A Modified SEIR Model with Weighted Graphs and Markovian Optimization for Describing the Spread and Treatment of Ebola Virus Disease

Team # 42010

February 9, 2015

1 Summary

The Ebola virus disease (EVD) has recently claimed thousands of victims in West Africa and elsewhere. While there is currently no cure, it is an interesting problem to imagine how best to use one to eradicate EVD.

We seek to develop a model to simulate the spread and possible treatment of EVD in the West African countries hit hardest by the recent outbreak (Guinea, Liberia, and Seirra Leone). We want a model that:

- **Accounts for geography**. For example, if all land traffic goes through a particular city, that city must act like a choking point for the spread of EVD. Also, it should take some time for the disease to spread far from its origin.
- **Allows for different treatment strategies**. Do cities spread treatments they possess, or hoard them? Should treatment distribution focus on curing sick people, or vaccinating healthy people? Should treatments be sent many small shipments, or just a few large ones?
- **Identifies a specific treatment strategy** as optimal.

We have modeled the countries under consideration as weighted graphs with major cities as their vertices, and with the weight assigned to each edge the travel distance between these cities. Within each vertex, we use a modified compartmental model (a variant of the SEIR model) to describe the spread of the disease in that population. The disease spreads both within and between cities, and different treatment strategies are explored.

Our model predicts that treatment should be delivered rapidly in small quantities, and that cities with high infected and high susceptible populations (along with nearby cities) should be targeted.

Contents

2 Letter for World Medical Association Announcement

The 2014 outbreak of Ebola in West Africa has lead to a devastatingly large loss of life, with around 9000 confirmed deaths so far. With several possible cures and vaccines in production, it has become important to understand how best to disperse treatments to avoid further deaths, and to finally put a stop to the epidemic.

Our team has created a mathematical model to simulate the spread and treatment of Ebola, and we have determined that it is best to rapidly send small quantities of treatment to cities with high infected populations and high susceptible populations. This prevents both the spread of the disease, and from people dying because it took too long to send medicine.

We urge humanitarian organizations to action, because our model predicts that there will be some few thousand more deaths by the end of this year if no treatment is provided and no other drastic measures are taken to prevent the further spread of Ebola. If treatment is not provided in the way we described, we predict that there will be up to at least a few hundred needless deaths.

Cordially,

Team # 42010

3 Restatement and Clarification of Problem

We propose a model to describe the spread and treatment of the Ebola virus disease (EVD). Since the epidemic that began in 2014 continues to devastate Guinea, Liberia, and Sierra Leone [4], we restrict ourselves to considering only these areas, which will inform our assumptions about sanitation conditions, the means of disease transmission, the time it takes to travel between cities, the likelihood of travel between cities, and the health infrastructure available to combat the epidemic and provide care for sick and recovering individuals.

Almost all known EVD outbreaks have lasted only a few months, and have taken place in relative geographic isolation; assuming that it persists, the longest and deadliest outbreak will be the current one the World Health Organization (WHO) and others are dealing with in West Africa (primarily in the aforementioned three countries), which has gone on for just over a year as of the writing of this paper. For this reason, we will restrict ourselves to simulating the spread and treatment of EVD in fixed geographic regions (that is, we will assume that our system is closed, and that there is no travel to and from other parts of the world), and for periods of not longer than a few years.

We will consider factors such as:

- how EVD spreads
- how treatment is administered
- strategies for dispersing treatment to needy areas
- how quickly the vaccine or drug necessary for treatment can be produced
- how much medicine is required to fully address an outbreak

4 Assumptions

4.1 Assumptions About the Character and Spread of EVD

- EVD is spread primarily through contact with body fluids of infected individuals; in particular, it is spread through direct contact with blood, feces, or vomit.
- The incubation period of EVD is between four to ten days.
- Individuals in the incubation period cannot infect susceptible individuals.
- Death from EVD occurs between six to sixteen days after showing symptoms.
- Since there are no known cases of remission, an individual that has recovered from EVD is no longer susceptible to infection.

- Infected individuals who show symptoms do not travel. This is intended to reflect both possible quarantine policies, and the overwhelming likelihood that very ill individuals do not travel. Hence, EVD is spread primarily via the travel of individuals for which the disease is still in the incubation period (so that they do not show symptoms).
- EVD moves from one place to another *exclusively* by the movement of individuals still in the incubation period. We are neglecting the possibilities that an animal (like a bat) or a dead body that was moved spread EVD to a new location.

4.2 Assumptions About Geography

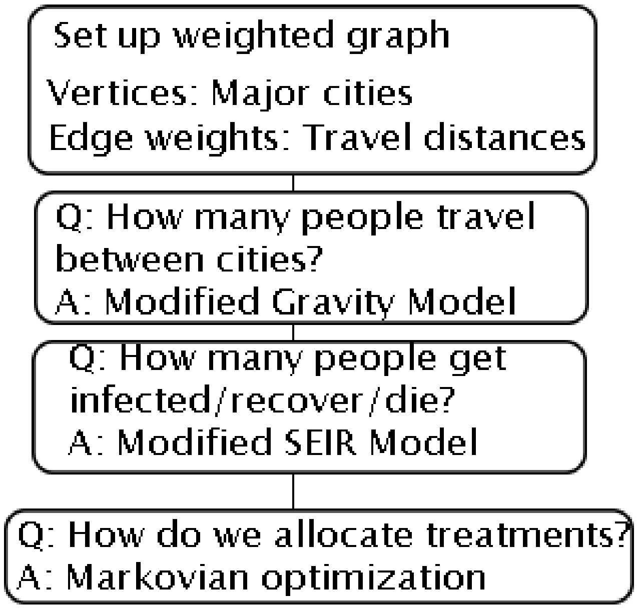
- Individuals only travel between cities that are 'near' each other in a sense that we will later make precise; in other words, travel between two geographically remote cities must go through intermediate cities.
- Travel between cities occurs only along well-traveled roads, so that the driving distance between cities is an accurate measure of the distance between them.
- Travel between cities takes at least one day, and occurs during the day. This is intended to reflect that most travel in West Africa happens in the day due to dangerous conditions at night, and that the primary means of transportation is by taxi.
- There is no air travel. For West African countries like Guinea, there is no infrastructure for air travel except in the capital, so this is a true statement.

