Optimal Strategy Conducted by Complex Networks Theory

Team #41895

Contents

1 Introduction

As described in the World Health Organization (WHO) fact sheet, Ebola virus disease, formerly known as Ebola hemorrhagic, is a severe, often fatal illness in humans[1]. The current West Africa Ebola outbreak since March 2014 has been so intensive that it has had profound economic, humanitarian, political and security dimensions: at least 22,495 people were infected and 8,981 of whom were reported death by WHO (See in **Figure 1**)[2]. Guinea, Liberia and Sierra Leone have been the most severely affected countries, and the lives and livelihoods of people there have been severely destroyed. Thus, efficient cure and protection strategies are in badly need.

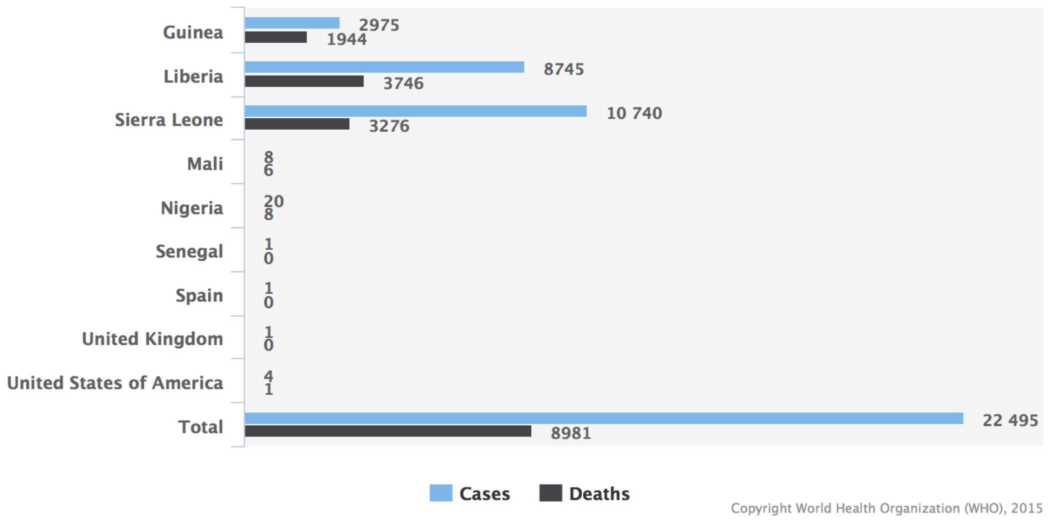


Figure 1 Cumulative reported Ebola cases (data up to 1 Feb. 2015)

Unlike other epidemic, Ebola is featured by extremely high fatality rate. The average fatality rate is around 50%, and it usually undergoes an incubation period from 2 to 21 days. The human-to-human transmission is via direct contact (through broken skin or mucous membranes) with the blood, secretions, organs or other bodily fluids of infected people, and with surfaces and materials (e.g. bedding, clothing) contaminated with these fluids.[1]

The above facts provide important information about Ebola, and are going to help us in the following modeling process.

1.1 Restatement of the Problem

According to the problem description, we were informed that the World Medical Association (WMA) had already found a productive cure, and we were asked to establish a mathematical model to assist WMA to optimize the eradication of Ebola. In general, we are going to help them to decide who, when, where and how to deliver the medicine. This model considered not only the spreading features of Ebola virus but also critical factors including geographical locations, population characteristics, and manufacturing of the drug, et al.

Due to the complexity of this problem, we resolved the modeling process into two phases as follow:

- Build a model to simulate the transmission of Ebola in different districts;
- Extend the model and enable it to reflect spatial information and feedback of different delivery strategies.

To be more specific, we are expected to discuss following issues in our paper:

- Considering the average household size and residential preferences of each districts, analyze probable social contact network structure;
- Speculate the state transitions among the suspected, the infected, the recovered and the dead against time variation;
- Discuss the roles certain social networks play in the transmission of Ebola virus, and simulate the infection chains on the networks;
- Compare the similarity between the simulating result and the real data, and make use of the analog tendency to predict the growing trend after the cure is put into use;
- Propose feasible indicators to determine the optimal positions of delivery based on previous networks;
- Establish a utilization function to depict the relationship between the quantity and effect of the medicine in each area;
- Compute the maximum utility under the constraint of certain production efficiency;
- Provide an overall estimation of the performance of this delivery system.

1.2 Previous Research Review

There has been plenty of study on the dynamics of epidemic spreading ever since mathematical approaches were used to analyze the properties of diseases. The most famous models among all the researches are SIS model and SIR model, which were proposed by Kermack and Mckendrick in the early 20th century[3]. These two models are simple yet robust, but they don't take social and spatial patterns into account.

For decades, scientists have tried various methods to extend the basic SIS and SIR model in order to enable them to fit population characteristics. For instance, the microscopic simulation model based on Cellular Automata has been widely used since 1990s[4]. However, this method enjoys rather low computational efficiency and lacks of the capability to represent complex relation.

Recently, the complex network model has been one of the hottest topics. Taking advantage of complex network, it's possible to represent each individual, and mimic their links and interactions with each other. In particular, Watts and Strogatz proposed small-world (WS) network model in 1998[5], while Barabási and Albert developed scale-free (BA) network model in 1999[6]. These two models considered small world property of real world and thus result in high similarity comparing to real network structure.

In our work, we preferred BA model to WS model since the BA model obeys power-law distribution, and considers growth property and preferential attachment property when the networks are growing[7]. The resulting networks apparently reveal heterogeneity on the degree of each node, thus the analog value and real value fit quite well.

2 Assumptions and Justifications

The central idea of our modeling is based on scale-free networks. In order to solve all the problems raised, we have following additional assumptions, and each rule here will be explained in details in following chapters.

- The first phase of epidemic outbreak happened naturally, namely without human interference;

- A recovered patient couldn't get infected a second time;
- People with closer relation tended to live closer;
- There's only one place to deliver the medicine, named treatment center, in each district;
- The time needed for each patient to get the cure is proportional to the distance between the patient and the treatment center;
- The treatment centers use FCFS strategy to serve people requesting the medicine;
- The total quantity of the drug is constrained under external factors, in another word, not infinite;
- A patient always recovers immediately once he/she gets the drug.

3 Notations

Table 1 Notations used in this paper

Notation	Description
Π_i	The probability of a new node i is connected to an existing node
k_i	The degree of node i
L	The average path length of a BA network
C	The clustering coefficient of a BA model network
$P(k)$	The probability of a node has degree k
P_I	The probability of a susceptible individual becomes infected
P_R	The probability of a infected individual recovers
P_D	The probability of a infected individual dies
X	The population size
X_0	The initial amount of infected person
$time\_family$	The average number of iterations that occurs within the family from a single infected individual

$time\_social$	The average number of iterations that occurs in the social from a single infected individual
$\boldsymbol{R}$	The eigenvector of the modified adjacent matrix
$PR(p_i)$	The i's element of the eigenvector
N	The total number of pages
d	Weight coefficient
$d(P_1, P_2)$	The distance between two individuals
x_i	The amount of medicine divided to district i
$g_i(x_i)$	The utility function of district i
$U(x_1, x_2, \cdots, x_N)$	The global utility function
$f_M(t_1, t_2, \cdots, t_M)$	The minimum of $U(x_1, x_2, \cdots, x_M)$
n	The length of time series.
l	The length of largest common subsequence with acceptable error range
Sim	similarity degree of two time series
max_people	total number of points in the randomly generated graph
max_family_size	maximum number of points within a family in the family network
init_infected_people	initial number of infected nodes in the randomly generated graph
non_intervention_time	the number of time intervals under natural conditions
intervention_time	the number of time intervals under medicine-available conditions
ratio_distance	the ratio of coordinates to the real distance
ratio_family_social	the ratio of the infection probability in family network to the infection probability in public network
pro_infect	the infection probability
pro_cure	the recovery probability under natural conditions
pro_dead	the death probability
medicine_cure	the recovery probability after taking medicines

4 Modeling for Epidemic Spreading

Social Contact Network is closely pertinent to the social relations and daily routine of individuals. As for Ebola, the route of transmission is dominant by direct physical contact. Under such circumstances, social networks actually serve as the basis of transmission. Before simulating the infection, we have to mimic a social network at first.

4.1 Scale-free (BA) Networks Model

Based on growth and preferential attachment property, Barabási, A. L. and Albert, R. (1999) provided an explicit construction rule as follow:

Growth rule:

- The growing process starts with an initial connected network of m_0 nodes.
- Only one new node is added to the network at a time.
- Each new node is connected to m ($\le m_0$) existing nodes.

Preferential attachment rule:

- The probability Π_i that a new node i is connected to an existing node j is proportional to the ratio of the degree k_i of node i and the total degree $\sum_j k_j$ of the existing node j. This relationship could be written in the following expression:

$$\Pi_i = \frac{k_i}{\sum_j k_j}$$

The resulting network has essential trait, called Matthew Effect, which has been widely discussed by considerable researchers. And this feature comes to an accord with common sense that people tend to forge relationships with those who already have large interpersonal relationship net.

The network accords following mathematical properties, which may be of help to analyze the population characteristics:

- **Average Path length**

The average path length of a typical BA model network is

$$L \propto \frac{logN}{loglogN}$$

It apparently denotes small world property.

- **Clustering coefficient**

The clustering coefficient of a typical BA model network is

$$C = \frac{m^2(m+1)^2}{4(m-1)}\left[ln(\frac{m+1}{m}) - \frac{1}{m+1}\right]\frac{[ln(t)]^2}{t}$$

This indicates that when the scale of network is large enough, the network won't show evident clustering features.

- **Degree distribution**

The degree distribution of a typical BA model network is scale-free, and could be described using the form of power law as follow:

$$P(k) \sim k^{-3}$$

Here is a random scale-free network generated by NetworkX library of python:

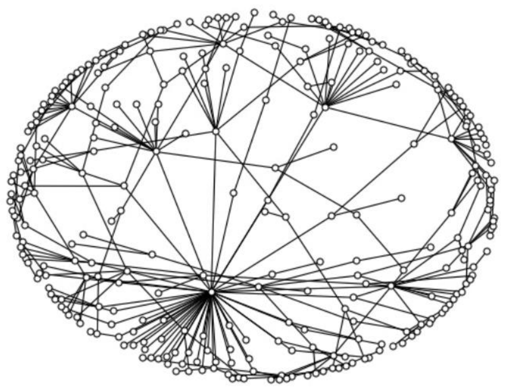


Figure 2 Basic scale-free network

4.2 Alternate Social Networks Model

4.2.1 Practical Basis

As illustrated above, Ebola won't transmit unless two people have rather close contact. In another word, we could assume that the probability of transmission is proportional to the frequency and extend of valid physical contact. This rule reminds us that the probability of transmission may vary in public social network and household network. It means that when a person is infected, his/her family members will risk mush higher rate to get influenced than any other acquaintance of this individual. This assumption is directly proved in the situation report of Liberia government (See **Figure 3**).

