Modeling the Spread of Epidemics Thorough Network Simulation

Daniel Koo[a], **Aleksandre Ninua**[a], **and Aditya Pimplaskar**[a]

[a]University of California, Los Angeles

This manuscript was compiled on June 10, 2020

In this article, we simulate and examine the spread of infection on random networks as well as a real life social network of interactions in a school, while adjusting and accounting for disease spread parameters like infection rate, contact rate, and mortality. Our general method of simulation utilizes a breadth-first algorithm that cause someone to possibly transmit the disease to their neighbors. We also incorporated variables such as nodes "quarantining" themselves and not interacting with their neighbors. Our study showed that centrality analysis of patient zero can predict the size of the outbreak, with positive correlation of centrality and infections and casualties. Through the cost-benefit analysis we determined a threshold quarantine rate of 0.60, which is the minimum requirement for net positive production in society.

simulation | networks | epidemic | disease transmission |

The spread of epidemics are frequently modeled through simulations on various networks. These models can be valuable tools in assessing the appropriate responses and countermeasures to a pandemic. To give a simple view of how a disease can spread, we simulate an arbitrary epidemic over a social network of a school (1). Additionally, a temporal factor is included as the disease is simulated over a number of days as explained in our *Methods* section. The simulation is affected by several different variables such as transmission rate, chances of becoming symptomatic, self-isolation from the network, and more.

Our analysis is mainly dependent on examining the spread of the network over different rates of quarantine i.e. the probability that a node will isolate themselves from the rest of the network. We later discuss the significance that the quarantine rate has on reducing numbers of infection and death throughout our time frame. We also produce the outcomes based on choosing various seed nodes (patient zeroes) with different centralities to demonstrate that certain infected nodes can be "super-spreaders" and cause a more rapid expansion of the disease through the network.

Finally, the article aims to find optimal rates of quarantine such that we are maximizing economic output and minimizing the number of deaths. We measured the "value" of our network with a cost-benefit function based on economic data such as GDP. We find that the optimal quarantine rate is, in fact, still very high as deaths were given a very large cost against the economic value of the network.

Methods

The Model. In considering the status of each node, we created compartments that nodes can be classified into; these compartments include: susceptible, infected asymptomatic, infected symptomatic, quarantined, recovered, and deceased.

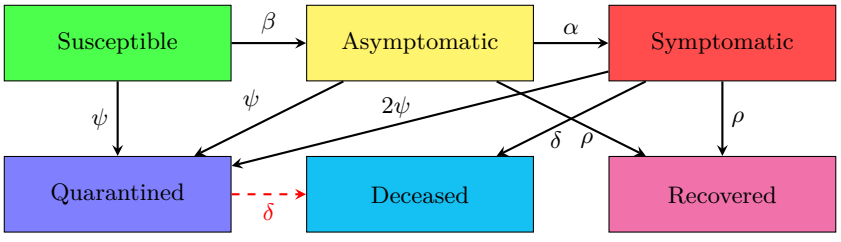


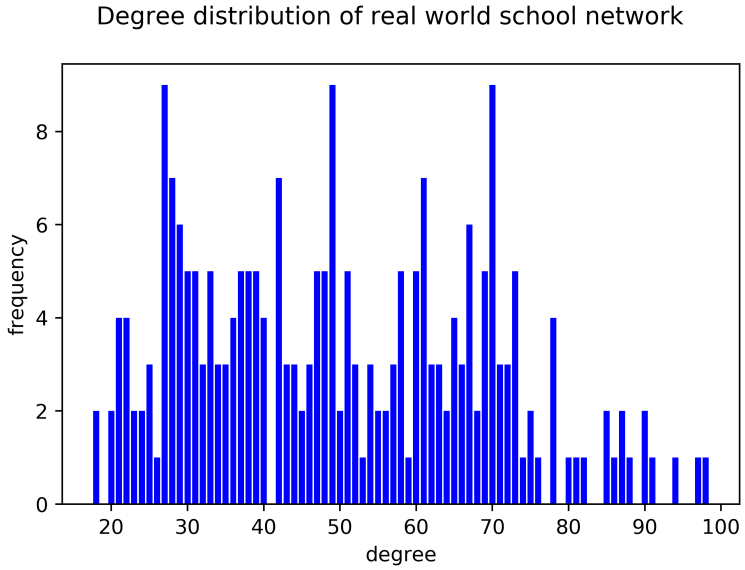
Fig. 1. Schematic diagram of our compartmental model and probabilities of flows between compartments.

Significance Statement

The global pandemic of COVID-19 has sparked a public discussion on the spread of epidemics and "flattening the curve". We can draw some important conclusions from modeling the spread of such diseases through network simulations. These simulations can also provide us with insights into the importance of disease parameters as well as various interventions such as social distancing and self-isolation, and how these variables affect the rate at which a disease can spread. With these results, we can better inform the reasoning behind some of the drastic social policies implemented during this situation.

We simulate the spread of the disease using a modified breadth-first traversal from every infected node (initially from the Patient Zero). Each infected node infects its susceptible neighbors with rate β. The infection rate is identical between symptomatic and asymptomatic nodes, as each edge represents an interaction that would be sufficient for the possible transmission. This is further compensated by the fact that symptomatic nodes have a higher rate of self-quarantining, and it is also consistent with the network design in *High-Resolution Measurements of Face-to-Face Contact Patterns in a Primary School.*, where they maintained a threshold time for an interaction to have infectious potential (1).