4.3 Assumptions About Treatment

- For simplicity, the medication we are using to eradicate EVD both cures infected individuals, and vaccinates susceptible individuals. Equivalently, we can suppose that quantities of vaccinations and cures come in pairs, so that we do not have to consider the two factors separately.
- Infected individuals given the medication are cured immediately, and susceptible individuals that are vaccinated are immediately and permanently immune.
- Each region can only handle administering some number of treatments per day.
- Each treatment requires the same quantity of medication, and that quantity always costs the same amount to produce.
- Enough money is being spent on eradicating EVD so that running out of money and not being able to manufacture new treatments is not a problem.
- It takes a fixed amount of time to produce a fixed number treatments.

5 The Model

5.1 At A Glance


Set up weighted graph
Vertices: Major cities
Edge weights: Travel distances
Q: How many people travel between cities?
A: Modified Gravity Model
Q: How many people get infected/recover/die?
A: Modified SEIR Model
Q: How do we allocate treatments?
A: Markovian optimization

We want to describe the spread and treatment of EVD in a fixed geographic region—specifically, the West African countries hit hardest by the recent outbreak, which are Guinea, Liberia, and Sierra Leone.

To take *geography* into account, we represent each country as a weighted graph, where the vertices correspond to major cities (which we have arbitrarily pegged to be cities with 20,000 people or more as of currently available data), and where edge weights correspond to the travel distance between cities in kilometers as measured by Google Maps.

People move between cities according to a *Modified Gravity Migration Model*, which we will discuss later. Essentially, people are more likely to move between cities that are close to one another (in the sense of least travel distance), and bigger cities have more people moving in and out.

Within vertices, we use a compartmentalized model—a *Modified SEIR Model*, which we will discuss later—to describe how many people get infected, recover from infection, or die each day.

To take into account *different treatment strategies*, we try many possible strategies (involving where to allocate treatment, and how much treatment to allocate to each city), and find the *optimal result* using a *Modified Monte Carlo Markov Chain model.*

To take the *evolution of the system over time* into account, we run the model at many *time steps*, each representing a single day. At the beginning of each time step, we run the model again on the results of the previous time step. We run the simulation for a few years at most, since the longest outbreak has only lasted a year so far.

5.2 Setting Up the Weighted Graph

We wish to set up a weighted graph to represent each country under consideration. As previously described (mainly so that we have enough population data to do calculations), we make our vertices *major cities*, which we define to be cities with populations of 20,000 or more people. For example, in the case of Sierra Leone, there are *seven* major cities:

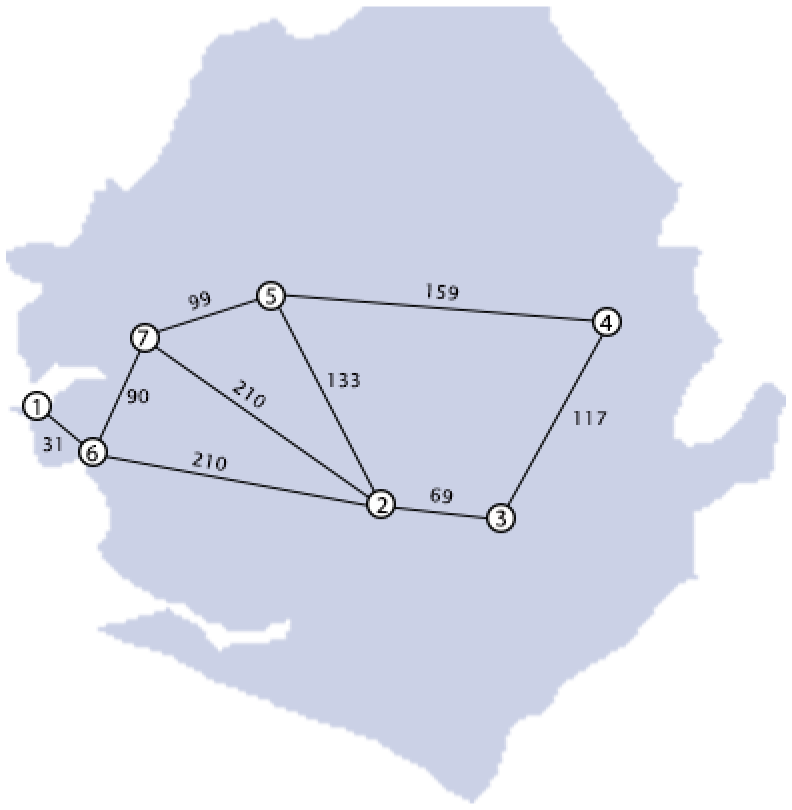
1. Freetown
2. Bo
3. Kenema
4. Koidu
5. Makeni
6. Waterloo
7. Port Loko

Each of these is a vertex on the graph representing Sierra Leone.

We connect vertices based on major roads. We have not taken into account travel by other means, like by train or by air, given that this infrastructure is either not present or not commonly used in the West African countries we are interested in.