Hence, the household structure does play an important role of affecting infection chain. In particular, the West Africa usually involves relatively large family size. According to African Health Observatory of WHO and ICF Macro, the average household size of Liberia, Guinea and Sierra Leone is 5.1, 5.4 and 5.9 respectively. [9, 10] From the later modeling, you'd see the huge effect of this indicator.

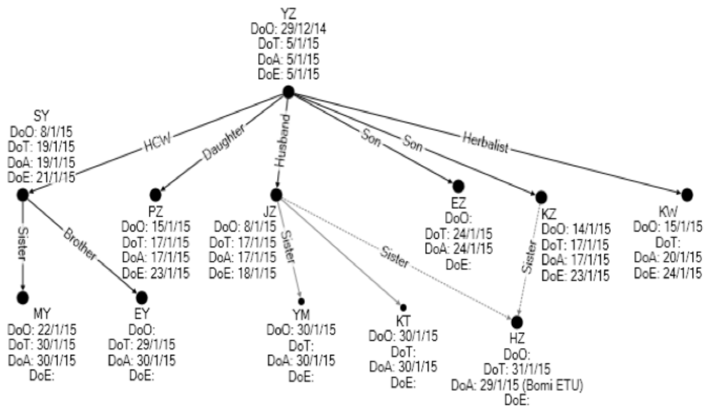


Figure 3 St. Paul bridge cluster infection chains[8]

4.2.2 Public Network & Family Network

Ni, S. (2008) provided a detailed explanation considering the social patterns stated above. Generally, we could partite the activity time of an individual into daytime and nighttime. In daytime, the virus is transmitted via public network; in nighttime, however, the virus spreads via each independent compact family network. **Figure 4** depicts the holistic schema.

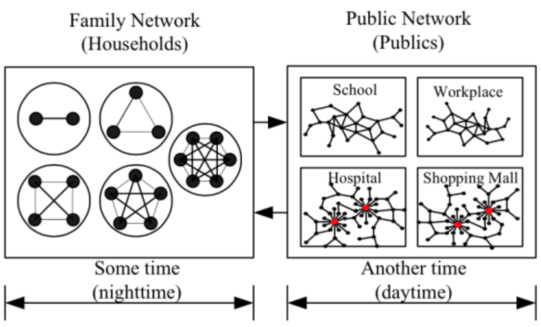


Figure 4 Schema of the alternate social networks[11]

The components of these two networks are the same group of residents. The only difference is the effective time. The two networks alternately dominate the transmission of Ebola virus.

4.2.3 Construction algorithm

- **Family Network**

We use small-world network (WS model) to simulate. This connectivity structure is simply constructed using complete graphs demanding a uniform distribution $u\{1, max\_family\_size\}$ of their sizes.

- **Public Network**

We use scale-free network (BA model) to simulate. Different from regular scale-free network, we modified the growth rule that one entire family network is added to the public network at a time. With preferential attachment rule remains valid, every new node builds its links with equation:

$$\Pi_i = \frac{k_i}{\sum_j k_j}$$

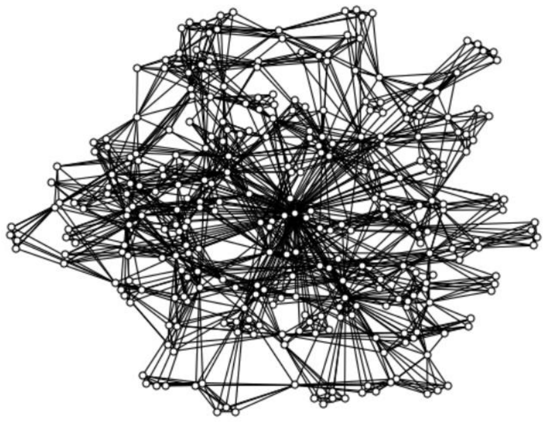


Figure 5 Scale-free network with household structure

Above is the alternate social network modified based on the scale-free networks. From the **Figure 5** we could see distinct clustering phenomenon of families.

Under this paradigm, there are two transmission rates, and thus two scale-dependent time interval coefficients. We define $time\_family$ as the average number of iterations that occurs within the family from a single infected individual[12]. Similarly, we define $time\_social$ for public network.

4.3 SIRD Model

Now we are ready to apply the classic SIR (Susceptible-Infected-Recovered) model on the network we've just built. This model is expressed by differential equation, and the solution to it is deterministic. Nevertheless, the real infection process obviously possesses randomness. In order to mimic the spreading of Ebola, we modify the state transition as follow:

$$S + I \overset{P_I}{\rightarrow} 2I$$

$$I \overset{P_R}{\rightarrow} R$$

The former equation shows that a susceptible individual suffer a risk P_I of becoming infected. The latter one denotes the process for recovery for patients.

Notice that the fatality rate of Ebola is rather high, so we add a supplementary state 'D', which stands for dead case. The dead state will be labeled 'deleted' from the graph and is unable to switch to any other state.

$$I \xrightarrow{P_D} D$$

When implementing by programs, we assumed that the population size is X and the initial amount of infected person is X_0. Following are detailed steps:

Step 1. Infected person contact with all his neighbors, carrying vital Ebola virus at a probability of P_I.

Step 2. All the infected turn into a state of recovered or dead, at a probability of P_R or P_D, respectively.

Step 3. Go back to Step 1. for every iteration.

5 Advanced Modeling for Delivery System

Our model in the last chapter successfully simulated the spreading of diseases among a certain group of people. Calculating the quantity of medicine will then be possible. However, it's incapable of denoting geographical information, and consequently cannot help us establish the delivery system. We have to extend this model and assign location to each node.

5.1 Introducing Location: Fruchterman-Reingold Algorithm

Recall that in the last chapter, we used the default layout of BA model to draw the graph. This approach lets us observe the structure clearly, but fails to represent the real distribution of human residence. In addition, we don't even have the coordinate of each node.

In order to improve the current representation, we used Fruchterman-Reingold algorithm[13] to simulate the position features of human residence. The basic idea of this function is force-directed replacement. Fruchterman, T. M. and Reingold, E. M. (1991) proposed two principles for drawing[12]:

- Vertices connected by an edge should be drawn near each other.
- Vertices should not be drawn too close to each other.

Treated a real district as a normalized area $[0,1] \times [0,1]$, everytime a node is placed in, we have to take into account the existing layout and available spaces. Thus we believed that this strategy shared something in common with the process of human settling down. When the nodes are randomly scattered in the normalized area, the family members tend to live closer yet still consider the capability of the given district.

Here are the specific implementation steps:

Step 1. For every node, calculate a vector that accumulates repulsive forces and attractive forces from all of other nodes.

Step 2. Move every node towards corresponding vector's direction.

Step 3. Go back to Step 1. for every iteration.

We conduct above algorithm on the alternate social network and regard it as simulating locations of residents. See **Figure 6** as an example.

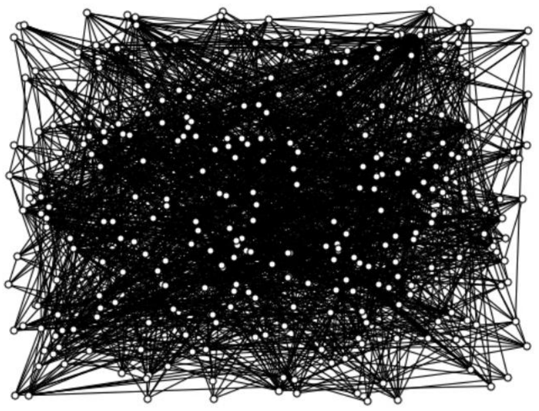


Figure 6 Random network generated by FR algorithm

5.2 Calculating Locations of Delivery: PageRank Algorithm

Based on the assumptions proposed in the previous chapter, each district could only have exactly one location of delivery. We call it the treatment center. The intuitive way to find the center is to find the most important node among the network, and this goal reminds us the famous PageRank algorithm.

PageRank is a method for computing a ranking for every web page based on the graph of the web[14]. In this algorithm, a page has high rank if the sum of the ranks of its backlinks is high. The rank indicates the importance of the page. To make an analogy with web graph and social network, PageRank asks each resident to vote for the most significant site. If there exists a link between two people, they vote for each other. The importance of one node is calculated by the total importance of every node that votes for it.

The real calculating process is denoted by the following mathematical equation:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

The PageRank values are therefore the eigenvector of the modified adjacent matrix:

$$\boldsymbol{R} = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ \vdots \\ PR(p_N) \end{bmatrix}$$

The answer is

$$\boldsymbol{R} = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \vdots \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} l(p_1, p_1) & l(p_1, p_2) & \cdots & l(p_1, p_N) \\ l(p_2, p_1) & \ddots & & \vdots \\ \vdots & & l(p_i, p_j) & \\ l(p_N, p_1) & \cdots & & l(p_N, p_N) \end{bmatrix} \boldsymbol{R}$$

where for each j, $\sum_{i=1}^{N} l(p_i, p_j) = 1$

We use this approach to determine the location of delivery. As seen in **Figure 7&8**, the yellow nodes are selected as the treatment centers.

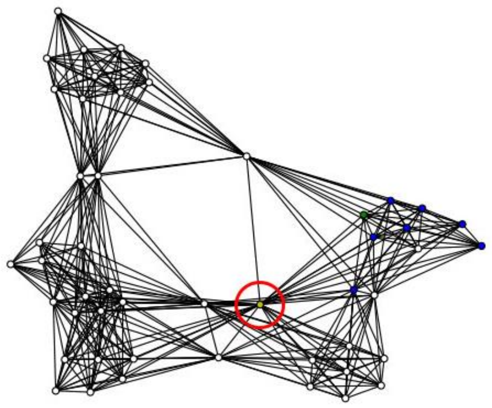
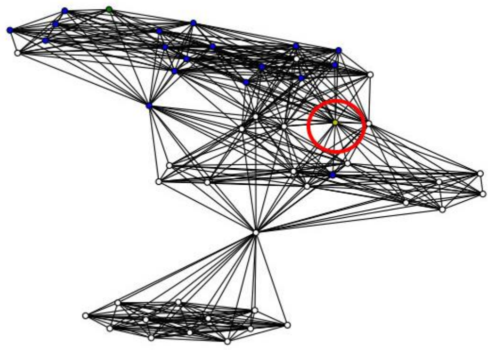



Figure 7,8 Resulting locations of delivery

5.3 Request Model Based on Weighted Networks

Using the location information computed by the FR algorithm, it's straightforward to obtain the distance between each two individuals.

$$d(P_1, P_2) = \sqrt[2]{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

We saved this value *d* as the weight of the corresponding edge in the graph.

