(Final Report) EECS4414 Information Networks: Deriving & Analysing Networks from COVID-19 Simulations

Milan Drapic
drapic23@my.yorku.ca
York University
Toronto, Ontario, Canada

Vishwa Perera
vishwape@my.yorku.ca
York University
Toronto, Ontario, Canada

Lubeed Rana
lubeed10@my.yorku.ca
York University
Toronto, Ontario, Canada

ABSTRACT

COVID-19 has been a common issue since March 2020 and has impacted our life immensely. As a result, a thorough examination of COVID-19's behaviour is essential for developing preventive techniques to limit its rapid spread. COVID-19 is a relatively new virus, therefore research is still ongoing to better understand it, and constructing a reliable COVID-19 model is a challenge.

Our goal is to create an accurate simulation that accurately reflects COVID-19's propagation while also properly simulating preventative actions. To depict the simulations, we employed stochastic cellular automata and stochastic agent-based mobility models.

The models, algorithms, and results, and analysis of our experiments are discussed in depth in this paper.

KEYWORDS

COVID-19, simulation, network models, graphs, virus, agent-based modeling, cellular automata, proximity networks

1 INTRODUCTION

1.1 Motivation

COVID-19 has had an extensive impact on every part of the globe since March 2020. This pandemic has lead to over 400 million infections, and over 5 million deaths worldwide. COVID-19 restrictions have lead to many business closures, and has led to an immense increase in unemployment. In April 2020 more than 100,000 businesses closed, leading to 30% of job losses that month in Canada[4].

The challenge we will look to solve is to find a way to accurately simulate the propagation of COVID-19 within a virtual society. With this model we can examine a network to understand the virus more deeply, and the nature of its spread. We can also deduce what restrictions, and precautions seem to be most effective in reducing the spread of the virus, and minimizing its impact. If we can create a model that accurately simulates how the virus has spread until now, then it can also help us predict how the virus will continue to propagate in the future.

2 PROBLEM DEFINITION

2.1 General

Problem 1:
Simulate different levels of mobility within a population and analyze the different levels of virus propagation.
Problem 2:
Implement a network $G(N, E)$ that shows contacts between nodes and simulates viral propagation within the contact graph.
Problem 3:
Implement preventative measures in each simulation to contain the spread of the virus.

Definition 2.1 (SIR Model)**.** The SIR model is a model in which each node $n \in N$ can have one of three different states at any spatiotemporal point of the model. These three states are: susceptible, infectious, recovered/removed. These states can only occur in the order listed above (i.e a recovered node will never reach the susceptible state again).

Definition 2.2 (Basic Reproductive Number R_0)**.** The R_0 value represents the number of secondary infections node $n \in N_{infectious}$ is expected to produce.

Definition 2.3 (Infectious Area)**.** An infectious area of a node $n \in N$ is the area around n where if n was in the infectious state, then n would be able to emit the virus into this surrounding area.

Definition 2.4 (Susceptible Area)**.** A susceptible area of a node $n \in N$ is the area around n where if n was in the susceptible state, and the virus came within the susceptible area of n, then n would be at a risk of infection.

Definition 2.5 (Contact)**.** A contact between two nodes occurs when the infectious area of one node $n_i \in N$ overlaps with the susceptible area of another node $n_s \in N$ at some time-frame $t \in T$.

Definition 2.6 (Contact Network)**.** Is a network $G(N, E)$ where the weight w of each edge $(n_1, n_2) \in E$ represents the number of time-frames nodes $n_1 and n_2$ spent in contact with one another during some time period [0,T].

Definition 2.7 (Propagation Network)**.** Consider some Contact Network $G(N_c, E_c)$. A propagation network $P(N, E)$, where $P \subseteq G$, is a directed graph that considers what contacts between nodes led to the propagation of the virus from one node to another.

2.2 Cellular Automata

Problem 1:
Create a square 2-D undirected grid network where each cell represents a person. This is a network $G(N, E)$ where N is the set of nodes and where E is set of edges. $e_i \in E$ is an edge that connects two neighbouring nodes.
Problem 2:
Create a stochastic infection propagation algorithm and record the infected edges per time step. The recorded edges will be used to create a separate directed network $D(N, E)$ where N is the set of nodes and where E is set of edges. $(n_i, n_j) \in E$ is a pair such that n_i (the source node) infects n_j (the target node). This network will used for analysis and deriving cascades. In addition to this, we also keep track of the number of nodes in each state (SIR) per item step.