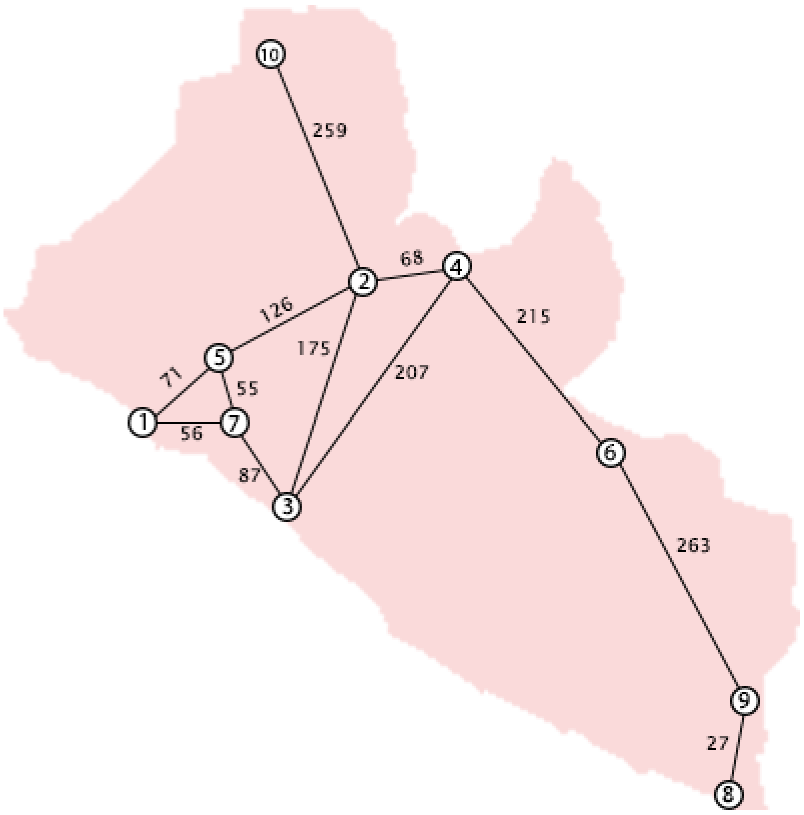
We want a notion of the distance d_{AB} between directly connected subregions A and B, so that we can gauge the likelihood of individuals moving from one subregion to another. Intuitively, travel between geographically remote subregions is unlikely. Since travel occurs on roads and not necessarily along a geodesic (i.e., the shortest path), we make the distance d_{AB} between A and B the distance along the major road (in kilometers, to three significant digits) connecting A and B, which we obtain from Google Maps. As we should hope, this notion of distance yields only positive distances, and is symmetric ($d_{AB} = d_{BA}$).

The demographic information and graphs of Guinea [1], Liberia [3], and Sierra Leone [2] are pictured below.



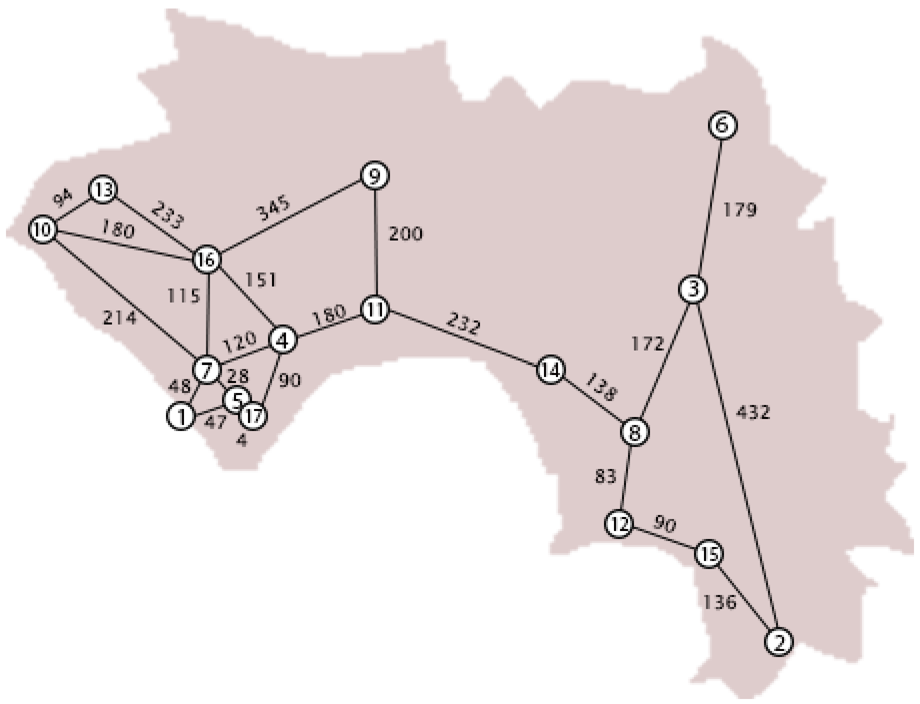
#	City	Population cf. 2004	Degree
1	Freetown	772,873	1
2	Bo	149,957	4
3	Kenema	128,402	2
4	Koidu	82,899	2
5	Makeni	80,840	3
6	Waterloo	34,079	3
7	Port Loko	21,961	3

Table 1: Sierra Leone demographics and connections.



#	City	Population cf. 2008	Degree
1	Monrovia	1,021,762	2
2	Gbarnga	56,986	4
3	Buchanan	50,245	3
4	Ganta	42,077	3
5	Kakata	34,608	3
6	Zwedru	25,349	2
7	Harbel	25,309	3
8	Harper	23,517	1
9	Pleebo	23,464	2
10	Foya	20,569	1

Table 2: Liberia demographics and connections.