Next, we chose the request model that commonly used in communication network. Assume that the medicine in each center is limited, and once a person gets infected, he/she starts to move towards the treatment center. The time he/she consumes is proportional to the distance between he/she and the center. Since the amount of patients may be rather large. The block at the center does happen frequently.

Under such circumstances, we build a waiting queue for all the patients requested for the cure. We followed the basic doctrine that all men are created equal. Consequently, we ignored any so-called priority and just use FCFS strategy to serve all the patients.

Figure 9 shows the requesting process. The blue node and green node just began to advance to the treatment center once they got infected. Due to the distance variance, they would arrive one after another, thus the patient represented by the blue node could get the cure earlier.

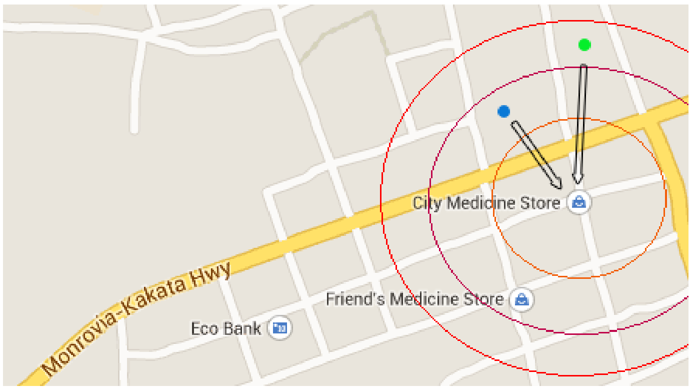


Figure 9 Request model

5.4 Modified SIRD Model: Additional Cure Rate

How to simulate the performance of our cure then? Since we assumed that the cure is so effective that the patient could get recovered immediately, we just made one subtle change to the patient's state transition: alter the probability P_R in formula

$$I \xrightarrow{P_R} R$$

to 1 when the patient is served by the treatment center.

Now we are able to combine all the factors that matter in our model, and simulating the Ebola spreading on the network. The result (**See in Figure 10**) reveals the affect of these elements.

- The spreading speed between family and public network is different;
- The transmission process shows endemicity;
- The residents staying closer to the treatment center are more likely to get the cure.

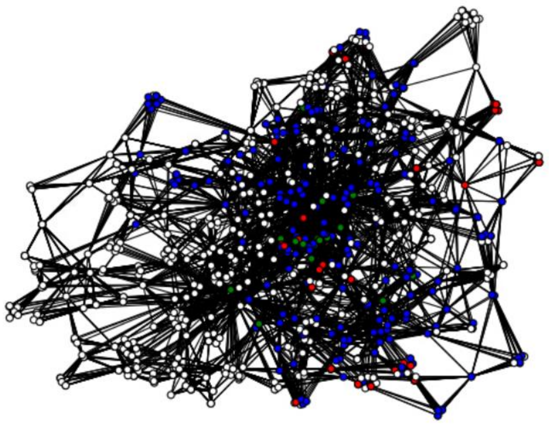


Figure 10 Spreading result (red-I, green-R, blue-D, white-S)

6 Dynamic Optimization Based on Our Model

Arriving at this point, a new question has raised: How to estimate the utility of the medicine? We simplified the utility as the predicted death toll: the fewer, the better.

6.1 Establishing the Utility Function

According to the model we constructed above, we can estimate the death toll in a short period of time under a given drug supply. **Figure 11** shows the result in Montserrado, Liberia, which expresses the function between two variables in a simulated environment.

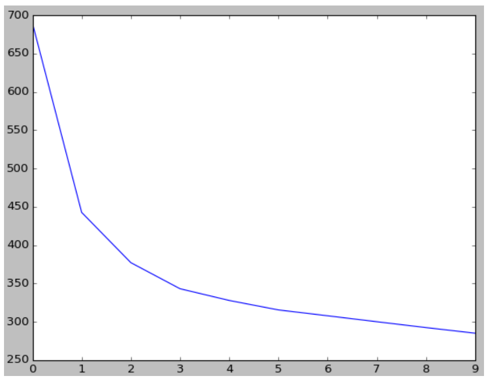


Figure 11 Death toll in Montserrado, Liberia

For each district, we computed the corresponding expectation of death toll by repeating 20 times. Due to a general consensus that all men are created equal, we build the utility function as

$$U(x_1, x_2, \cdots, x_N) = \sum_{i=1}^{N} g_i(x_i)$$

subject to the constrains $\sum_{i=1}^{N} x_i \leq c$ where $x_i \geq 0, i = 1,2, \dots, N$

To be clear, *i* denotes a particular area; *N* stands for the amount of areas; *c* limits the drug supply per-interval.

6.2 Solver: Determine Optimal Delivery Strategy

Before we determine the optimal strategy, we acknowledged that the whole modeling process was based on simulation environment. What's more, the cost of transportation is not a principal factor because the volume of the drug is rather small and discussing the cost is meaningless when humans' lives could have been threatened. Then the problem came down to finding the minimized death toll under a given quantity of medicine. It's a typical dynamic programing problem.

Based on the dynamic programming method[15], we minimized the utility function. For $c \geq 0$, define the function

$$f_M(t_1, t_2, \cdots, t_M) = \min U(x_1, x_2, \cdots, x_M)$$

For $0 \leq M < N$,

$$f_{M+1}(t_1, t_2, \cdots, t_M, t_{M+1}) = \min_{0 \leq p \leq c} [f_M(t_1, t_2, \cdots, t_M) + g_{M+1}(p)]$$

We can obtain an optimal solution by iterative process.

7 Testing the Model: Result & Analysis

7.1 Initial Spreading Simulating Patterns

We used the SIRD model on Alternate Social network to simulate the spreading of Ebola in West Africa countries. The following line graphs are the corresponding results of Conakry, Guinea, Montserrado, Liberia and Freetown, Sierra Leone. Epidemic situation in these cities are the most severe among all the districts. Blue dots denote the real data of infected people while red ones represent simulation results. Time interval is one week.

In these graphs, time series had been smoothed. They indicated trends in the numbers of infected people instead of exactly describing the process.

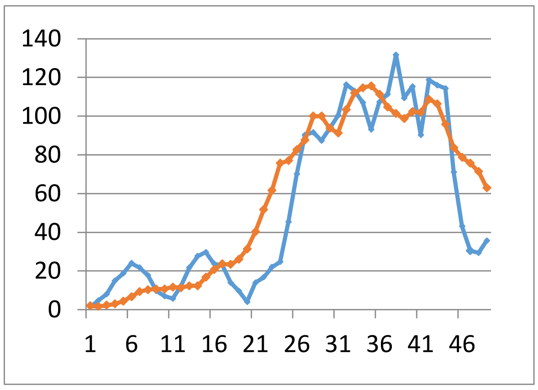
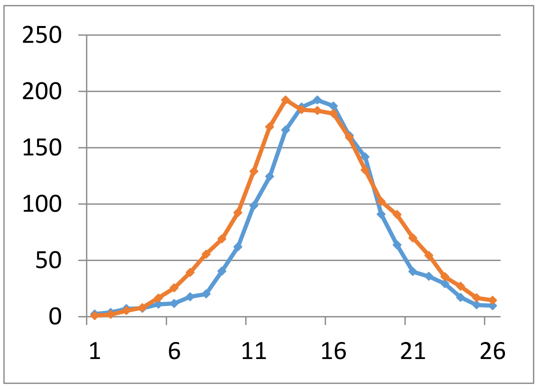


Figure 12 Conakry



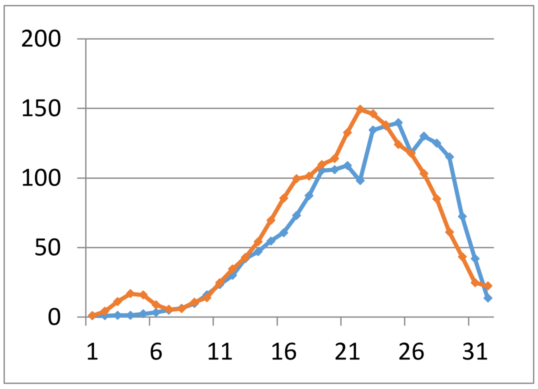


Figure 13 Montserrado **Figure 14** Freetown

We obtained time series of both simulation numbers and actual numbers of infected people. Observing the curves, we found that the tendency of simulation results is similar to that of real data. Next, we used LCSS (Largest Common Subsequence) [16] measure to analyze the similarity of two time series of each city. After smoothing the time series with three-step average, we translated the time series for no more than four time units, and roughly estimated the lower bound of similarity degree using $Sim = l/n$, where l is the length of largest common subsequence with acceptable error range of 20%, n is the length of time series. Here are testing results:

- Conakry, Guinea; $n = 49$,$l \geq 34$, $Sim > 0.69$
- Monrovia, Liberia; $n = 26$, $l \geq 21$, $Sim > 0.80$
- Freetown, Sierra Leone; $n = 26$, $l \geq 19$, $Sim > 0.71$

For each city, the similarity degree of time series is high enough, so we assured that our basic spreading model is effective.

7.2 Predicted tendency after delivering the cure

The three countries (Guinea, Liberia, Sierra Leone) were divided into a series of districts. For every district, we simulated the numbers of infected people and then compared those to the data retrieved from WHO[17].

During the testing process, we discovered that if the total infected cases were less than 300, simulation results and real time series showed apparent difference. It seemed reasonable as the scale was so small that it may fail to show consistent patterns. So we leaved out the data of those districts and only use rough average value to replace it.

According to the criterion above, we chose 2 districts of Guinea, 3 districts of Liberia, 9 districts of Sierra Leone, and evaluated suitable parameters. We adjust the data by three-step average, and observed that simulation time series are all similar to real time series to some extent. **Figure 15~28** display the plots of analog value and real data. The dot representation is the same as what described in the last paragraph.