2.3 Mobility Model

Problem 1:
Create N different individual agents with unique risk attributes.
Problem 2:
For each agent $n \in N$ determine a trajectory $\tau = \{(x_0,y_0), (x_1,y_1), (x_2,y_2),\dots , (x_t,y_t)\}$ where $x_i \in [0, X]$, $y_i \in [0, Y]$ and represent coordinates in an $X \times Y$ sized 2-D cartesian plane at each time frame.
Problem 3:
Given each trajectory τ_n from each node $n \in N$, construct a contact network $T(N, C)$, where N is the list of individual agent nodes in the model and C is a sequence of pairs (n_i, n_j) where n_i, $n_j \in N$ and represent a pair of nodes that have come in contact with one another during their trajectories τ_i, τ_j.
Problem 4:
Given contacts $c_{ij} \in C$, and the state of each node at each spatiotemporal point in C. Determine a probabilistic way of representing infection between nodes $n_i, n_j \in N$ in the model.
Problem 5:
Create a cascading/propagation Network $P(N_{infected}, I)$, where $N_{infected} \subseteq N$, as $N_{infected}$ represents all the nodes in N which were at one point in the infectious state, and I represents each pair of nodes $n_i, n_j \in N_{infected}$ where n_i infected n_j at one point in the model.
Problem 6:
Visually represent the trajectories of nodes in a 2-D simulation.
Problem 7:
Determine a way to divide all N nodes into R private communities, and P public communities.

3 RELATED WORK

3.1 Agent-Based Modeling

Agent-based modeling (ABM) uses data structures to represent each individual within the community and allows each of them to move around and interact with each other based on a set of rules. This model lets us define the transmission of the virus on an individual level.

In the study of Wang et al.[6], they simulated two different R_0 values in a community with low density. Interestingly, each variation produced different results. A R_0 value of 2.7 infected over half the population, while a R_0 of 1.6 infected just over one seventh.

3.2 Cellular Automata

Two dimensional cellular automata (CA) [3] have the ability to simulate complex systems with a simplistic design. First we create a grid with uniformly arranged cells where time and space are discrete and interactions occur locally. Each cell has a state and a set of rules.

For our research we only consider probabilistic stationary CA to create a first basic model. Under these conditions we can create an environment where neighbouring cells are used to derive the probability that an individual cell is infected on each time step. The model we would like to propose will allow the infection to spread to eight compass directions and each cell can take on one of the following states: susceptible, infected, or removed (SIR)[1]. With this model the CA is limited to two dimensional visualizations and a static graph structure.

3.3 Proximity Networks

Proximity networks [2] were derived from epidemic simulations using high-quantity human mobility data. These networks attempt to accurately model the distance (proximity) of human interactions within a simulated environment at a specific time step/snapshot. In our research we can create a simplified version where susceptible individuals have a probability of being infected within a certain radius of an infected individual.

3.4 Trajectory Networks

Trajectory networks are networks that are similar to proximity networks, in that they analyze the distances between nodes, and create relationships between nodes based on their proximities. However trajectory networks incorporate the temporal dimension, by essentially merging a sequence of proximity networks into a single network[5]. In this project we build off the work of Pechlivanoglou et al [5] as they have executed a similar analysis of trajectory networks in epidemiology.

4 METHODOLOGY

4.1 Stochastic Agent-Based Cellular Automata

The simulation is displayed on a 40×40 2-D grid, where row zero and column zero is the top-left corner. This creates 1600 susceptible nodes. Each susceptible node has some susceptibility value (δ), which is assigned stochastically. Initially 1% of the nodes are stochastically infected. By this we mean that each node had a 1% of chance being infected. The visualization criteria are as follows: nodes are blue if they are susceptible, red if they are infected, and black if they have been removed. Edges should follow a similar rule: the edge colour should match the node colour, or if the endpoints are a different colour, the edge colour should remain the same as the prior iteration.

4.1.1 Propagation. Within the cellular automata, propagation is accomplished by looping through each of the susceptible nodes and inspecting its neighbours. The chance of infection (β) is set to 10%. The probability an infected node U infects susceptible node V is: $p_{U->V} = \delta_V \cdot \beta$. The node, on the other hand, will be recovered/removed if it has already been infected for 14 time steps.

This propagation algorithm is repeated for 121 time steps (time step 0 to 120 inclusively). This algorithm converges since each infected node will be put into the removed state after 14 iterations.

4.2 Stochastic Agent-Based Mobility Model

In this model, we represent the simulation on a 1200×900 pixel 2-D cartesian plane, where the value (0,0) is represented in the top-left corner of the plane. This means that an increase in y, is represented by movement down the plane, and an increase in x, is represented by movement to the right of the plane. The model contains 99 initial nodes. Nodes can take one of three states in the SIR model (Defintion 2.1). Susceptible nodes will be colored blue, infectious

nodes will be colored red, and recovered nodes will be colored black in the simulation. One node will be initially set to the infectious state. To create the model, we used Python3.8 with the Pygame library to run our simulation, Matplotlib to visualize our networks along with the data associated with our networks, and finally we use NetworkX to store our networks, and to run operations on them. We run the simulation at 100 Frames Per Second (FPS). We initialize 24 private communities and 4 public communities. Each private community has an area of 100×100, and each public community has an area of 200×200. An example of what the simulation looks like is in Figure 1.