#	City	Population cf. 2014	Degree
1	Conakry	1,667,864	2
2	Nzérékoré	194,178	2
3	Kankan	193,830	3
4	Kindia	135,000	4
5	Manéah	130,000	3
6	Siguiri	95,000	1
7	Dubréka	90,000	5
8	Kissidougou	90,000	3
9	Labé	90,000	2
10	Kamsar	85,000	3
11	Mamou	65,000	3
12	Guéckédou	63,000	2
13	Boké	61,449	2
14	Faranah	60,000	2
15	Macenta	58,000	2
16	Fria	55,000	5
17	Coyah	49,821	2

Table 3: Guinea demographics and connections.

5.3 Modeling EVD Spread at the Vertex Level

[SCHEMATIC SHOWING WHAT HAPPENS AT VERTEX LEVEL AT EACH TIME STEP]

Most of the actual modeling will happen at the vertex level. We want to represent the spread and treatment of the disease within our particular city; to do this we divide the population up into distinct categories:

- The set S of *susceptible individuals*. These individuals can contract EVD. They may freely travel to other vertices.
- The set E of *exposed individuals*. These individuals have contracted EVD, but are in the incubation period (fixed at N days), and so do not yet show symptoms. They cannot infect susceptible individuals. They may freely travel to other vertices.
- The set I of *infected individuals*. These individuals have contracted EVD, are past the incubation period, and show symptoms. They can infect susceptible individuals. They do not travel to other vertices.
- The set R of *recovered individuals*. These individuals have either contracted EVD but recovered, contracted EVD but have been treated, or were susceptible individuals that received vaccinations. They are immune to EVD. They may freely travel to other vertices.

Additionally, we may consider the set D of individuals that have succumbed to EVD.

At the beginning of each time step, migration in and out of each vertex occurs. Migration is modeled using a *Modified Gravity Migration Model*, which will be discussed later. As previously mentioned, travel only occurs between directly connected vertices, and only susceptible, exposed, and recovered individuals may travel. At the end of each time step, some number of

1. susceptible individuals become exposed ($S \to E$).
2. exposed individuals become infected ($E \to I$).
3. infected individuals die ($I \to D$).
4. infected individuals recover ($I \to R$).

The specific changes happen according to the differential equations of our modified *SIR Model*, which will be discussed later. Schematically, we can represent these relationships (ignoring the vaccination of susceptible individuals, which we will discuss later) using

$$\begin{tikzcd} S \arrow[r] & E \arrow[r] & I \arrow[r] \arrow[d] & R \\ & & D & . \end{tikzcd}$$

5.4 Modeling EVD Treatment and Prevention

[SCHEMATIC]

We want to model the treatment of EVD, which involves both the movement and allocation of available medication.

We assume that medication is delivered to some fixed location in shipments of size N every T time steps. At each time step, we decide if we want to use medication in certain subregions, or if we want to send it elsewhere. If we send some of it elsewhere, we must also decide how much to send.

Characteristics to consider in treatment strategies include:

1. the location to start distributing medicine.
2. the importance of addressing areas with a large infected population.
3. amount of medicine N produced in each shipment. The more time T spent, the more medicine we can produce.
4. the decision to keep medicine in a subregion, or to send it elsewhere.

At the beginning of each time step, medicine at a given vertex is *used* in some quantity, and *moved* in some quantity. We recall that we are supposing that each vertex only has the infrastructure for treating N cases per time step.

When medicine is used,

- susceptible individuals become immune ($I \to R$).
- exposed individuals recover ($E \to R$).
- infected individuals recover ($I \to R$).

We are assuming that deliveries follow the shortest path.

5.5 Model Parameters

5.5.1 Universal Parameters

- $\alpha \in [0, 1]$, the death parameter, is the probability that an infected individual will die in the current time step.
- $\beta \in [0, 1]$, the infection parameter, is the probability that a susceptible individual will become infected in the current time step.
- $\gamma \in [0, 1]$, the recovery parameter, is the probability that an infected individual will recover in the current time step.

5.5.2 Per Node

- $N(t)$, the *total population* as a function of time.
- $S(t)$, the *number of susceptible individuals* as a function of time.
- $E(t)$, the *number of exposed individuals* as a function of time.
- $I(t)$, the *number of infected individuals* as a function of time.
- $R(t)$, the *number of recovered individuals* as a function of time.
- $D(t)$, the *number of individuals* that have died from EVD as a function of time.

5.5.3 Per Node, Per Edge

- d_n, the *distance* to neighboring node n measured in kilometers
- $M_{I,n}$, the *number of people migrating in* from neighboring node n in the current time step
- $M_{O,n}$, the *number of people migrating out* to neighboring node n in the current time step

5.6 The Modified SIR Model

A modified *SIR Model* (where S stands for susceptible, I stands for infected, and R stands for recovered) is what we are using at the vertex level per time step to describe the spread of EVD. Given the usual SIR Model, the evolution of our system from one time step to the next would be given by the solutions to

$$\frac{dS}{dt} = -\beta\frac{I}{N}S$$
$$\frac{dI}{dt} = \beta\frac{I}{N}S - \alpha I$$
$$\frac{dR}{dt} = \gamma I,$$

where α is the death parameter, β is the infection parameter, and γ is the recovery parameter, each of which measures the likelihood of their namesake.

The first equation says that the change in S per time step is directly proportional to S, and to the percentage of infected individuals (with β as the proportionality constant). In other words, the more people there are, the more likely people are to become infected; moreover, the more sick people there are, the more likely people are to become infected.

Both principles agree with everyday experience.

The second equation says that the change in I per time step is the number of people that move from susceptible to infected, minus the number of people that move from infected to recovered.

The third equation says that the change in R per time step is directly proportional to I (with γ being the proportionality constant). In other words, some percentage of infected people will recover.

This model assumes that the birth rate equals the death rate, and that we are dealing with a closed system (so that there is no migration). However, given our model of interacting vertices, we need to account for the movement of individuals between neighboring vertices.