For our model we use a real life network derived by observing schoolchildren and teachers during their day at the school from *High-Resolution Measurements of Face-to-Face Contact Patterns in a Primary School.* The network consisted of 236 nodes with 226 students and 10 teachers. We believe that this network is also a good representation of general daily interactions in a closed environment - such as offices or prisons. Such environments are much more prone to outbreaks due to enclosed spaces with limited ventilation - and also because people interact not only with their close friends, but they also have to interact with coworkers and classmates, creating edges between otherwise isolated clusters. The average degree in the network is 49, with diameter 3 - with a total of 236 nodes present - it is quite tight knit. Therefore spreading a contagious disease can be very easy in our network, and precisely because of that, we can more accurately observe effects of mitigation efforts. Due to the tight structure of this network, we expect the epidemic spread to occur much more rapidly than a simulation on a more traditional and "open" social network.



Property	Value
Radius	2
Diameter	3
Average shortest path	1.86
Average degree	49.99
Average neighbor degree	55.69

Every node has a chance of quarantining themselves at a rate ψ, or if they're symptomatic, 2ψ. Quarantined nodes are removed from the network, they cannot pass the disease or get infected - but if they were already infected by the time they went into quarantine, they still have a chance to develop symptoms, recover, or die. Similarly, deceased nodes are ignored from the network. Recovered nodes remain in quarantine, as we did not want to make assumptions about immunity or infectiousness of recovered patients.

Every node is assigned a symptom onset, recovery, and death rate randomly chosen from Gamma distributions $s \sim \Gamma(5.81, 0.95)$, $r \sim \Gamma(8.16, 0.33)$, $d \sim \Gamma(4.94, 0.26)$ respectively (2, 3). These rates were approximated using early COVID-19 data, and represent average times for symptom onset (6.11 days), recovery (24.7 days), and death (18.8 days). These randomly assigned rates are meant to reflect the different experiences of dealing with the virus across health, and age ranges.

Algorithm 1 Daily infection crawl through the network

At the start of each day, the infection starts spreading from all the infected people in the queue;
while *queue is not empty* **do**
  for *each neighbor of the node at the top of the queue* **do**
    if *this neighbor is alive, has never been infected, and is not quarantined* **then**
      Infect them with probability β and if infected, add them to the queue
    end
  end
end
At the end of each day;
for *every node in the network* **do**
  Quarantine the node with probability ψ, or 2ψ if they're symptomatic
end
for *every infected node* **do**
  if *Asymptomatic* **then**
    recover or develop symptoms at a unique rate
  else
    recover or die at a unique rate
  end
  if *Alive, still infected, and not quarantined* **then**
    put them back in the queue
  end
end

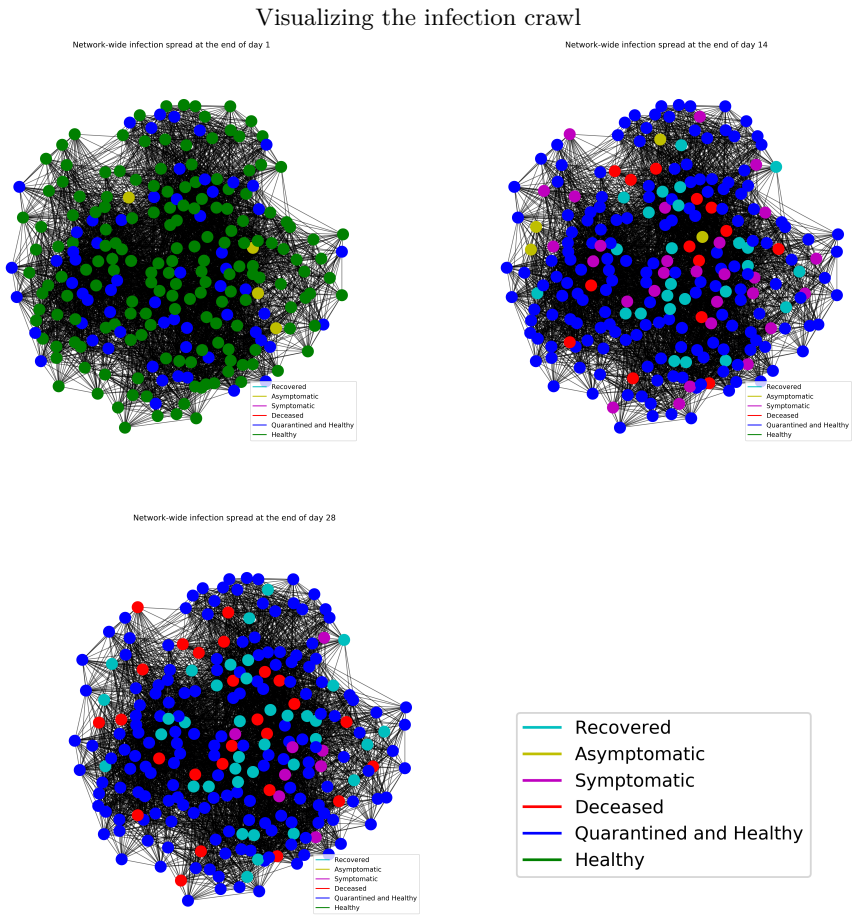


Fig. 2. This figure depicts the spread of the disease from a random seed node in our network. Nodes are color-labelled based on the compartments that they fall into. As each day passes, each node is subject to moving compartments as described in the algorithm above.