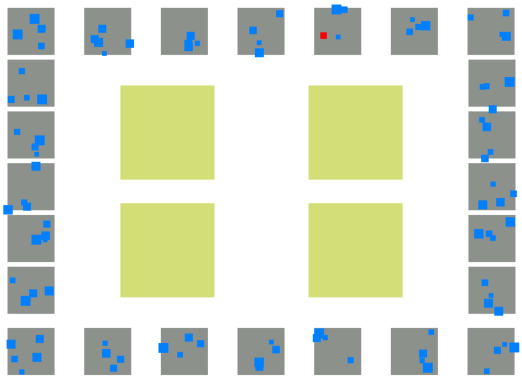

Figure 1: Simulation at t=100

4.2.1 Agent Creation. In the stochastic agent-based mobility model, we create nodes with attributes assigned randomly depending on their risk level. Initially we have three different risk levels: low, medium, and high.

Risk	Width of (S/I) area	Immunity	Recovery
Low	[10-13]	[0.97-1.00)	[900-1100]
Medium	[14-17]	[0.95-0.97)	[1050-1350]
High	[18-21]	[0.93-0.95)	[1100-1700]

Figure 2: The range of values for each node given a different risk level. Shows the range for: Width of susceptible (S)/infectious(I) area (*pixels*), Immunity on Contact (*percentage*), Recovery Time (*time-frames*)

Figure 2 shows the range of values for each node given a different risk level. The width of the susceptible/infectious area is the width of a square node on the cartesian area. The immunity represents the probability of fighting off infection while in the susceptible state, after contact with an infectious node. The recovery time represents the number of time-frames it will take for a node to recover from an infection. Each node is then assigned its own private, and public community. For each node $n \in N$ assign value $n.home = (x_h, y_h)$ which represents the upper-left coordinates of its private community, and assign $n.public = (x_p, y_p)$ which represents the upper-left coordinates of its public community.

4.2.2 Stochastic Node Mobility. To simulate stochastic node mobility in the simulation within some W^2 area, we would randomly assign a pair of coordinates $(dx_i, dy_i) \in [0, W]$, where dx_i represents the x value for the destination in node i, and dy_i represents the y value for the destination in node i. Each node also stores coordinates $(x_i, y_i) \in [0, W]$ which represent the current position of the center of node $i \in N$ on the 2-D cartesian area. For each time-frame, we check if for each node $i \in N$, $x_i \equiv dx_i$. If false, increment/decrement x_i in the direction of dx_i, otherwise do nothing. Then we check if for each node $i \in N$, $y_i \equiv dy_i$. If false, increment/decrement y_i in the direction of dy_i, otherwise do nothing. If the node reached its destination, then randomly assign a new pair of coordinates and repeat the process.

4.2.3 Stochastic Node Mobility Within Communities. To simulate node mobility within private and public communities, we assign each node, an attribute *isHome* which is initially set to *True*. If *isHome* is True when node $n \in N$ reaches its destination, then we need to decide whether to set its next destination to another point within its private community, or to assign the next destination to some point in its public community. If the node is in its private area, then the probability that its next destination is in its public area is 1% (which would result in the isHome variable being set to False). If the the node was in its public area, and the node reached its destination, then the probability that its next destination is in its private area is 10% (which would result in the isHome variable being set to True). Therefore the nodes are more likely to spend time at their private homes than in the public areas. This algorithm can be seen in Algorithm 1.

4.2.4 Node Movement Based on Trajectories. To be able to accurately compare the effectiveness of preventative measures, we need to find a way to compare the spread of a virus within the same population, given the same mobility. To solve this problem, we ran the simulation once with stochastic node mobility within communities, and sent the output of the trajectories of each node into a file called $'trajectories.json'$. Then at the beginning of each following simulation we retrieve the trajectories of each node, which is a sequence of coordinates $\tau = \{(x_0,y_0), (x_1,y_1), (x_2,y_2),\dots, (x_t,y_t)\}$ where $x_i \in [0, X]$, $y_i \in [0, Y]$ and represent coordinates at each time-frame $t \in T$. Given these trajectories, we update the position of each node at every time-frame.