This model also assumes that individuals contract the disease and are infectious instantaneously. In this case (and for most real diseases), there is an incubation period, where individuals that have contracted the disease do not yet show symptoms, and are not yet infectious. This becomes particularly important when we consider that we are preventing infected individuals from traveling; this means that exposed individuals become the effective carriers of the disease to new areas.

Hence, we modify our model firstly to be an SEIR model (where the extra letter E stands for exposed), and secondly to account for travel between vertices. The modified equations governing the changes in S, E, I, and R per vertex V, per time step are

$$\frac{dS}{dt} = -\beta\frac{I}{N}S + \sum_{k\in\mathrm{Nb}(V)}\left[\frac{S_k}{N_k - I_k}M_{I,k} - \frac{S}{N-I}M_{O,k}\right] \tag{1}$$

$$\frac{dE}{dt} = \beta\frac{I}{N}S - E_I + \sum_{k\in\mathrm{Nb}(V)}\left[\frac{E_k}{N_k - I_k}M_{I,k} - \frac{E}{N-I}M_{O,k}\right] \tag{2}$$

$$\frac{dI}{dt} = -\gamma I - \alpha I + E_I \tag{3}$$

$$\frac{dR}{dt} = \gamma I + \sum_{k\in\mathrm{Nb}(V)}\left[\frac{R_k}{N_k - I_k}M_{I,k} - \frac{R}{N-I}M_{O,k}\right], \tag{4}$$

where $\mathrm{Nb}(V)$ is the set of vertices connected to V, $M_{I,k}$ is the number of individuals that have traveled in from a neighboring vertex k at the beginning of the time step, $M_{O,k}$ is the number of individuals that have traveled out to a neighboring vertex k at the beginning of the time step, and E_I is the number of individuals that have moved from being exposed to being infected at the beginning of the time step.

The way we interpret each equation is almost the same as in the simpler SIR Model, but we have added migration terms; we assume for simplicity that the demographics of people migrating to a new place are the same as the demographics in V, and that the demographics of people migrating in from neighboring vertices are the same as the demographics of people in those vertices. In other words, if half of the people in a neighboring vertex are susceptible, then half of the people migrating in from that vertex are susceptible too.

5.7 The Modified Gravity Migration Model

The *Gravity Migration Model* says that we can use a formula analogous to the equation that gives the gravitational force between two massive objects (in the sense of Newtonian gravity) to model the likelihood that people move between the two cities. In particular, if we are looking at cities with populations P and P' at a distance d away from each other, the strength of the relationship R between the two is given by

$$R = K\frac{PP'}{d^2},$$

where K is some proportionality constant.

We want an equation that gives the number of people that travel from subregion A to subregion B, and from subregion B to subregion A in one time step. But this equation cannot be it (or at least is not a good model as-is) because it is symmetric in A and B; this would mean that the populations of every subregion would remain constant, and that people would only be exchanged.

It is a good place to start, because the inverse dependence on d^2 means that people are more likely to travel to cities that are close by, but we must incorporate some asymmetry somewhere.

We will assume that people have a sense (whether through news outlets or word of mouth) of if neighboring subregions have *more*, or *less* infected individuals. If a neighboring subregion has *more* infected individuals, people are less likely to move there, and more likely to leave there. We account for this sort of behavior by multiplying by the ratio of infected individuals, so that we obtain that the number of people M_{AB} that migrate from A to B is given by

$$M_{AB} = K\frac{I_A}{I_B}\frac{p_A p_B}{d_{AB}^2},$$

where K is again just a proportionality constant.

Of course, in the beginning, and in many others cases, one of I_A or I_B will be zero. To avoid this, we add in a random fudge factor, and make M_{AB} simply that in that case.

5.8 Treatment Strategies and Optimization

We ran our model through a Monte Carlo Markov Chain with Modified Criteria for the Movement Condition (MCMCMCMC) that we wrote for this problem to explore the phase space of treatment strategies to systematically find the solution that minimizes the number of deaths in a way that avoids getting stuck at local minimas.

Whereas a traditional Monte Carlo Markov Chain (MCMC) returns a bayesian posterior distribution that represents the probability distribution for each parameter in a model fit, our spin on it returns a similar set of histograms for each parameter. A tighter peak around a minimum reveals a stronger dependence of the effectiveness of the treatment on that parameter, and a wider peak represents a less critical parameter. This gives us an easy way to quantitatively compare the importance of each parameter to the treatment solution by comparing the standard deviations of their posteriors.

Our MCMCMCMC run had $2n + 4$ free parameters where n is the number of cities in the graph. For both vaccines and medicines, we fit how to weight each city's worthiness of receiving treatment. This will account for factors that are hard to quantify such as geographical location, frequency traveled through, and ability to transmit the disease and keep it alive.

We also fit the number of days to save up the vaccines and medicine before releasing them to the public. We suspected that it may be more effective to try to hit the disease hard all at once rather than slowly as the medicine comes out.

The last set of parameters that we fit was the amount of medicine and vaccines for a city to save after the disease is eradicated in case the disease comes back from travelers from other cities.

We ran our model through a large number of iterations with MCMCMCMC to get a large posterior distribution.

6 Implementing the Model

For the programming, we used the C++11 language because of it's fast speed, easy to use data structures and libraries. We developed all the code on Mac OS X using Xcode.

6.1 Simulation

6.1.1 Data Structures

In this model we focus on some African countries, thus we use use the data structures `country`, `city`, and the parameters of the simulation `params` to describe each of the African countries we used in our simulations.