Seed selection for centrality analysis.

For this analysis, we used an infection rate $\beta = 0.099$, which was calculated using empirical data regarding COVID-19's spread (4). We evaluated an estimate for β as $R_0(\tau)^{-1}$ where τ was the average recovery time.

We retrieved values for three different centrality measures on our network: closeness centrality, betweenness centrality, and PageRank. These values were then sorted, and we selected the nodes with the minimum, median, and maximum centrality values to be seed nodes (patient zero) for our simulations.

Every simulation was run for 15 iterations, and results were averaged.

Parameter scanning and cost-benefit function.

Our cost-benefit function aims to calculate a rough estimate for economic value contributed by individuals. Our goal is to utilize this cost function to gauge optimal quarantine rates – the quarantine rate that yields that greatest benefit is assumed to be a candidate for the optimal quarantine rate to sustain as much economic activity as possible, while minimizing death and infection.

Using economic metrics like GDP seemed like a natural way to assess economic cost-benefit relationships.

For each individual, cost is calculated as follows:

$$\textit{Cost-Benefit}(node) = \begin{cases} \text{GDP per capita per day} & \text{Asymptomatic and not quarantined} \\ 0.5 * \text{GDP per capita per day} & \text{Asymptomatic and quarantined} \\ \text{Average cost of being infected} & \text{Symptomatic} \\ \text{Cost associated with death} & \text{Deceased} \end{cases}$$

We sum over these values for all individuals to calculate the total value of the network.

We calculated the quantities used above as follows:

- GDP Per Capita per day was calculated by dividing the USA GDP Per Capita (5) by 365 days.
- For quarantined, unsymptomatic individuals, we assume a 50 percent decrease in productivity, given limited telecommuting capacity.
- Average cost of being infected was an average of median in-hospital and at-home costs of being symptomatic (6); we averaged over the rate of hospitalization (7).
- Cost associated with death was calculated using a 10 million dollar one-time cost (8), but scaling it by average life expectancy (9) and dividing by 365 days.

Plugging in our known values yields the following cost function:

$$\textit{Cost-Benefit}(node) = \begin{cases} 172.2929027 & \text{Asymptomatic and not quarantined} \\ 86.14630137 & \text{Asymptomatic and quarantined} \\ 4516.73 & \text{Symptomatic} \\ 348.5656523 & \text{Deceased} \end{cases}$$

Algorithm 2 Daily cost function

At the end of each day, we run the cost function loop and add up the outputs and costs for each node depending on their current condition;

for *each node in the network* **do**
  if *this node is not quarantined, not symptomatic, and alive* **then**
    add the daily average per capita output to the total
  end
  if *this node is quarantined, not symptomatic, and alive* **then**
    add half of the daily average per capita output to the total
  end
  if *this node is symptomatic, and alive* **then**
    add daily average cost of sickness to the total
  end
  if *this node is dead* **then**
    add daily average cost of death to the total
  end
end

Results

Seed selection based on Centrality. We observed total infected populations and deaths starting at seed nodes of differing centrality in order to compare infection outcomes. These simulations were conducted on various quarantine rates and averaged over 15 realizations. In our school network (1), there are 236 total nodes.

For non-trivial quarantine rates, we observe that starting the infection with a seed (patient zero) of higher centrality will lead to a greater number of total infections as well as a greater number of deaths.

Table 1. Comparison of infected populations and deaths based on closeness centrality

$N = 236$	Total Deaths			Total Infections		
Quarantine Rate ψ	Minimum Closeness	Median Closeness	Max Closeness	Minimum Closeness	Median Closeness	Max Closeness
0.0	97.5333	97.2	97.3333	236	236	236
0.25	26.8	38.4667	50.1333	53.8	106.3333	118.7333
0.5	3.8667	6.9333	16.6667	8.9333	21.1333	33.8667
0.75	1.8667	3.7333	5.6	3.4667	7.7333	14.4

Table 2. Comparison of infected populations and deaths based on betweenness centrality

$N = 236$	Total Deaths			Total Infections		
Quarantine Rate ψ	Minimum Bet.	Median Bet.	Max Bet.	Minimum Bet.	Median Bet.	Max Bet.
0.0	98.4	96.7333	100.4667	236	236	236
0.25	23.8	33.4667	49.6667	53.8	106.3333	118.73333
0.5	3.3333	8.8	18.2	32.2665	21.0667	8.0667
0.75	2	4.2	7	14.4667	6.9333	4

Table 3. Comparison of infected populations and deaths based on PageRank (PR) centrality

$N = 236$	Total Deaths			Total Infections		
Quarantine Rate ψ	Minimum PageRank	Median PageRank	Max PageRank	Minimum PageRank	Median PageRank	Max PageRank
0.0	100.1333	98.2667	98.8	236	236	236
0.25	17.1333	38.1333	50.2	59.8667	85.8	123.7333
0.5	2.6	7.6	17.8	6.8	20.4667	38.9333
0.75	1.4667	3.1333	5.5333	3.6	8.4	13.8667

We can also proceed to observe trajectories over time of the compartments we considered in our model.