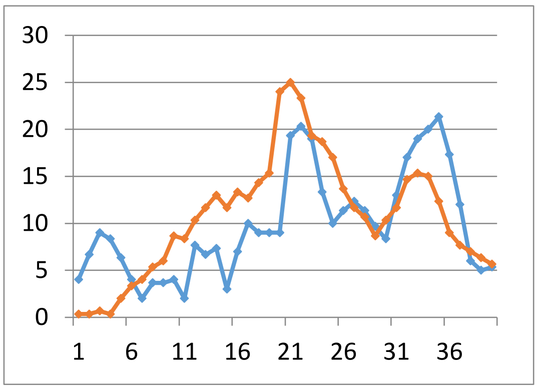


Figure 15 Conakry

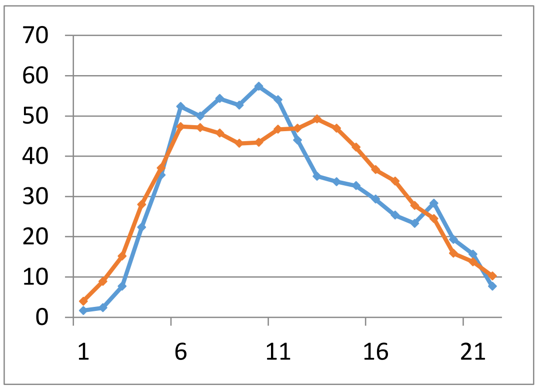


Figure 16 Macenta

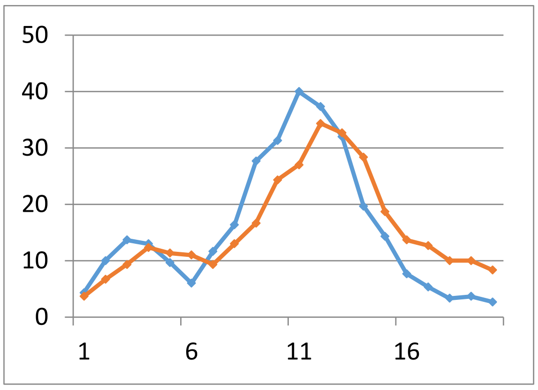


Figure 17 Lofa

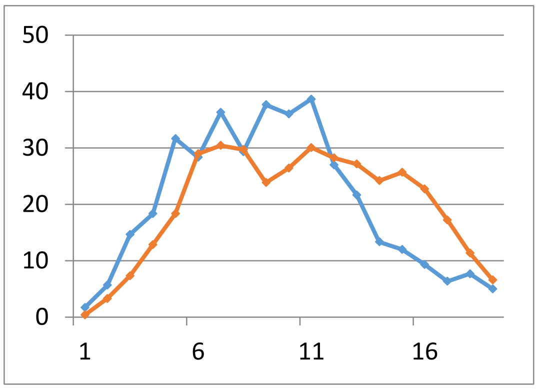


Figure 18 Margibi

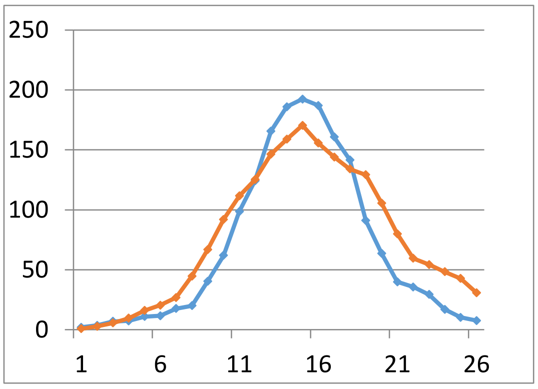


Figure 19 Montserrado

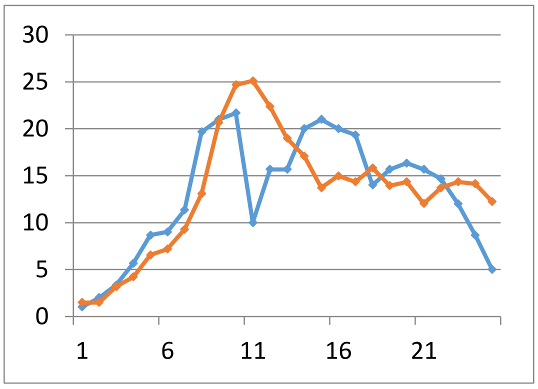


Figure 20 Bo

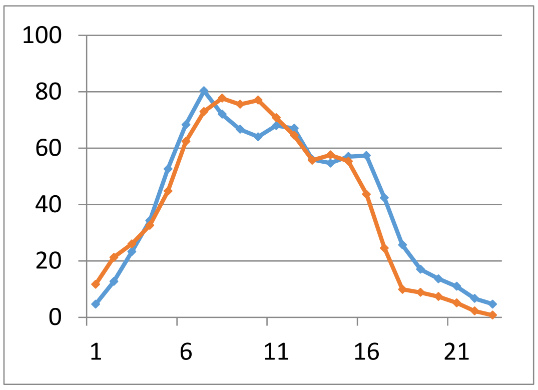


Figure 21 Bombali

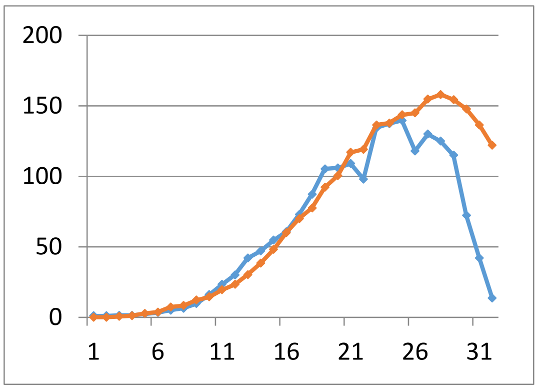


Figure 22 Freetown

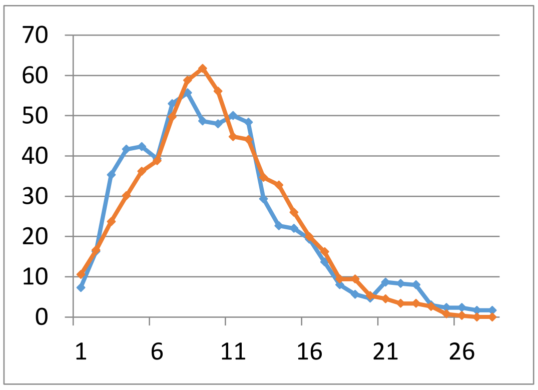


Figure 23 Kailahun

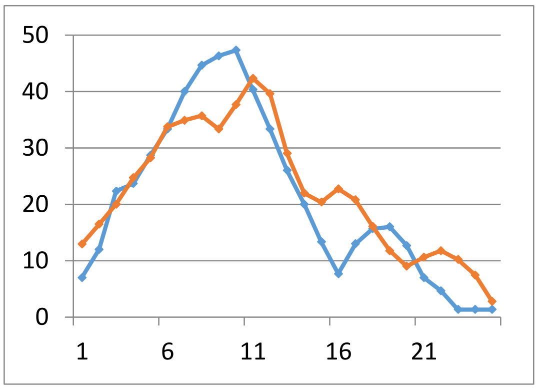


Figure 24 Kenema

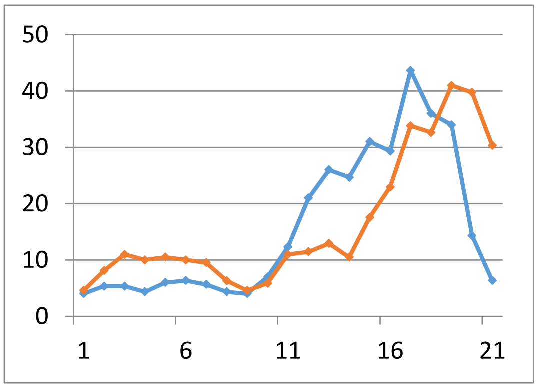


Figure 25 Kono

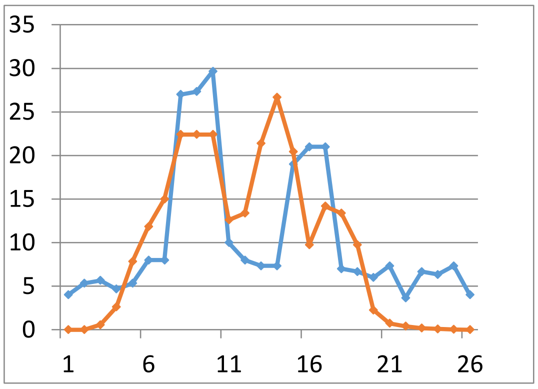


Figure 26 Moyamba

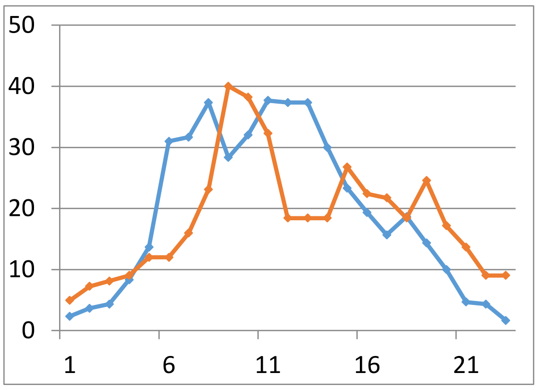


Figure 27 Tonkolili

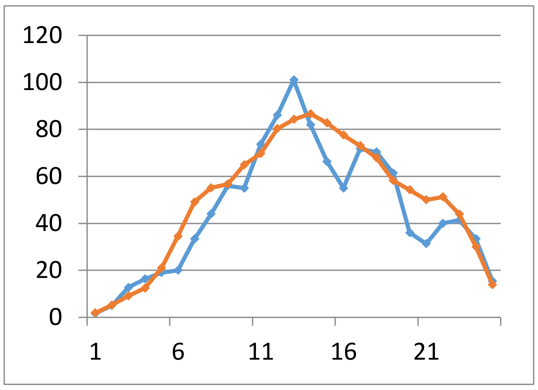


Figure 28 Western Rural

After adding geographical factors, the advanced model somehow showed a bit more deviation on accuracy. However, taking randomness into consideration, it could be accepted. On the random graphs, the optimal positions of treatment center were given by Pagerank algorithm. In the real world, we could assume that the location of treatment center approximately lie in the eigenvector centrality.