- `country` contains
  - A set of cities `cities`
  - The amount of medicine in the country `medicineAmount` is a nonnegative integer.
  - The amount of vaccines in the country `vaccineAmount` is a nonnegative integer.
  - A rate of production of both medicine and vaccines `rateOfProduction` is a nonnegative integer.
  - The cycle count for the country (in days) `cycles` is a nonnegative integer.
- `city` (for city k) contains
  - Susceptible population size `S` is a nonnegative integer.
  - Infected population size `I` is a nonnegative integer.
  - A queue `E` where element `E[n]` (which is a nonnegative integer) is the population of susceptible individuals after n days.
  - Recovery population size `R` is a nonnegative integer.
  - Death population size `D` is a nonnegative integer.
  - Total population size `N` at a given cycle is a nonnegative integer.
  - Initial total population size `N0` is a nonnegative integer.
  - Vaccinated population size `V` is a nonnegative integer.
  - A boolean flag to indicate if the city starts out with infected individuals `beginInfected`
  - A boolean flag to indicate if the city is a port city `isPort`. Port cities are the first ones in the country to get treatments.
  - A list of neighbors `C` where `C[n]` the nth neighbor of C.
  - A function that determines the edge weight to the neighboring city `dist()` return a positive integer.
- `params` (which is to be used by the Markov Chain) contains
  - The weight of the medicine `m[]`.
  - The weight of the vaccines `v[]`.
  - The amount of cycles to wait until disbursing medicine to port cities `m_wait`.
  - The amount of cycles to wait until disbursing vaccines to port cities `v_wait`.
  - The amount of medicine to save `m_save`. This will be used when a city's infected population is 0 and has unused medicine.
  - The amount of vaccines to save `v_save`. This will be used when a city's infected and susceptible population are both 0 and has unused vaccines.

6.1.2 Data Types

In the simulation, we used unsigned long integers for the city population sizes to avoid integer overflow. To keep track of the cities we used their memory addresses to keep everything consistent.

6.1.3 Simulation Algorithm

The following code describes what happens in the simulation. The `vec params` is the vector of parameters that the Markov Chain modifies. The function simulation takes those parameters and uses them for weights in specific areas of the program that we will see.

In the simulation function, a country is created, a loop is ran and then the total number of deaths is calculated.

```
int CYCLES = 365 /*run for 1 year*/
simulation(vec params):
    country p
    for(i in range(0,CYCLES) ) {
        p.updateCycle()
        p.moveBetweenCities()
        p.moveWithinCities()
        p.administerTreatment()
        p.moveTreatment()
        p.produceTreatment()
        p.receiveTreatment()
    }
    int D = 0;
    for(city c in p.cities) ){
        D = D + c.D
    }
    return D
```

6.1.4 Country Creation

When a country is created, we construct it using one of four input files. Three of the input files are countries and one of them is a all countries put together. The contructor function for countries first reads in all of the data from the selected input file. As it reads in the city names from the file it adds them to the list of cities `cities`.

After reading in the file, the function creates two 2-D arrays and runs the Floyd Warshall Shortest Path with Path Reconstruction. One array `floydDist` keeps track of the shortest path distances and `next` keeps track of the shortest paths. After the Floyd Warshall algorithm finishes we use the next 2-D array to give each city a function that delivers the neighbor to travel to get medicine to its destination the fastest.

6.1.5 Looping through Cycles

- `p.updateCycle()`
  - This essentially increments `CYCLE`: `CYCLE = CYCLE + 1`, so that a day has passed.
- `p.moveBetweenCities()` updates each city's state using the differential equations (note: P' is the population size of P where P can be S, E, or R of a neighboring city.
  - $M_O = K\frac{(N-I)(N'-I')}{d^2}(\frac{I}{I'})$ is the outgoing migration scalar, and K is the migration constant.
  - $S \leftarrow S - \frac{S}{N-I}M_O$
  - $E \leftarrow E - \frac{E}{N-I}M_O$
  - We assume the infected are too ill to move or are quarentined so the infected population is not changed here.
  - $R \leftarrow R - \frac{R}{N-I}M_O$
  - $N \leftarrow N - \frac{R}{N-I}M_O - \frac{E}{N-I}M_O - \frac{S}{N-I}M_O$
- `p.moveWithinCities()`
  - $S \leftarrow S - \beta\frac{I}{S}N$
  - $E \leftarrow E + \beta\frac{I}{S}N$ this enqueues individuals to the incubation queue so that they are now exposed and can become infected after the incubation period.
  - `front` $\leftarrow$ `dequeue(E)`
  - $D \leftarrow D + \alpha I$
  - $R \leftarrow R + \gamma I$
  - $I \leftarrow I +$ `front` $-I(\alpha + \gamma)$
  - $N \leftarrow N + \alpha I$
- `p.administerTreatment()` If the city has any treatment it will administer medicine to the infected and vaccines to the susceptible and exposed immediately. Since the exposed may already be infected but are unaware we get rid of some of the vaccines. If the city is cleansed of infected then it will select a neighboring city to give treatments. An amount of medicine and vaccines may saved. The 2 save parameters handle add weight to the choice of how much to give, and the Markov Chain adjusts the weights.
- `p.moveTreatment()`
  - Medicine has to travel from city to city to get to its destination. In this function we move vaccines and medicine to neighboring cities so that it gets closer to its destination. Each city is aware of which neighboring city to deliver medicine so

  that the medicine reaches its destination the fastest. If medicine has arrived to the city that cycle it will be saved in the city for one cycle and then moved out of the city.

- `p.produceTreatment()`
  - Adds a fixed amount of medicine to its `medicineAmount` and a fixed amount of vaccines to `vaccineAmount` so that it can be sent out.
- `p.receiveTreatment()`
  - If the country has any treatments to disburse it does so here. It sends treatments to selected cities, but the treatments are first moved to the port cities. When selecting which cities to send treatments to it selects cities based on the magnitude of $\omega\frac{IS}{N}$ where ω is the weight of medicine or vaccines of the city set in the parameters initially or by the Marcov Chain.