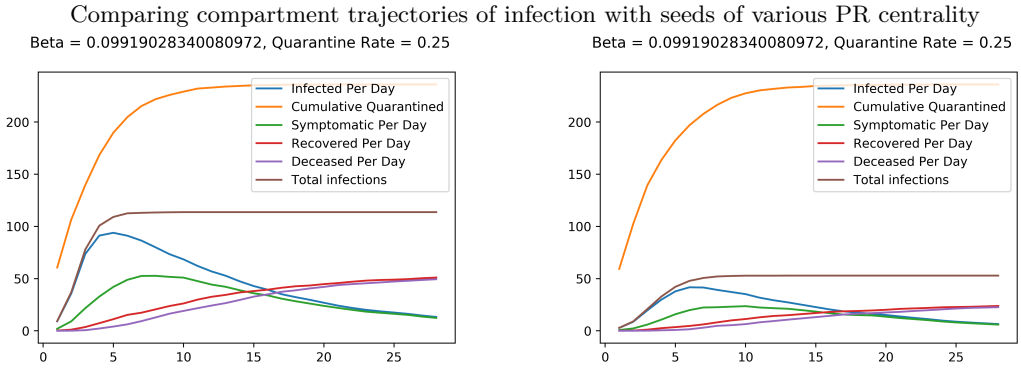


Fig. 3. The graphs above depict compartmental trajectories of infection spread for the maximum (left) and minimum (right) PageRank centrality seed nodes with a quarantine rate of 0.25. We see a significantly larger spike in number of infections and deaths in the maximum centrality infection compared to the minimum centrality infection.

Optimal parameters for Cost Function. We find from our results that increasing quarantine rates offer a significant reduction in infection and death counts. However, we would like to examine the choice of the quarantine rate that optimizes our cost function. We evaluated our cost-benefit function at quarantine rates from 0 to 1 with intervals of 0.05, and averaged our results over 50 trials.

Table 4. Cost-benefit output of various quarantine rates

Quarantine Rate ψ	0.05	0.1	0.15	0.2	0.25	0.3	0.35
Output	-6709815.962	-5719719.92	-4703912.194	-3540081.447	-2559433.285	-1877078.704	-1338843.891
Quarantine Rate ψ	0.4	0.45	0.5	0.55	0.6	0.65	0.7
Output	-887940.771	-433736.49	-121511.561	-48042.89	53252.315	178383.785	241997.6
Quarantine Rate ψ	0.75	0.8	0.85	0.9	0.95	1	
Output	267996.303	295915.583	320452.446	338100.827	327153.516	321470.363	

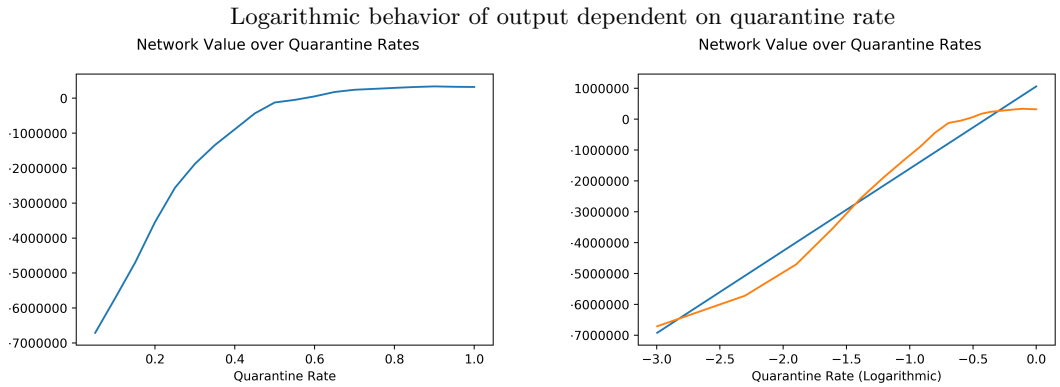


Fig. 4. Upon plotting log(quarantine rate) vs. the observed network value, we see a roughly linear curve, suggesting that network value varies logarithmically on the quarantine rate. This suggests that for lower quarantine rates, the curve is far steeper, which may suggest diminishing returns on increasing quarantine rate.

The data shows that our cost-benefit function results in a net positive output for quarantine rate $\psi > 0.55$, and peaks at $\psi = 0.90$. Under such circumstances, on average, the number of infections totaled 4.68, with 2.24 recoveries, and 2.16 deaths - 231.32 remained susceptible by the end of the 28 day period of the simulation.