We used this model to simulate medicine delivery results. In each district, different amount of medicine would lead to different death toll. Except two districts (Moyamba and Kailahun) in which Ebola had halted, others' simulation result generated utility functions. Since the independent variable is defined on a discrete set, we made a table to illustrate the function:

Table 2 Predicted death toll against quantity of medicine for each district

District \ Medicine	0	1	2	3	4	5	6	7	8	9
Bo	84.52	71.43	66.71	63.75	61.3	58.03	56.20	52.55	49.59	47.51
Bombali	306.60	188.53	158.04	145.33	135.74	126.15	116.10	110.67	104.66	99.58
Conakry	56.27	36.34	28.83	24.29	21.23	18.36	16.87	14.18	14.00	11.96
Freetown	374.85	337.25	334.32	330.24	324.07	319.14	321.84	318.60	319.45	319.84
Kenema	16.57	8.56	4.72	1.54	1.87	1.32	1.76	1.98	1.76	2.19
Kono	154.22	101.98	85.82	72.41	68.78	60.51	56.25	51.37	48.61	45.10
Lofa	39.20	24.20	18.40	14.40	9.10	8.60	8.00	7.30	4.60	5.20
Macenta	208.86	167.78	159.76	156.39	150.94	147.23	145.62	139.82	137.47	133.09
Margibi	73.32	55.57	44.09	37.14	30.68	27.40	25.57	21.42	21.03	19.97
Montserrado	689.58	442.74	377.16	343.11	327.82	315.46	307.81	300.05	292.41	285.07

Tonkolili	103.49	93.42	63.90	49.55	49.89	35.19	33.68	31.14	27.90	25.35
Western Rural	216.24	159.04	140.25	127.70	121.46	116.14	111.40	106.58	101.26	98.27

7.3 Dynamic programming result

For each given predicted death toll, we performed dynamic programming process to calculate the optimal distribution and the least requirement of medicine. We fitted the curve of requirement and number of death:

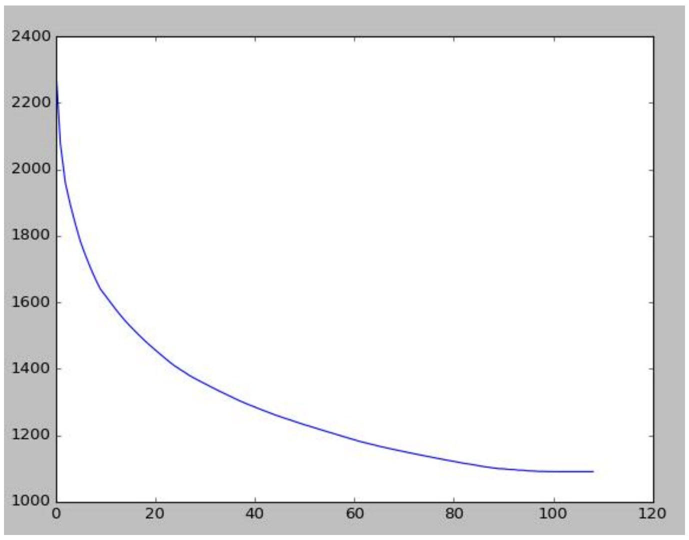


Figure 29 Total death toll against total quantity of medicine per interval

The horizontal axis shows the requirement of medicine and the vertical axis shows the expected death number.

8 Sensitivity Evaluation

The simulation analytical method proposed is a stochastic model, which implies that the following parameters are not suitable for sensitivity evaluation:

- *max_people, max_family_size, init_infected_people*
- *non_intervention_time, intervention_time*
- *ratio_distance, ratio_family_social*

The first line contains 3 parameters that the graph is directly determined by. Those in the following 2 lines depend on the authentic data. All of them will be set to constant.

There are four essential parameters in our model. To test each parameter, tests were run on a standard environment. Our specific settings can be found in the appendix.

8.1 Evaluation for *pro_infect*

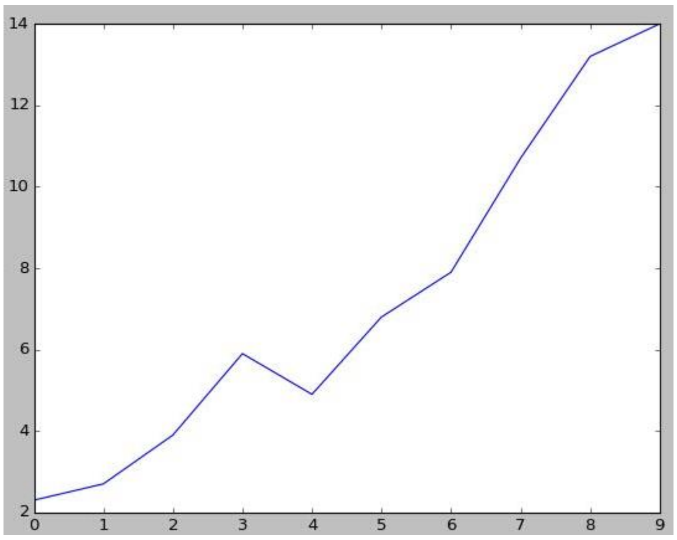


Figure 30 Average death toll with variation of *pro_infect*

We set $pro\_infect = init\_pro\_infect + 0.05 \times X$. It's unquestionable that the number of the infected will produce certain growth with the increasing in possibility of infection, which results in more deaths.

8.2 Evaluation for *pro_cure*

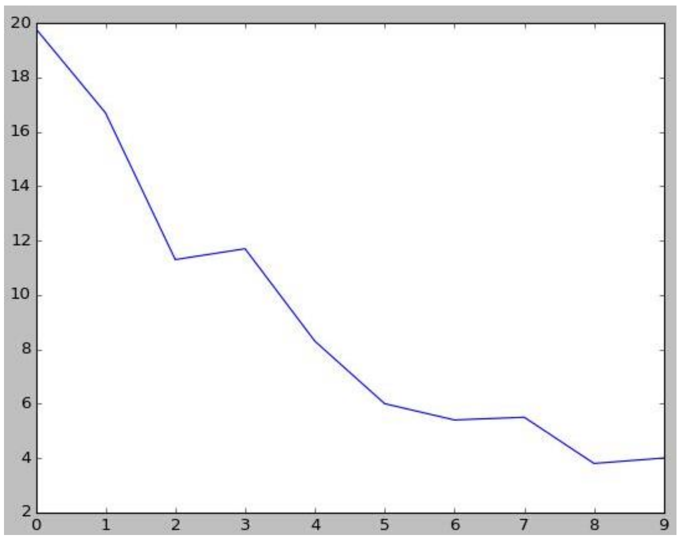


Figure 31 Average death toll with variation of *pro_cure*

We set $pro\_cure = init\_pro\_cure + 0.05 \times X$. It's obvious that the number of patients will reduce when the possibility of recovery rises up, which results in less deaths.

8.3 Evaluation for *pro_dead*

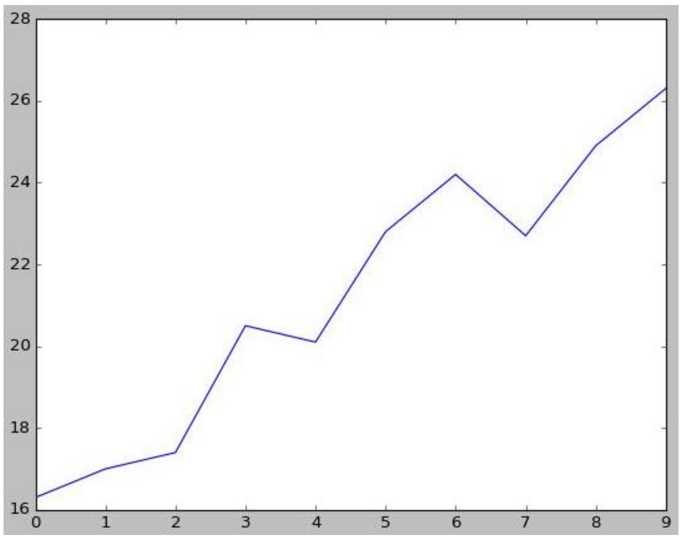


Figure 32 Average death toll with variation of *pro_dead*

We set $pro\_dead = init\_pro\_dead + 0.05 \times X$. The variation of death toll is inconspicuous comparing with the above 2 parameters, but the growing tendency is obvious. Dead people are out of infectivity; therefore the outbreaks of disease have weak relevance with the possibility of death. On the other hand, the final death toll is pertinent to it.

8.4 Evaluation for *medicine_cure*

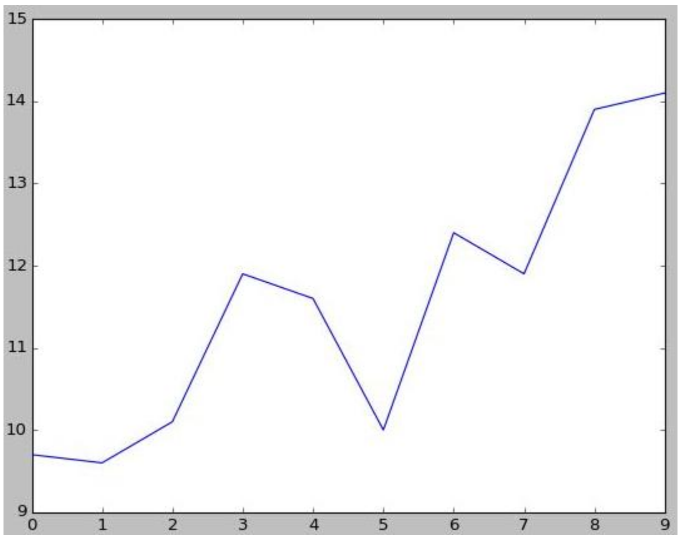


Figure 33 Average death toll with variation of *medicine_cure*

We set $medicine\_cure = init\_medicine\_cure - 0.05 \times X$. Because drugs are insufficient, only a few of patients can receive aid from them. The reduction of treatment effect has smaller impact on the final death toll than *pro_dead*.

9 Strength and Weaknesses

9.1 Strength Analysis

- **Fully consideration of spreading model**

  BA scale-free networks can well depict the real world relation networks. Alternate network model adds the difference of social networks and family networks, gives a further and more accurate simulation of real life. We base conventional SIRD model on the alternate networks, so that the role of household structure was taken into account. Thus, our model is much more refined than basic models, and simulation results are similar to real cases.

- **Suitable introduction of geographic information**

  Fruchterman-Reingold algorithm provides us a method to simulate spatial distribution of population. Only in this way can we connect infected people data with treatment center location and introduce request time in following steps of model.

9.2 Weaknesses Analysis

- **Randomness and timespan make the results not accurate enough**

  Our algorithm generates a random network and then implements calculation on it. Randomness leads to variable results, and inaccuracy becomes an inevitable problem. Besides, error will accumulate with increasing simulation timespan, which affect a lot.

- **Hard to select the suitable values of some parameters**

  In the process of generating networks, we can only determine probability parameters with necessary tests. As for the simulation of districts in West Africa, family effect and social effect are hard to compare. It is more difficult to choose parameters for SIRD model and total number of people.

- **Too much simplification of delivery system**

  Due to the complexity of modeling, we introduce quite a lot of simplification and hypothesis in the advanced model. Random network with geographic information has great difference with real settlements condition, and we assume that delivery problem as a requirement model of infected people. Under this circumstance, the model of delivery system will deviate from actual.

9.3 Further Research

- How to determine the value of parameters, including probability values, effect coefficient, and number of relative people;
- Introduce the effects of other districts into the holistic spreading model;
- More specific measure of medicine utility and more realistic delivery scheme;
- How to utilize the data of established treatment centers to optimize medicine utility.