6.2 Markov Chain Monte Carlo

```
Markov(vec p, int iter, int numCities):
    dold = simulation(p)
    for(i in range(0,iter)
        p' = p
        // Add step to weights
        for(k in range(0, numCities))
            p.v[k] = p.v[k] + normalRand()
            p.m[k] =p. m[k] + normalRand()
        p[m_wait] =  p[m_wait] + normalRand()
        p[v_wait] =  p[v_wait] + normalRand()
        p[m_save] =  p[m_save] + normalRand()
        p[v_save] =  p[v_save] + normalRand()

        // Check new weights
        dnew = simulation(p)
        quality = dold / dnew
        randNum = uniformRand()
        if(quality <= randNum OR 1 <= quality)
            p = p'
            dold = dnew
            accepted = accepted + 1
            // Save p to plot
```

The parameters for the Markov Chain are the set of parameters `p`, number of iterations `iter` and the number of cities `numCities`. It first runs the simulation so that there is something to compare the number of deaths in the loop (which is the return value of the simulation). The loop starts and `p'` is set to `p` and then modified when the weights are updated. The weights are shifted by a normally distributed random number. The simulation is ran again returning a value into `dnew`.

`quality` becomes a ratio of `dold` and `dnew`. randNum is given uniformally distributed random number from 0 to 1 , and it is used to create diversity in the collection of accepted parameters. The quality of the last simulation is checked; if quality is greater than or equal to 1 that means that there were less deaths with the new parameters. We check to see if quality is less than randNum just to add diversity to the collection. Finally we store the accepted `p`'s.

6.3 Runtime

Our implementation of the algorithm has the following runtime:

$$O(I_{Markov} N_{cycles} N_{cities}^2)$$

where I_{Markov} is the number of iterations of the Markov chain, N_{cycles} is the number of cycles the simulation is ran, and N_{cities} is the number of cities in the graph.

7 Results

We found that medicine should be sent out in quick, small shipments rather than infrequent large ones. We also found that dispersing the medicine more widely rather than hoarding it just in case was a better treatment strategy. Putting more vaccine weight on cities with high susceptible populations worked well, in combination with putting more medicine weight on cities with high infected populations.

Also, a good strategy is to not just focus on cities with high infected populations, but to vaccinate the susceptible populations of nearby cities so that the disease does not spread.

8 Strengths and Limitations

8.1 Strengths

- The model is a conceptually simple extension of preexisting compartmental models, and achieves reasonable agreement with real world outbreaks.
- The model can be generalized to handle infectious diseases *other than* EVD by tweaking parameters like how infectious the disease is and how deadly it is. Certain characteristics not apparently present in the case of EVD, like a chance of remission after recovery, are not hard to incorporate.

- Using our modified Markov model, we have a sense of *how confident we are* that a given treatment strategy is optimal.
- Simulations involving a reasonably large number of cities (34 in the West African case) run in a reasonable amount of time. The most computationally expensive aspect of our model is the optimization of treatment strategy via a modified Markov model, but other optimization strategies, like using genetic algorithms, could easily be implemented instead.

8.2 Limitations

- Certain prescient real world concerns, like the possibility that EVD spreads to another country by air, are not taken into account. For that reason, we can only use this model to study EVD in isolated regions.
- We are assuming, in the case of multiple countries that are connected to one another, that borders remain open throughout our simulation. During the recent West African outbreak, Guinea closed its borders to Sierra Leone to contain the spread of EVD; our model ignores changes like this.
- Weighted graphs were constructed and a time step length of one day was chosen due to features particular to the West African case; travel happens primarily by land, and dangerous conditions at night force most travel to occur during the day, limiting how far people can travel (and how far medicine can spread) in a short period of time. Our model, then, cannot immediately be generalized to locations like the United States, because we must account for the additional connections between cities, and the possibility of quick travel between remote areas.
- We have focused on treatment through cures and vaccines, but not at all on prevention by isolating infected patients. This is because many West African cities lack the health infrastructure for isolation wards; nonetheless, it still happens, and we are not taking it into account.
- Our model parameters are highly sensitive; changing initial conditions by small amounts causes drastically different end results, which is not realistic.

9 Conclusion and Future Work

We constructed a model for the spread and treatment of EVD in Guinea, Liberia, and Sierra Leone. According to our model, small but quick-moving shipments of treatments that target cities with high infected populations and the nearby cities was the optimal treatment strategy, often saving many thousands of people from dying.

Future work would focus on working to make the model less sensitive to perturbations to the initial conditions. Additionally, we could expand the model to more countries in West Africa, or more countries in general.