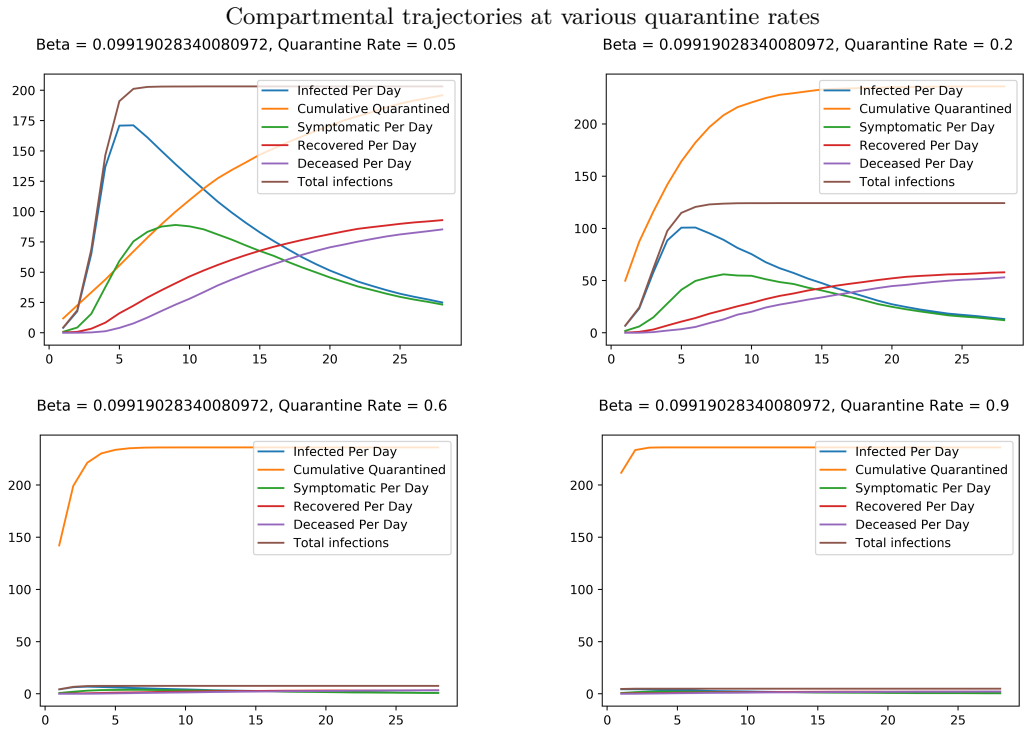


Fig. 5. These visualizations give a general idea on how the infection curve flatten with even a small increase in the quarantine rate. The compartments at a quarantine rate of 0.6 and 0.9 suggest that a majority of individuals remain susceptible, and a very small number (on average) are infected, for an observed positive (beneficial) output. This is a natural result as a majority of the population being quarantined removes the likelihood of potential infectious interactions.

Discussion

Centrality of patient zero influences infection spread.

In cases with non-trivial quarantine rates, we found that if the disease progression begins with a node of higher centrality, the disease will have a higher toll in terms of deaths and total infections. This aligns with theories from current research describing "super-spreaders", or individuals who have greater contact with others, and their increased propensity to pass the disease on (10).

It is important to note that for the cases with no quarantine ($\psi = 0$), the impact of the different centralities of patient zeroes is not as pronounced. Given the relatively large timescale of this configuration on a network of this size, the distinction between infections with low and high centrality seed nodes is obscured by the rapid and unimpeded spread of the virus. If we were to look at a smaller timescale, the significance of the different centralities would once again be more visible.

Understanding node centralities, or colloquially, the importance of an individual in a social network, can be a predictor that can give an early glimpse into how an infection may progress throughout a population.

Additionally, observing that with no quarantine deaths and infections are at their highest points suggests that quarantining to some degree is important. We explore this phenomenon in our parameter scanning analysis.

Optimal parameters for Cost Function. In our trials comparing network cost-benefit value for different quarantine rates, we found that the rate with the greatest output is $\psi = 0.90$. Naturally, increasing the quarantine rate as much as possible, while retaining some level of positive output will be ideal. In rough terms - putting aside the morality of the situation, with the existing medical costs of the infection and the current infection rate, people are worth more alive with limited productivity, than risking their lives for higher output. In fact, regardless of the productivity penalty applied to quarantine, short of net negative, the relative performance under the different quarantine rates remained the same. Therefore showing that preventative measures, not limited to quarantine, are simply more cost effective - and also that adopting such measures in a stricter manner could be the essential first step towards easing lockdown.

However, an important observation comes from the increasing behavior of the cost-benefit output as the quarantine rate rises. The output grows as the log of the quarantine rate, suggesting that as the quarantine rate grows, the growth of the output slows down. These diminishing returns suggest that the practically optimal quarantine rate may sit below our observed optimal one. Our cost function was very coarse-grain, which makes it possible that higher order effects are missed.

It is also important to note that we observe net positive output starting at the rate $\psi = 0.60$, and while numerically the optimal rate is indeed $\psi = 0.90$, the cost-benefit function does not account for broader societal implications of prolonged lockdown, and their eventual impact on the economic output. Therefore, rather than simply taking the $\psi = 0.90$ as a baseline measure, while intuitive, would be a mistake. It is important to consider the whole interval $0.60 < \psi < 0.90$ with supplementary analysis about long-term economic and societal effects of quarantine in decision making.

Summary

Our study suggested that node centrality of patient zero can be used to predict the gravity of infection spread; as the centrality of the infected patient increases, so does the potential to infect more individuals, and we see a corresponding increase in cases and deaths. Additionally, we found that even a small increase in quarantine rate resulted in significant flattening of the infection and death curves. Our cost-benefit analysis of quarantine rates encourage a quantitative method for assessing public health decisions – we see a threshold quarantine rate of 0.60, which we believe to be a minimum quarantine rate for a productive society. Naturally, a trivial quarantine rate of 1 is ideal in protecting everyone, but our study aimed to strike a balance.

ACKNOWLEDGMENTS. We would like to acknowlege Dr. Heather Zinn-Brooks for her guidance throughout this project as well as Dr. Leif Zinn-Bjorkman for teaching some of the authors about ordinary differential equations.