10 Conclusions

In this report, we simulated the epidemic development trend in West Africa countries and minimize the death number with certain total medicine per time interval.

By constructing random networks and adding SIRD model, we performed elementary simulation. After testifying the effectiveness of spreading model, we introduced people's geographical data and calculated the suitable location of treatment center. This time the simulation results of infected people deviate from real series to some extent, while it could be accepted with so many random factors taken into consideration. Infected people would go to the treatment for medicine, so we changed our focus to medicine holding. We designed a request model on weighted networks with cure rate added on SIRD model. We chose the death number as the index of medicine delivery utility. Dynamic programming worked out the global optimum solution with given medicine stock.

Given that we knew almost nothing about the new medicine, we could only fit a curve of medicine holding and expected death toll on the whole. The final curve is a convex function, which accords with our percept that margin utility (the reduction of death toll) monotonically decrease with medicine holding increase.

Non-technical Letter for WMA

Ladies and Gentlemen,

The Ebola outbreak in Liberia, Guinea and Sierra Leone has presented the world an unprecedented challenge. The high fatality rate up to 50% once made the public so panicked. But now, we've got a chance to get rid of it once and for all. We, the World Medicine Association, successfully developed a kind of medicine that can cure patents whose situation is not serious.

We are appealing for international aid to help all 3 countries establish their own health systems. See. Our people are dying, Our children are in danger. This action allows of no delay.

According to the latest Sitrep, our researchers built a robust model to assess the severity of each area. To be more intuitional, we can describe the evaluation result by a hierarchy of emergency.

1. The following areas are the key sites that we must pay close attention to.

Liberia	Guinea	Sierra Leone
Montserrado	Macenta	Freetown

2. The following districts is effectively under controll through isolated treatment, personal protection, medical observation, sterilization, et al.

Liberia	Guinea	Sierra Leone
Lofa, Margibi	Conakry, Gueckedou,	Bo, Bombali, Kailahun, Kenema, Kono, Moyamba, Tonkolili, Western Rural

3. Only sporadic cases were reported in the following areas.

Liberia	Guinea	Sierra Leone
Bomi, Bong, Grand, Grand Cape Mount, Nimba, Rivercess	Beyla, Coyah, Dubreka, Faranah, Forecariah, Kerouane, Kissidougo, Lola, N'zerekore	Kambia, Koinadugu

4. No cases were reported in these places for a period of time or no outbreaks in the past. They could be considered safe.

Liberia	**Guinea**	**Sierra Leone**
Gbarpolu, Grand Gedeh, Grand Kru, Maryland, River Gee, Sinoe, Kindia	Boffa, Boke, Dabola, Dalaba, Dinguiraye, Fria, Kankan, Kouroussa, Mali, Pita, Siguir, Telimele, Tougue, Yomou	None

With the good news always comes the bad. Though we are producing at full capacity, due to the complicated procedures and preciousness of ingredients, there still exists a huge gap between production speed and actual needs. We could only afford average supply of 10 courses of treatment for each district per day.

In order to confront difficulties and to fight against obstacles, we will assign higher priority to those districts in higher risk when designing the supply plan.

All medicines will be offered publicly in the treatment centers, following the principle of First-Come-First-Serve. We will definitely try our best to cure every patient. By no means will we give up on anyone.

God bless you all!

References

[1] World Health Organization. *WHO | Ebola virus disease*. Retrieved from http://www.who.int/mediacentre/factsheets/fs103/en/

[2] World Health Organization. *Ebola Situation Report - 4 February 2015*. Retrieved from http://apps.who.int/ebola/en/ebola-situation-report/situation-reports/ebola-situation-report-4-february-2015

[3] Kermack, W. O., & McKendrick, A. G. (1927, August). A contribution to the mathematical theory of epidemics. In *Proceedings of the Royal Societ y of London A: Mathematical, Physical and Engineering Sciences* (Vol. 115, No. 772, pp. 700-721). The Royal Society.

[4] Sirakoulis, G. C., Karafyllidis, I., & Thanailakis, A. (2000). A cellular aut omaton model for the effects of population movement and vaccination on epidemic propagation. *Ecological Modelling*, 133(3), 209-223.

[5] Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-worl d'networks. *nature*, 393(6684), 440-442.

[6] Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random net works. *science*, 286(5439), 509-512.

[7] Ni, S. J. (2009). Research on Modeling of Infectious Disease Spreading B ased on Complex Network Theory (Doctoral dissertation). Tsinghua Univer sity.

[8] Liberia Ministry of Health & Social Welfare. *Situation Report on the E BOLA Virus disease epidemic in Liberia as of March 2014 to 31st January, 2015*. Retrieved from http://www.mohsw.gov.lr/documents/Sitrep%20261%20Jan%2031st%202015.pdf

[9] African Health Observatory - WHO. Liberia & Guinea: Analytical sum mary - Social determinants. Retrieved from http://www.aho.afro.who.int/profiles_information/index.php/Liberia:Analytical_summary_-_Social_determinants & http://www.aho.afro.who.int/profiles_information/index.php/Guinea:Analytical_summary_-_Social_determinants

[10] Statistics Sierra Leone and ICF Macro. 2009. *Sierra Leone Demographic and Health Survey 2008: Key Findings*. Calverton, Maryland, USA: SSL and ICF Macro.

[11] Ni, S., Weng, W., & Fan, W. (2008). Threshold of SIS epidemics in alternate social networks. *Acta Physica Polonica B*, 39(3), 739.

[12] Kiskowski, M. A. (2014). A Three-Scale Network Model for the Early Growth Dynamics of 2014 West Africa Ebola Epidemic. PLOS Currents Outbreaks.

[13] Fruchterman, T. M., & Reingold, E. M. (1991). Graph drawing by forced irected placement. *Software: Practice and experience*, 21(11), 1129-1164.

[14] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web.

[15] Bellman, R. (1956). Dynamic programming and Lagrange multipliers. Proceedings of the National Academy of Sciences of the United States of America, 42(10), 767.

[16] Bollobás, B., Das, G., Gunopulos, D., & Mannila, H. (1997, August). Time-series similarity problems and well-separated geometric sets. In *Proceedings of the thirteenth annual symposium on Computational geometry* (pp. 454-456). ACM.

[17] World Health Organization. *Ebola data and statistics*. Retrieved from http://apps.who.int/gho/data/node.ebola-sitrep.ebola-country?lang=en.