10 Appendix: Code

```
//
//  main.cpp
//  COMAP_EBOLA
//
//

#include <iostream>
#include <queue>
#include <climits>
#include <cstdio>
#include <deque>
#include <cmath>
#include <vector>
#include <fstream>
#include <random>

#include "country.h"

// Input files
#define INPUT_GUINEA        "guinea.txt"
#define INPUT_GUINEA_BIG    "guineaBIG.txt"
#define INPUT_LIBERIA       "liberia.txt"
#define INPUT_SIERRA_LEONE  "sierra_leone.txt"
#define INPUT_WEST_AFRICA   "west_africa.txt"

#define INPUT_FILE INPUT_WEST_AFRICA

// Output files
#define MARKOV_RESULTS "results.txt"

// Simulation Parameters
#define RATE_OF_PRODUCTION 2000
#define MARKOV_ITERATIONS 100

using namespace std;
```

```
typedef struct {
    double vStep;
    double mStep;
    int mWaitStep;
    int vWaitStep;
    int mSaveStep;
    int vSaveStep;
} vec6;


unsigned long int simulation(vec params, string fileName){

    /* Currentry is created with:
     * Given file name
     * Fixed rate of production
     * The Markov model parameters
     */
    country p(fileName, RATE_OF_PRODUCTION, params);

    // Country's cities go through cycles
    for (int i = 0; i < CYCLES; i++) {
        p.updateCycle();
        p.moveBetweenCities();
        p.moveWithinCities();
        p.administerTreatment();
        p.moveTreatment();
        p.produceTreatment();
        p.receiveTreatment();
    }

    // Number of deaths in simulation are calculated and returned
    unsigned long int deaths = 0;
    for(city *c : p.cities)
        deaths += c->D;

    return deaths;
}


vector<vec> MCMC(vec params, int iter, vec6 step, string fileName){
```

```
unsigned long int dold, dnew, accepted;
double quality, randNum;

// Used to adjust parameters and store results respectively
vec newparams;
vector<vec> histogram;


// Run simulation and store death count in dold
dold = simulation(params, fileName);

// Initialize accepted to 0 (it keeps track of how many times the Markov Chain takes
accepted = 0;

// Setup distributions and random number generator for varying step sizes
std::uniform_real_distribution<double> qualityDistribution(0.0,1.0);
std::normal_distribution<double> vWeightStepDistribution(0.0, step.vStep);
std::normal_distribution<double> mWeightStepDistribution(0.0, step.mStep);
std::normal_distribution<int> mWaitStepDistribution(0.0, step.mWaitStep);
std::normal_distribution<int> vWaitStepDistribution(0.0, step.vWaitStep);
std::normal_distribution<int> mSaveStepDistribution(0.0, step.mSaveStep);
std::normal_distribution<int> vSaveStepDistribution(0.0, step.vSaveStep);
std::default_random_engine generator;
generator.seed((unsigned int)time(0));


for (int i = 0; i < iter; i++) {

    // Make a copy of old parameters to alter with step
    newparams = params;

    // Add a real normal randomly generator step size to the medicine weights
    for(map<int, double>::iterator it = params.medicineWeights.begin(); it != params
        it->second += mWeightStepDistribution(generator);
    }

    // Add a real normal randomly generator step size to the vaccine weights
    for(map<int, double>::iterator it = params.vaccineWeights.begin(); it != params.
        it->second += vWeightStepDistribution(generator);
    }
```

```
        // Add a discrete normal randomly generator step size to each of the integer par
        params.medicineDaysToWait += mWaitStepDistribution(generator);
        params.vaccineDaysToWait += vWaitStepDistribution(generator);
        params.medicineSave += mSaveStepDistribution(generator);
        params.vaccineSave += vSaveStepDistribution(generator);



        // Perform simulation
        dnew = simulation(params, fileName);


        // Compare old death count to new death count
        quality = (double)dold/dnew;

        // Set randNum a real uniformly distributed number between 0 and 1
        randNum = qualityDistribution(generator);

        // Check quality:
        // keep newparams if death count is better than old keep newparams
        // OR if quality is less than the real uniformly distributed random variable ran
        if( quality <= randNum || 1.0 <= quality){

            if(1 <= quality ){

            }

            // Set old settings to the new ones and increment the acceptance count
            params = newparams;
            dold = dnew;
            accepted++;

            // Store new guesses in histogram
            histogram.push_back(params);
        }
    }

    // Show acceptance rate
    cout << "Acceptance rate: " << 100.0 * accepted / (double)iter << " %" << endl;


    return histogram;
```

```
}


int main(int argc, const char * argv[]) {

    // Start sequence of random numbers
    srand((unsigned int)time(0));

    // Initial parameters of estimation and step parameters for MCMC
    vec6 steps = (vec6){0.01,0.01,1,1,100,100};
    vec params;

    // Take in number of cities
    ifstream IN(INPUT_FILE);
    int numCities;
    IN >> numCities;
    IN.close();

    // Initial values assigned to estimation parameter
    for (int i = 0; i < numCities; i++) {
        params.vaccineWeights[i] = 1.0/((double)numCities);
        params.medicineWeights[i] = 1.0/((double)numCities);
    }

    params.vaccineDaysToWait = 4;
    params.medicineDaysToWait = 4;

    params.medicineSave = 0;
    params.vaccineSave = 0;

    // MCMC is ran and results retrieved
    vector<vec> results = MCMC(params, MARKOV_ITERATIONS, steps, INPUT_FILE);


    // File is created with contents of results
    ofstream OUT(MARKOV_RESULTS);

    // Output results to output.txt file
    for (int i = 0; i < results.size(); i++) {
        for (int k = 0; k < results[i].vaccineWeights.size(); k++) {
            OUT << results[i].vaccineWeights[k] << " ";
        }
        for (int k = 0; k < results[i].medicineWeights.size(); k++) {
            OUT << results[i].medicineWeights[k] << " ";
        }

        OUT << results[i].medicineDaysToWait << " ";
        OUT << results[i].vaccineDaysToWait << " ";
        OUT << results[i].medicineSave << " ";
        OUT << results[i].vaccineSave << " ";
        OUT << std::endl;
    }

    OUT.close();

    return 0;
}
```

References

[1] Institut National de la Statistique de Guine. *Guinea Cities.* February 2015. `http://www.citypopulation.de/Guinea-Cities.html`.

[2] Statistics Sierra Leone. *Sierra Leone Cities.* February 2015. `http://www.citypopulation.de/SierraLeone.html`.

[3] Liberia Institute of Statistics & Geo-Information Services. *Liberia Cities.* February 2015. `http://www.citypopulation.de/Liberia.html`.

[4] World Health Organization. *Ebola Virus Disease.* February 2015. `http://www.who.int/csr/disease/ebola/en/`.