1. J Stehlé, et al., High-resolution measurements of face-to-face contact patterns in a primary school. *PLOS ONE* **6**, e23176 (2011).
2. SA Lauer, et al., The incubation period of coronavirus disease 2019 (COVID-19) from publicly reported confirmed cases: Estimation and application. **172**, 577–582 (year?) Publisher: American College of Physicians.
3. R Verity, et al., Estimates of the severity of coronavirus disease 2019: a model-based analysis. **0** (year?) Publisher: Elsevier.
4. S Sanche, et al., Early release - high contagiousness and rapid spread of severe acute respiratory syndrome coronavirus 2 - volume 26, number 7—july 2020 - emerging infectious diseases journal - CDC. (year?).
5. GDP per capita (current US$) - united states | data (year?).
6. SM Bartsch, et al., The potential health care costs and resource use associated with COVID-19 in the united states: A simulation estimate of the direct medical costs and health care resource use associated with COVID-19 infections in the united states. , 10.1377/hlthaff (year?).
7. Coronavirus (COVID-19) facts los angeles | valley presbyterian hospital (year?) Library Catalog: www.valleypres.org.
8. WK Viscusi, *Pricing Lives: Guideposts for a Safer Society*. (Princeton University Press), (2018).
9. FastStats (year?) Library Catalog: www.cdc.gov.
10. A Gómez-Carballa, X Bello, J Pardo-Seco, F Martinón-Torres, A Salas, The impact of super-spreaders in COVID-19: mapping genome variation worldwide. , 2020.05.19.097410 (year?) Publisher: Cold Spring Harbor Laboratory Section: New Results.

Supplementary Materials - Code

Math 168 Final Project

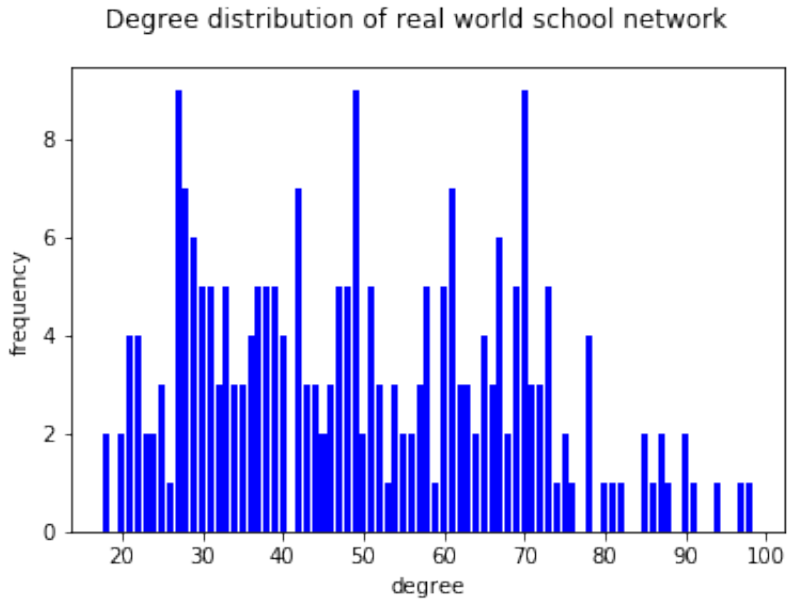
Aditya Pimplaskar, Daniel Koo, Aleksandre Ninua

In [4]:
```
# Packages

import networkx as nx
import numpy as np
import numpy.random as rand
import random
from collections import deque
import matplotlib.pyplot as plt
import math
import scipy.stats as stats
import operator
import collections
```

In [5]:
```
# Data cleaning and retrieving network features

def cleanGraph(Gr):
    # takes gexf file
    # output: better indexed network, dict of node labels to easier in
dices
    inds = {}
    for i in list(Gr.nodes):
        inds[i] = Gr.nodes[i]
    Gr = nx.convert_node_labels_to_integers(Gr)
    return Gr, inds

day1 = nx.read_gexf("data/sp_data_school_day_1_g.gexf_")
G, inds = cleanGraph(day1)
#G = nx.gnp_random_graph(600,0.1)
pagerank = nx.pagerank(G)
bet = nx.betweenness_centrality(G)
close = nx.closeness_centrality(G)

def getMaxMinMid(centralities):
    # returns node label for max, min, mid
    import operator
    length = len(centralities)
    nodelist = list(sorted(centralities.items(), key = operator.itemge
```

```
tter(1)))

    mmax = nodelist[length-1][0]
    mmin = nodelist[0][0]
    mmid = nodelist[int(length/2)][0]

    return mmax, mmin, mmid

maxpr, minpr, midpr = getMaxMinMid(pagerank)
maxbet, minbet, midbet = getMaxMinMid(bet)
maxclose, minclose, midclose = getMaxMinMid(close)

print("Pagerank max:", maxpr, "| Pagerank min:", minpr, "| Pagerank mi
d:", midpr)
print("Betweenness max:", maxbet, "| Betweenness min:", minbet, "| Bet
weenness mid:", midbet)
print("Closeness max:", maxclose, "| Closeness min:", minclose, "| Clo
seness mid:", midclose)

degrees = sorted([x for n, x in G.degree()], reverse = True)
degreeCount = collections.Counter(degrees)
deg, cnt = zip(*degreeCount.items())

fig, ax = plt.subplots()
plt.bar(deg, cnt, width=0.80, color='b')
fig.suptitle("Degree distribution of real world school network")
plt.xlabel("degree")
plt.ylabel("frequency")
plt.savefig("DegreeDist.png", dpi = 500)

neighbor_degs = nx.average_neighbor_degree(G).values()

print("radius: %f" % nx.radius(G))
print("diameter: %f" % nx.diameter(G))
print("average shortest path: %f" % nx.average_shortest_path_length(G)
)
print("average degree: %f" % np.mean(degrees))
print("average neighbor's degree: %f" % np.mean(list(neighbor_degs)))
```

```
Pagerank max: 23 | Pagerank min: 136 | Pagerank mid: 223
Betweenness max: 23 | Betweenness min: 7 | Betweenness mid: 183
Closeness max: 23 | Closeness min: 179 | Closeness mid: 186
radius: 2.000000
diameter: 3.000000
average shortest path: 1.860440
average degree: 49.991525
average neighbor's degree: 55.693933
```