Appendix

- **Python source code for spreading simulation**

```
import sys
import random
import operator
import networkx as nx
import matplotlib.pyplot as plt
import pylab as pl

def formation(BA, init_infect, init_risk, init_cure, init_dead):
    hav_infect = 0
    while (hav_infect < init_infect):
        pick = random.choice(BA.nodes())
        if (BA.node[pick]['color']!='w'): continue
        BA.node[pick]['color'] = 'r'
        hav_infect += 1
    hav_risk = 0
    while (hav_risk < init_risk):
        pick = random.choice(BA.nodes())
        if (BA.node[pick]['color']!='w'): continue
        #BA.node[pick]['color'] = 'b'
        BA.node[pick]['color'] = 'w'
        hav_risk += 1
    hav_cure = 0
    while (hav_cure < init_cure):
        pick = random.choice(BA.nodes())
        if (BA.node[pick]['color']!='w'): continue
        BA.node[pick]['color'] = 'g'
        hav_cure += 1
    hav_dead = 0
    while (hav_dead < init_dead):
        pick = random.choice(BA.nodes())
        if (BA.node[pick]['color']!='w'): continue
        BA.node[pick]['color'] = 'b'
        hav_dead += 1

def distan(k1, k2):
    return ((pos[k1][0]-pos[k2][0])**2 + (pos[k1][1]-pos[k2][1])**2)**0.5

def calc(a, b, c):
    if (a<b): return 1
    return (b/a)**c

def iterator_graph(BA, pro_infect, pro_cure, pro_dead, const_distan,
ratio_distan, dis_min, time_family, time_social):

    def infect(a, b, center, pro_infect, pro_distan):
        poss = random.random()
        t_dis = distan(a,b)
        poss = poss/calc(t_dis, dis_min, const_distan)
        if (poss<pro_infect):
            if ((BA.node[a]['color']=='r') & (BA.node[b]['color']=='w')):
                BA.node[b]['color']='r'
                BA.node[b]['request'] = distan(b, center) * ratio_distan
                return 1
            if ((BA.node[a]['color']=='w') & (BA.node[b]['color']=='r')):
                BA.node[a]['color']='r'
                BA.node[a]['request'] = distan(a, center) * ratio_distan
                return 1
        return 0

    t_del_infect = 0

    for t in range(time_family):
      for a in BA.nodes():
            # try to infect
            if (BA.node[a]['color']!='r'): continue
            fa = BA.node[a]['family']
            tot = BA.node[a]['belong']
            if (tot == 1): continue
            b = a - fa
            while (fa + b == a):
                b = random.randint(0, tot-1)
            b = fa + b
            t_del_infect += infect(a, b, center, pro_infect, const_distan)

    for t in range(time_social):
        for a in BA.nodes():
            if (BA.node[a]['color']!='r'): continue
            b = a
            while (BA.node[b]['family'] == BA.node[a]['family']):
                b = random.choice(BA.neighbors(a))
            t_del_infect += infect(a, b, center, pro_infect, const_distan)
    return BA, t_del_infect

def barabasi_albert_graph(n, m, k):
    G=nx.empty_graph(m)
    targets=range(m)
    repeated_nodes=[]
    source=0 # notice 0

    while source<n:
        add=random.randint(1,k)
        if (add>n-source): add = n-source
        for i in range(k):
            G.add_node(source+i, color='w', family=source, belong=add,
request=0, hav_medicine=False)
            for j in range(i+1,k):
                if (i==j): continue
                G.add_edges_from(zip([source+i],[source+j]))

            G.add_edges_from(zip([source+i]*m,targets))
            repeated_nodes.extend(targets)
            repeated_nodes.extend([source+i]*m)

        targets=set()
        while len(targets)<m:
            x=nx.random_graphs.random.choice(repeated_nodes)
            targets.add(x)
        source += add
    return G

def nature(a, pro_cure, pro_dead):
    if (BA.node[a]['hav_medicine']==False):
        poss = random.random()
        if (poss<pro_cure):
            if (BA.node[a]['color']=='r'):
                BA.node[a]['color']='g'
        else:
            poss = random.random()
            if (poss<pro_dead):
                if (BA.node[a]['color']=='r'):
                    BA.node[a]['color']='b'
    elif (BA.node[a]['hav_medicine']==True):
        poss = random.random()
        if (poss<pro_cure+delta_cure):
            if (BA.node[a]['color']=='r'):
                BA.node[a]['color']='g'
        else:
            poss = random.random()
            if (poss<pro_dead):
                if (BA.node[a]['color']=='r'):
                    BA.node[a]['color']='b'

def heal(queue, pro_cure, pro_dead, put_medicine):
    ans = []
    use = 0
    for a in queue:
        if (use<put_medicine):
            use += 1
            poss = random.random()
            BA.node[a]['hav_medicine']=True
            del treatment_hash[a]
            if (poss<pro_cure+delta_cure):
                if (BA.node[a]['color']=='r'):
                    BA.node[a]['color']='g'
            else:
                poss = random.random()
                if (poss<pro_dead):
                    if (BA.node[a]['color']=='r'):
                        BA.node[a]['color']='b'
        else:
            poss = random.random()
            if (poss<pro_cure):
                if (BA.node[a]['color']=='r'):
                    BA.node[a]['color']='g'
                    del treatment_hash[a]
            else:
                poss = random.random()
                if (poss<pro_dead):
                    if (BA.node[a]['color']=='r'):
                        BA.node[a]['color']='b'
                        del treatment_hash[a]
                else:
                    ans.append(a)
    return ans

if __name__ == '__main__':

    # 4 const, related to graph
    max_people = 300
    each_family = 6
    confirmed_people = 2
    probable_people = 0

    times = 10
    heal_times = 5
    pro_infect = 0.3
    pro_cure = 0.001
    pro_dead = 0.15
    delta_cure = 1
    const_distan = 3
    ratio_distan = 2
    per_day = 0.3 # this para is const
    medicine = 5
    dis_min = 0.2
    time_family = 2
    time_social = 1

    treatment_queue = []
    treatment_hash = {}

    BA= barabasi_albert_graph(max_people,1,each_family)
    formation(BA,confirmed_people,max_people - confirmed_people -
probable_people,0,0)
    #pos = nx.spring_layout(BA)
    #plt.show("init")
    #plt.savefig("init.png")

    infect=[]
    cure=[]
    dead=[]
    del_infect=[]
    # assign positions for people
    pos = nx.spring_layout(BA)

    # pick center
    bc = nx.eigenvector_centrality(BA)
    bc = sorted(bc.items(), key = operator.itemgetter(1), reverse = True)
    center = bc[0][0]
    #print(bc)
    tot_infect = 0
    for i in range(times):
        t_infect = 0
        t_cure = 0
        t_dead = 0
        BA, t_del_infect = iterator_graph(BA, pro_infect, pro_cure, pro_dead,
const_distan, ratio_distan, dis_min, time_family, time_social)
        for a in BA.nodes():
            nature(a, pro_cure, pro_dead)

        for a in BA.nodes():
            if (BA.node[a]['color']=='r'): t_infect += 1
            elif (BA.node[a]['color']=='g'): t_cure += 1
            elif (BA.node[a]['color']=='b'): t_dead += 1
        infect.append(t_infect)
        cure.append(t_cure)
        dead.append(t_dead)
        del_infect.append(t_del_infect)
        tot_infect += t_del_infect
        print("time = %d, del_infect = %d" %(i,del_infect[i]))
        #print("time = %d, infect = %d, cure = %d, dead
= %d" %(i,infect[i],cure[i],dead[i]))

    print("totel infect = %d" %(tot_infect))
    nature_death = dead.pop()
    print("nature death = %d" %(nature_death))
    pl.plot(range(times),del_infect)
    pl.show()

    #control = input("continue?")
    #if (control == 0): sys.exit()

    colorlist=[]
    for i in range(BA.number_of_nodes()):
        colorlist.append(BA.node[i]['color'])

    #BA.node[center]['color'] = 'y'
    nx.draw(BA,pos,with_labels=False,node_size=30,node_color=colorlist)
    plt.show("result")

    save_BA = BA.to_undirected()
    repeat_times = 10
    save_tot_death = []
    #for put_medicine in range(medicine):
    control_del = 10
    save = pro_dead
    for count in range(control_del):
        pro_dead = save
        pro_dead += count * 0.05

        put_medicine = medicine
        tot_death = 0.0

        for repeat in range(repeat_times):
            BA = save_BA.to_undirected()

            dead = []
            del_infect = []
            for i in range(heal_times):
                t_infect = 0
                t_cure = 0
                t_dead = 0
                # t_del_infect = 0
                BA, t_del_infect = iterator_graph(BA, pro_infect, pro_cure,
pro_dead, const_distan, ratio_distan, dis_min, time_family, time_social)
                for a in BA.nodes():
                    #cure(a, pro_cure, pro_dead)
                    if (BA.node[a]['color']=='r'):
                        if (BA.node[a]['request']>0): BA.node[a]['request'] -=
per_day
                        if (BA.node[a]['request']<=0):
                            if (a in treatment_hash.iterkeys())==False:
                                treatment_hash[a]=1
                                treatment_queue.append(a)
                    if (a in treatment_hash.iterkeys())==False:
                        if (BA.node[a]['color']=='r'):
                            nature(a, pro_cure, pro_dead)

                treatment_queue = heal(treatment_queue, pro_cure, pro_dead,
put_medicine)

                for a in BA.nodes():
                    if (BA.node[a]['color']=='r'): t_infect += 1
                    elif (BA.node[a]['color']=='g'): t_cure += 1
                    elif (BA.node[a]['color']=='b'): t_dead += 1
                dead.append(t_dead)
                del_infect.append(t_del_infect)
            #print(dead.pop() - nature_death)
            death_tmp = dead.pop() - nature_death

            tot_death += death_tmp
        float_times = 0.0
        float_times = repeat_times
        save_tot_death.append(tot_death/float_times)
        #print("medicine = %d, total death = %.1f" %(put_medicine,
tot_death/float_times))

    #print(save_tot_)
    pl.plot(range(control_del),save_tot_death)
    pl.show()
    #plt.savefig("result.png")
```

- **Python source code for dynamic programing**

```
import pylab as pl

tot_area = 12
#tot_medicine =
#if (tot_area * 9 + 1 < tot_medicine):
tot_medicine = tot_area * 9 + 1

predict_death = [[0 for col in range(10)] for row in range(tot_area)]
pick_medicine = [[0 for col in range(tot_medicine)] for row in range(tot_area)]
min_death = [20000] * tot_medicine

# Sierra_western rural
predict_death[0] =
[216.2392,159.041,140.2521,127.6984,121.4631,116.1423,111.4035,106.5816,101
.2608,98.26789]
# Sierra_tonkolili
predict_death[1] =
[103.4915,93.42016,63.90078,49.54625,49.89354,35.19173,33.68682,31.14005,27
.89871,25.35194]
# Sierra_kono
predict_death[2] =
[154.223,101.980,85.819,72.414,68.780,60.512,56.252,51.366,48.610,45.102]
# Sierra_Kenema
predict_death[3] =
[16.568,8.559,4.718,1.536,1.865,1.317,1.756,1.975,1.756,2.195]
# Sierra_Bombali
predict_death[4] =
[306.599,188.534,158.036,145.329,135.740,126.152,116.101,110.671,104.664,99
.581]
```

```
# Sierra_BO
predict_death[5] =
[84.521,71.431,66.710,63.753,61.361,58.025,56.200,52.550,49.592,47.515]
# Sierra_FREETOWN
predict_death[6] =
[374.851,337.250,334.322,330.238,324.074,319.143,321.840,318.604,319.451,31
9.837]
# Liberia_Margibi
predict_death[7] =
[73.320,55.569,44.088,37.142,30.679,27.398,25.565,21.417,21.031,19.970]
# Liberia_Lofa
predict_death[8] = [39.2,24.2,18.4,14.4,9.1,8.6,8,7.3,4.6,5.2]
# Liberia_Montserrado
predict_death[9] =
[689.581,442.748,377.164,343.114,327.818,315.455,307.807,300.055,292.407,28
5.073]
# Guinea_conakry
predict_death[10] =
[56.272,36.341,28.832,24.289,21.230,18.356,16.872,14.184,13.999,11.959]
# Guinea_macenta
predict_death[11] =
[208.861,167.776,159.761,156.393,150.938,147.234,145.617,139.825,137.467,13
3.089]

for put_area in range(tot_area):
    for put_medicine in range(10):
        if (put_medicine > 0 ) &(predict_death[put_area][put_medicine] >
predict_death[put_area][put_medicine-1]):
            predict_death[put_area][put_medicine] =
predict_death[put_area][put_medicine-1]

for put_area in range(tot_area):
    if (put_area == 0):
        for put_medicine in range(10):
            min_death[put_medicine] = predict_death[put_area][put_medicine]
            pick_medicine[put_area][put_medicine] = put_medicine
        #print(min_death)
        continue
    for add_medicine in range(tot_medicine-1,-1,-1):
        min_death[add_medicine] += predict_death[put_area][0]
        pick_medicine[put_area][add_medicine] = 0
        for put_medicine in range(10):
            if (add_medicine - put_medicine < 0 ): break
            tmp = min_death[add_medicine - put_medicine] +
predict_death[put_area][put_medicine]
          if (min_death[add_medicine] > tmp):
             min_death[add_medicine] = tmp
             pick_medicine[put_area][add_medicine] = put_medicine

#print(min_death)

pl.plot(range(tot_medicine), min_death)
pl.show()

def output(now_area, last_medicine):
   if (now_area == 0)& (last_medicine == 0): return
   output(now_area-1, last_medicine -
pick_medicine[now_area-1][last_medicine])
   path.append(pick_medicine[now_area-1][last_medicine])

for add_medicine in range(tot_medicine-1,-1,-1):
   path = []
   output(tot_area, add_medicine)
   print("%.4f :" %(min_death[add_medicine]))
   print(path)
```

