and Fission in Small Groups." *Journal of Anthropological Research*, Vol.33(4), pp.452-473. (1977)

[16] Godman, H. "How long does the flu last?" Harvard Health Publishing. (2020)

[17] World Health Organization (Europe), "Influenza: Data and Statistics" World Health Organization (Europe), https://www.euro.who.int/en/health-topics/ communicable-diseases/influenza/data-and-statistics(2020)

[18] Clayville, L. "Influenza update: a review of currently available vaccines." *P & T : a peer-reviewed journal for formulary management* vol. 36(10), pp 659-84. (2011)

[19] Salathe, M., Kazandjieva, M., Lee, J. W., Levis, P., Feldman, M. W., and Jones, J. H. "A high-resolution human contact network for infectious disease transmission." *Proceedings of the National Academy of Sciences of the United States of America*, 107(51), 22020–22025. (2010).

[20] Kazandjieva M, et al. "Experiences in measuring a human contact network for epidemiology research." *HotEmNets '10: Proceedings of the ACM Workshop on Hot topics in Embedded Networked Sensors, (Association for Computing Machinery, Killarney, Ireland)*, (2010)

[21] Huang, C., Liu, X., Sun, S. et al. "Insights into the transmission of respiratory infectious diseases through empirical human contact networks." *Scientific Reports*, vol.6, no. 31484. (2016)

[22] Leo, Y. S. et al. "Severe acute respiratory syndrome — Singapore, 2003". *Morb. Mortal. Wkly Rep.* 52, 405–411 (2003)

[23] Eubank, S., Guclu, H., Anil Kumar, V. et al. "Modelling disease outbreaks in realistic urban social networks." *Nature*, vol. 429, pp.180–184 (2004).

[24] Stehle J, Voirin N, Barrat A, Cattuto C, Isella L, et al. "High-Resolution Measurements of Face-to-Face Contact Patterns in a Primary School" *PLOS ONE*, 6(8): e23176, (2011)

[25] Xue, Ling et al. "A data-driven network model for the emerging COVID-19 epidemics in Wuhan, Toronto and Italy." *Mathematical biosciences*, vol. 326, (2020)

[26] Vasickova, P., Pavlik, I., Verani, M., and Carducci, A. "Issues Concerning Survival of Viruses on Surfaces." *Food and Environmental Virology* vol. 2(1), pp.24–34, (2010)

[27] Xie, X., Li, Y., Chwang, A., Ho, P., and Setol, W. "How far droplets can move in indoor environments–revisiting the Wells evaporation-falling curve." *Indoor air* vol. 17(3), pp.211-25 (2007)

[28] Montgomery, G., and Runger, C. Applied statistics and probability for engineers—3rd ed: Wiley, (2002)

[29] Lacey, M. “The Binomial Distribution” Yale Statistics. http://www.stat.yale.edu/Courses/1997-98/101/binom.htm (2020)

[30] Davis., K. “Table of Chi-square Statistics” University of Texas. https://web.ma.utexas.edu/users/davis/375/popecol/tables/chisq.html (2020)

[31] Pearson, Karl. ”On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling”. *Philosophical Magazine.* Series 5. 50 (302): 157–175. (1900).

BIOGRAPHICAL SKETCH

Madison Pickering is a second year master's student with a focus in Distributed Systems. She began attending The University of Texas at Dallas for her Bachelor's degree in Computer Science in 2017 and graduated Cum Laude in 2020. Her bachelor's degree was generously supported by the university through the Academic Excellence Scholarship. She formally began her master's degree in the summer of 2020, where her thesis research was similarly supported by the university through the Master's Summer Research Fellowship.

Madison first became exposed to research as a Fast Track student, where she was invited by Dr. Venkatesan to work on a distributed micropayment system. After the project's completion, she began work on a joint project with a bioengineering intern at NASA, as well as her capstone project which focused on resilient smart parking systems. Sadly, both of the projects were interrupted by COVID-19, driving her research focus towards relief efforts. Her current research interests include distributed systems, particularly where they relate to the creation of novel systems aimed to better humanity or the environment.

CURRICULUM VITAE

Madison Pickering

Contact Information:

The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Cell: (214) 551-7864
Email: `map@utdallas.edu`

Educational History:

M.S., Computer Science, University of Texas at Dallas
(2020-Expected Graduation 2021)

B.S., Computer Science, University of Texas at Dallas
(2017-2020) Graduation Cum Laude

Experience:

University of Texas at Dallas, Distributed Systems Lab — Undergraduate Research Assistant: Worked on the formation of a secure, distributed NFC-based payment system at all levels - background research, implementation, and paper writing. Gained an understanding of common security attacks and methods to prevent them.

Self Employed, University of Texas at Dallas — Calculus Tutor: Taught and tutored Derivative Calculus by solving problems with students, addressing fundamental deficiencies and answering questions along the way. Gained an ability to explain purely quantitative concepts in qualitative ways through common-sense examples and proofs.

Professional Recognitions and Honors:

Masters Summer Research Fellowship for exceptional research performed by thesis track Masters students working closely with one or more faculty members.
Academic Excellence Scholarship a four year scholarship awarded to outstanding students
UTD CS^2 Honors Program Invitation: an undergraduate Computer Science honors program
Johns Hopkins University International Talent Search Award for High Honors in mathematical, verbal, or spatial ability. Highly selective; the top 1% of applicants were awarded.

Papers:

NetworkSIR and EnvironmentalSIR - Two Network-Based Models for Examining Indirect Disease Spread: First author, currently in the process of submission
Tracing malicious transactions in a disconnected micro-payment system: second author, currently in the process of resubmission