In [7]:
```
# Simulations

def BFS_t(Gr,zero,p,h,d,s,x,r):

        #Prameters:
        #Gr - Graph
        #zero - patient zero
        #p - probability of transmitting infection
            #by a symptomatic host at every itneraction
        #s - probability of developing symptoms once infected
        #h - probability of quarantining once symptomatic
        #r - probability of recovering
        #x - probability of death
        #d - number of days simulation is run
        if d%2 == 0:
           nrows = int(d/2)
        else:
           nrows = int(d/2)+1
        ncols = 2
        #f, axes = plt.subplots(nrows, ncols, figsize = (40,40))

        #Status arrays
```

```
        infected = [False] * Gr.number_of_nodes()
        symptomatic = [False] * Gr.number_of_nodes()
        quarantined = [False] * Gr.number_of_nodes()
        recovered = [False] * Gr.number_of_nodes()
        deceased = [False] * Gr.number_of_nodes()
        infected_days = [0] * Gr.number_of_nodes()

        #Metrics
        inf = 1
        rec = 0
        dead = 0
        days_rem = d

        #Probability arrays
        death_rate = [x] * Gr.number_of_nodes()
        recovery_rate = [r] * Gr.number_of_nodes()
        symptom_rate = [s] * Gr.number_of_nodes()

        if x < 0:
            for i in range(0,Gr.number_of_nodes()):
                death_rate[i] = 1/np.random.gamma(4.94, 1/.26)
        if r < 0:
            for i in range(0,Gr.number_of_nodes()):
                recovery_rate[i] = 1/np.random.gamma(8.16, 1/.33)
        if s < 0:
            for i in range(0,Gr.number_of_nodes()):
                symptom_rate[i] = 1/np.random.gamma(5.81, 1/0.95)

        #Output array
        GDP_per_capita = 62886.8
        GDP_daily_per_capita = GDP_per_capita / 365
        life_expectancy = 78.6
        hospital_rate = 0.13
        hospital_cost = 14366
        symptom_cost = 3045
        infected_cost = hospital_cost*hospital_rate + symptom_cost*(1-
hospital_rate)
        death_cost = 10000000 / (life_expectancy * 365)


        total_output = 0

        #Result arrays
        queue = []
        infected_nodes = []
        symptomatic_nodes = []
        quarantined_nodes = []
        recovered_nodes = []
        deceased_nodes = []
```

```
        # element at position i is the number of infected people on da
y i
        num_infected_per_day    = []
        num_symptomatic_per_day = []
        num_quarantined_per_day = []
        num_recovered_per_day   = []
        num_deceased_per_day    = []
        num_total_infected    = []
        num_sus = []

        queue.append(zero)
        infected[zero] = True
        infected_nodes.append(zero)
        while days_rem > 0:
            days_rem-=1

            while queue:
                s = queue.pop(0)
                for i in Gr.neighbors(s):
                    if infected[i] == False and recovered[i] == False
and deceased[i] == False and quarantined[i] == False:
                        if rand.uniform(0,10) < p*10:
                            infected[i] = True
                            infected_nodes.append(i)
                            inf+=1

            for i in range(0,len(infected)):
                if quarantined[i] == False:
                    rand_num = rand.uniform(0,10)
                    if symptomatic[i] == False:
                        if rand_num < h*10:
                            quarantined[i] = True
                            quarantined_nodes.append(i)
                    else:
                        if rand_num/2 < h*10:
                            quarantined[i] = True
                            quarantined_nodes.append(i)
                if infected[i] == True:
                    infected_days[i]+=1
                    if symptomatic[i] == False:
                            if rand.uniform(0,10) < symptom_rate[i]*10
:
                                symptomatic[i] = True
                                symptomatic_nodes.append(i)
                            elif rand.uniform(0,10) < recovery_rate[i]
*10:
                                    recovered[i] = True
                                    rec+=1
                                    inf-=1
                                    recovered_nodes.append(i)
```

```
                                        infected[i] = False
                                        symptomatic[i] = False
                                        infected_nodes.remove(i)
                    else:
                            if rand.uniform(0,10) < death_rate[i]*10:
                                        deceased[i] = True
                                        dead+=1
                                        inf-=1
                                        deceased_nodes.append(i)
                                        infected[i] = False
                                        symptomatic[i] = False
                                        symptomatic_nodes.remove(i)
                                        infected_nodes.remove(i)
                            elif rand.uniform(0,10) < recovery_rate[i]