- **Result of dynamic programing: distribution scenario**

```
1090.4298 :
[9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9]
1090.4298 :
[9, 9, 9, 9, 9, 9, 9, 9, 8, 9, 9, 9]
1090.4298 :
[9, 9, 9, 9, 9, 9, 8, 9, 8, 9, 9, 9]
1090.4298 :
[9, 9, 9, 9, 9, 9, 7, 9, 8, 9, 9, 9]
1090.4298 :
[9, 9, 9, 8, 9, 9, 7, 9, 8, 9, 9, 9]
1090.4298 :
[9, 9, 9, 7, 9, 9, 7, 9, 8, 9, 9, 9]
1090.4298 :
[9, 9, 9, 6, 9, 9, 7, 9, 8, 9, 9, 9]
1090.4298 :
[9, 9, 9, 5, 9, 9, 7, 9, 8, 9, 9, 9]
```

```
1090.6488 :
[9, 9, 9, 4, 9, 9, 7, 9, 8, 9, 9, 9]
1090.6488 :
[9, 9, 9, 3, 9, 9, 7, 9, 8, 9, 9, 9]
1091.1878 :
[9, 9, 9, 4, 9, 9, 5, 9, 8, 9, 9, 9]
1091.1878 :
[9, 9, 9, 3, 9, 9, 5, 9, 8, 9, 9, 9]
1092.2488 :
[9, 9, 9, 3, 9, 9, 5, 8, 8, 9, 9, 9]
1092.6348 :
[9, 9, 9, 3, 9, 9, 5, 7, 8, 9, 9, 9]
1094.4738 :
[9, 9, 9, 3, 9, 9, 5, 8, 8, 9, 7, 9]
1094.8598 :
[9, 9, 9, 3, 9, 9, 5, 7, 8, 9, 7, 9]
1096.6348 :
[9, 9, 9, 3, 9, 9, 5, 7, 5, 9, 9, 9]
1097.1348 :
[9, 9, 9, 3, 9, 9, 5, 7, 4, 9, 9, 9]
1098.8598 :
[9, 9, 9, 3, 9, 9, 5, 7, 5, 9, 7, 9]
1099.3598 :
[9, 9, 9, 3, 9, 9, 5, 7, 4, 9, 7, 9]
1101.4368 :
[9, 9, 9, 3, 9, 8, 5, 7, 4, 9, 7, 9]
1103.5318 :
[9, 9, 9, 3, 9, 9, 5, 7, 4, 9, 5, 9]
1105.6088 :
[9, 9, 9, 3, 9, 8, 5, 7, 4, 9, 5, 9]
1108.1556 :
[9, 8, 9, 3, 9, 8, 5, 7, 4, 9, 5, 9]
1111.0296 :
[9, 8, 9, 3, 9, 8, 5, 7, 4, 9, 4, 9]
1113.3716 :
[9, 5, 9, 3, 9, 9, 5, 7, 4, 9, 5, 9]
1115.4486 :
[9, 5, 9, 3, 9, 8, 5, 7, 4, 9, 5, 9]
1118.3226 :
```

```
[9, 5, 9, 3, 9, 8, 5, 7, 4, 9, 4, 9]
1121.2806 :
[9, 5, 9, 3, 9, 7, 5, 7, 4, 9, 4, 9]
1123.8816 :
[9, 5, 9, 3, 9, 5, 5, 7, 4, 9, 5, 9]
1126.7556 :
[9, 5, 9, 3, 9, 5, 5, 7, 4, 9, 4, 9]
1129.6096 :
[9, 5, 9, 3, 9, 3, 5, 7, 4, 9, 5, 9]
1132.4836 :
[9, 5, 9, 3, 9, 3, 5, 7, 4, 9, 4, 9]
1135.4406 :
[9, 5, 9, 3, 9, 2, 5, 7, 4, 9, 4, 9]
1138.4335 :
[8, 5, 9, 3, 9, 2, 5, 7, 4, 9, 4, 9]
1141.4216 :
[9, 5, 9, 3, 9, 2, 5, 5, 4, 9, 4, 9]
1144.4145 :
[8, 5, 9, 3, 9, 2, 5, 5, 4, 9, 4, 9]
1147.4735 :
[8, 5, 9, 3, 9, 2, 5, 5, 4, 9, 3, 9]
1150.6555 :
[8, 5, 9, 2, 9, 2, 5, 5, 4, 9, 3, 9]
1153.7375 :
[8, 5, 7, 3, 9, 2, 5, 5, 4, 9, 3, 9]
1156.9195 :
[8, 5, 7, 2, 9, 2, 5, 5, 4, 9, 3, 9]
1160.2005 :
[8, 5, 7, 2, 9, 2, 5, 4, 4, 9, 3, 9]
1163.6555 :
[8, 5, 7, 2, 9, 2, 5, 5, 4, 9, 3, 7]
1166.9365 :
[8, 5, 7, 2, 9, 2, 5, 4, 4, 9, 3, 7]
1170.7775 :
[8, 5, 7, 1, 9, 2, 5, 4, 4, 9, 3, 7]
1174.3455 :
[8, 5, 7, 2, 9, 2, 5, 4, 4, 9, 3, 5]
1178.0495 :
[8, 5, 7, 2, 9, 2, 5, 4, 4, 9, 3, 4]
```

```
1181.8905 :
[8, 5, 7, 1, 9, 2, 5, 4, 4, 9, 3, 4]
1186.4335 :
[8, 5, 7, 1, 9, 2, 5, 4, 4, 9, 2, 4]
1190.7135 :
[8, 5, 7, 1, 9, 2, 5, 4, 4, 9, 3, 2]
1195.2565 :
[8, 5, 7, 1, 9, 2, 5, 4, 4, 9, 2, 2]
1199.8595 :
[8, 5, 5, 1, 9, 2, 5, 4, 4, 9, 3, 2]
1204.4025 :
[8, 5, 5, 1, 9, 2, 5, 4, 4, 9, 2, 2]
1208.8205 :
[8, 5, 7, 1, 9, 2, 1, 4, 4, 9, 3, 2]
1213.3635 :
[8, 5, 7, 1, 9, 2, 1, 4, 4, 9, 2, 2]
1217.9665 :
[8, 5, 5, 1, 9, 2, 1, 4, 4, 9, 3, 2]
1222.5095 :
[8, 5, 5, 1, 9, 2, 1, 4, 4, 9, 2, 2]
1227.2305 :
[8, 5, 5, 1, 9, 1, 1, 4, 4, 9, 2, 2]
1231.8095 :
[8, 5, 5, 1, 9, 2, 1, 4, 2, 9, 2, 2]
1236.5305 :
[8, 5, 5, 1, 9, 1, 1, 4, 2, 9, 2, 2]
1241.6135 :
[8, 5, 5, 1, 8, 1, 1, 4, 2, 9, 2, 2]
1246.6732 :
[6, 5, 5, 1, 9, 1, 1, 4, 2, 9, 2, 2]
1251.4120 :
[5, 5, 5, 1, 9, 1, 1, 4, 2, 9, 2, 2]
1256.4950 :
[5, 5, 5, 1, 8, 1, 1, 4, 2, 9, 2, 2]
1261.8158 :
[4, 5, 5, 1, 8, 1, 1, 4, 2, 9, 2, 2]
1267.6158 :
[4, 5, 5, 1, 8, 1, 1, 4, 1, 9, 2, 2]
1273.2528 :
```

```
[4, 5, 5, 1, 6, 1, 1, 4, 2, 9, 2, 2]
1279.0528 :
[4, 5, 5, 1, 6, 1, 1, 4, 1, 9, 2, 2]
1285.1548 :
[4, 5, 3, 1, 6, 1, 1, 4, 2, 9, 2, 2]
1290.9548 :
[4, 5, 3, 1, 6, 1, 1, 4, 1, 9, 2, 2]
1297.1901 :
[3, 5, 3, 1, 6, 1, 1, 4, 1, 9, 2, 2]
1303.6531 :
[3, 5, 3, 1, 6, 1, 1, 3, 1, 9, 2, 2]
1310.5991 :
[3, 5, 3, 1, 6, 1, 1, 2, 1, 9, 2, 2]
1317.9331 :
[3, 5, 3, 1, 6, 1, 1, 2, 1, 8, 2, 2]
1324.9537 :
[3, 3, 3, 1, 6, 1, 1, 2, 1, 9, 2, 2]
1332.2876 :
[3, 3, 3, 1, 6, 1, 1, 2, 1, 8, 2, 2]
1339.7966 :
[3, 3, 3, 1, 6, 1, 1, 2, 1, 8, 1, 2]
1347.4446 :
[3, 3, 3, 1, 6, 1, 1, 2, 1, 7, 1, 2]
1355.1966 :
[3, 3, 3, 1, 6, 1, 1, 2, 1, 6, 1, 2]
1362.8446 :
[3, 3, 3, 1, 6, 1, 1, 2, 1, 5, 1, 2]
1370.8536 :
[3, 3, 3, 0, 6, 1, 1, 2, 1, 5, 1, 2]
1378.8686 :
[3, 3, 3, 0, 6, 1, 1, 2, 1, 5, 1, 1]
1388.9196 :
[3, 3, 3, 0, 5, 1, 1, 2, 1, 5, 1, 1]
1398.5077 :
[3, 3, 3, 0, 4, 1, 1, 2, 1, 5, 1, 1]
1408.0966 :
[3, 3, 3, 0, 3, 1, 1, 2, 1, 5, 1, 1]
1419.5776 :
[3, 3, 3, 0, 3, 1, 1, 1, 1, 5, 1, 1]
```

```
1431.9407 :
[3, 3, 3, 0, 3, 1, 1, 1, 1, 4, 1, 1]
1444.4943 :
[2, 3, 3, 0, 3, 1, 1, 1, 1, 4, 1, 1]
1457.2014 :
[2, 3, 3, 0, 2, 1, 1, 1, 1, 4, 1, 1]
1470.2913 :
[2, 3, 3, 0, 2, 0, 1, 1, 1, 4, 1, 1]
1483.6963 :
[2, 3, 2, 0, 2, 0, 1, 1, 1, 4, 1, 1]
1498.0509 :
[2, 2, 2, 0, 2, 0, 1, 1, 1, 4, 1, 1]
1513.0509 :
[2, 2, 2, 0, 2, 0, 1, 1, 0, 4, 1, 1]
1528.3469 :
[2, 2, 2, 0, 2, 0, 1, 1, 0, 3, 1, 1]
1544.5079 :
[2, 2, 1, 0, 2, 0, 1, 1, 0, 3, 1, 1]
1562.2589 :
[2, 2, 1, 0, 2, 0, 1, 0, 0, 3, 1, 1]
1581.0478 :
[1, 2, 1, 0, 2, 0, 1, 0, 0, 3, 1, 1]
1600.9788 :
[1, 2, 1, 0, 2, 0, 1, 0, 0, 3, 0, 1]
1620.6385 :
[1, 0, 1, 0, 2, 0, 1, 0, 0, 3, 1, 1]
1640.5695 :
[1, 0, 1, 0, 2, 0, 1, 0, 0, 3, 0, 1]
1671.0675 :
[1, 0, 1, 0, 1, 0, 1, 0, 0, 3, 0, 1]
1705.1175 :
[1, 0, 1, 0, 1, 0, 1, 0, 0, 2, 0, 1]
1742.7185 :
[1, 0, 1, 0, 1, 0, 0, 0, 0, 2, 0, 1]
1783.8035 :
[1, 0, 1, 0, 1, 0, 0, 0, 0, 2, 0, 0]
1836.0465 :
[1, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0]
1893.2447 :
```

```
[0,  0,  0,  0,  1,  0,  0,  0,  0,  2,  0,  0]
1958.8287 :
[0,  0,  0,  0,  1,  0,  0,  0,  0,  1,  0,  0]
2076.8937 :
[0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0]
2323.7267 :
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
```