*10:
                                        recovered[i] = True
                                        rec+=1
                                        inf-=1
                                        recovered_nodes.append(i)
                                        infected[i] = False
                                        symptomatic[i] = False
                                        symptomatic_nodes.remove(i)
                                        infected_nodes.remove(i)


                    if quarantined[i] == False and recovered[i] == Fal
se and deceased[i] == False:
                            queue.append(i)
            for i in range(0,Gr.number_of_nodes()):
                if quarantined[i] == False and deceased[i] == False an
d symptomatic[i] == False:
                    total_output+=GDP_daily_per_capita
                elif quarantined[i] == True and deceased[i] == False a
nd symptomatic[i] == False:
                    total_output+=0.5*GDP_daily_per_capita
                elif symptomatic[i] == True and deceased[i] == False:
                    total_output-= infected_cost
                elif deceased[i] == True:
                    total_output-= death_cost
            # Update per-day numbers
            num_infected_per_day.append(inf)
            num_symptomatic_per_day.append(len(symptomatic_nodes))
            num_quarantined_per_day.append(len(quarantined_nodes))
            num_recovered_per_day.append(rec)
            num_deceased_per_day.append(dead)
            num_total_infected.append(inf+rec+dead)
            num_sus.append(Gr.number_of_nodes()-inf-rec-dead)


            colvec = [0]* Gr.number_of_nodes()
```

```
            for i in range(Gr.number_of_nodes()):
                if quarantined[i] == False and infected[i] == False an
d deceased[i] == False:
                    colvec[i] = "g"
                if quarantined[i]:
                    colvec[i] = 'b'
                if deceased[i]:
                    colvec[i] = 'r'
                if infected[i]:
                    colvec[i] = 'y'
                if symptomatic[i]:
                    colvec[i] = 'm'
                if recovered[i]:
                    colvec[i] = 'c'
                ColorLegend = {"Recovered": "c", "Asymptomatic":"y", "
Symptomatic":"m", "Deceased":"r", "Quarantined and Healthy":"b", "Heal
thy": "g"}

            if days_rem == 0 or days_rem == 14 or days_rem == 27:
                fig = plt.figure(figsize = (10,10))
                fig.suptitle("Network-wide infection spread at the end
of day " + str(d - days_rem))
                #n = nx.draw_networkx(Gr, pos=nx.kamada_kawai_layout(G
r), node_color=colvec, cmap=plt.cm.rainbow) #visualizes
                layout = nx.kamada_kawai_layout(Gr)
                ax = fig.add_subplot(1,1,1)
                for label in ColorLegend:
                    ax.plot([0],[0],color=ColorLegend[label],label=lab
el)
                nx.draw_networkx_nodes(Gr, pos = layout, node_color =
colvec)
                nx.draw_networkx_edges(Gr, pos = layout, alpha = 0.6)
                plt.axis('off')
                sm = plt.cm.ScalarMappable(cmap=plt.cm.rainbow, norm =
None)
                sm.set_array([])
                plt.legend()
                #cbar = plt.colorbar(sm)
                plt.savefig("Network-wide infection spread at the end
of day " + str(d - days_rem), dpi = 500)

        return [infected_nodes,quarantined_nodes,symptomatic_nodes,rec
overed_nodes,deceased_nodes, num_infected_per_day, num_quarantined_per
_day, num_symptomatic_per_day, num_recovered_per_day, num_deceased_per
_day,num_total_infected,num_sus,total_output]

def multi_BFS_t(Gr, zero, beta, qrnt, days, s_rate, x_rate, r_rate, n,
prefix):
    avg_res_per_day = [[0] * days] * 7
```

```
    for i in range(n):
        res = BFS_t(Gr, zero, beta, qrnt, days, s_rate, x_rate, r_rate
)[5:12]
        for j in range(len(res)):
            avg_res_per_day[j] = [x + y for x, y in zip(avg_res_per_da
y[j], res[j])]
            # print(res[j])
            # for k in range(days):
            #     avg_res_per_day[j][k] += res[j][k]

    for l in range(7):
        for m in range(days):
            if avg_res_per_day[l][m] != 0:
                avg_res_per_day[l][m] /= n
    plot_numbers_per_day(avg_res_per_day, beta, qrnt, days, prefix)
    return avg_res_per_day
```

In [8]:
```
# Visualizations

def plot_numbers_per_day(res, beta, qrnt, days, prefix):
    days_axis = [i for i in range(1, days+1)]
    labels = ["Infected Per Day", "Cumulative Quarantined", "Symptomat
ic Per Day", "Recovered Per Day", "Deceased Per Day","Total infections
","Susceptible"]
    fig = plt.figure()

    fig.suptitle("Beta = " + str(beta) + ", Quarantine Rate = " + str(
qrnt), fontsize=12)
    for p in range(len(res)-1):
        ax = fig.add_subplot(111)
        ax.plot(days_axis, res[p], label=labels[p])
        ax.legend(loc="upper right")
        filename = "figure "+ prefix+" " + str(beta) + " " + str(qrnt)
+".png"
        plt.savefig(filename, dpi = 500)
```