4.2.5 Contact Network. To address Problem 3 from Section 2.3 we had to construct a contact network $T(N, C)$, where N is the list of individual nodes in the model and C is a sequence of pairs (n_i, n_j) where $n_i, n_j \in N$ and represent a pair of nodes that have come in contact with one another during their trajectories τ_i, τ_j. This contact network T will be a weighted, and undirected graph. The edges are undirected since they represent a **mutual** contact [2.5] between two nodes. Contacts are mutual since we assume that the infectious area of a node is the exact same size as its susceptible area. However, the edges are weighted since the weight of the edge represents the number of time-frames where n_i and n_j were in contact with one another.

To keep track of our contact network T in our simulation, we initialized an undirected graph with the NetworkX library. Then for each time-frame we analyzed the collisions in the simulation.

Algorithm 1 Algorithm for Stochastic Node Mobility Within Communities

procedure moveNodes(N)
▷ N is a list of all nodes in the model.
  for $i \in N$ **do**
    if $i.x \neq i.dx$ **then**
      i.position.right ← i.position.right + $\frac{(i.dx-i.x)}{|i.dx-i.x|}$
      ▷ Note that $\frac{(i.dx-i.x)}{|i.dx-i.x|} \equiv (-1) \vee (+1)$.
    end if
    if $i.y \neq i.dy$ **then**
      i.position.bottom ← i.position.bottom + $\frac{(i.dy-i.y)}{|i.dy-i.y|}$
    end if
    ▷ If node i has reached its destination, and needs new one.
    if $(i.x = i.dx) \wedge (i.y = i.dy)$ **then**
      $z \leftarrow$ random integer $z \in [1, 100]$
      if $i.isHome$ **then**
        if z > 1 **then**
          W ← 100
          destination ← $i.home$
        else
          $i.isHome \leftarrow False$
          W ← 200
          destination ← $i.public$
        end if
      else
        if z > 10 **then**
          W ← 200
          destination ← $i.public$
        else
          $i.isHome \leftarrow True$
          W ← 100
          destination ← $i.home$
        end if
      end if
    ▷ Assign new destination as random values within designated range.
      i.dx ← $x \in [i.destination_x, i.destination_x + W]$
      i.dy ← $y \in [i.destination_y, i.destination_y + W]$
    end if
    $i.\tau.append((i.x, i.y))$
  end for
end procedure

Collisions between nodes represent nodes coming in contact [2.5] with one another.

When the nodes n_i, n_j collide, we add an edge between them if this was their first collision, and set the weight of the edge to 1. If this is not their first collision then increment their weights by 1.

The algorithm for contact network maintenance in our simulation is shown in Algorithms 2 and 3.

4.2.6 Virus Propagation Network. To address Problem 5 from Section [2.3], we had to deduce a way of creating a cascading/propagation

Algorithm 2 Algorithm for Contact Network Maintenance

procedure updateT($T(N, C)$)
  for $n \in N$ **do**
    $collisions \leftarrow pygame.collisions(n, N)$
    if $collisions > 1$ **then**
      for $x \in collisions[2...collisions.length]$ **do**
        $addContactEdge(n, x)$
      end for
    end if
  end for
end procedure

Algorithm 3 Algorithm for adding edge to Contact Network

procedure addContactEdge($n_1, n_2, T(N, C)$)
  if $(n_1, n_2) \exists C$, where C is the Edge List of T (i.e the list of previous contacts) **then**
    $(n_1, n_2).weight \leftarrow (n_1, n_2).weight + 1$
  else
    if $\neg(n_1 \exists N)$, where N is the Node List of T **then**
      $T.addNode(n_1)$
    end if
    if $\neg(n_2 \exists N)$, where N is the Node List of T **then**
      $T.addNode(n_2)$
    end if
    $T.addEdge((n_1, n_2), 1)$
      ▷ adds (n_1, n_2) to C, with weight of 1
  end if
end procedure

Network $P(N_{infected}, I)$, where $N_{infected} \subseteq N$, as $N_{infected}$ represents all the nodes in N which were at one point in the infectious state, and I represents each pair of nodes $n_i, n_j \in N_{infected}$ where n_i infected n_j at one point in the model. To do this we had to find a way to represent a node n_i infecting another node n_j in the simulation [2.3]. We do this by creating a directed, unweighted graph P in our simulation using the NetworkX library. P is directed since infection from one node to another is not mutual, therefore it has a direction. We maintain P in our simulation by analyzing all the contacts (i.e collisions) between each infected node $n_i \in N_{infected}$ with every other node $n \in N$. Then for every combination of n and n_i, we generate a random number p between [0-1]. If this number p is greater than $n.immunity$ and n is in the susceptible state, then n_i infects n and n enters the infectious state. n will then stay infectious for n.recovery_time time-frames, after which it will enter the recovered state until the end of the simulation. The infection from n_i to n will be represented by an edge (n_i, n) which will be added to the edge list I of P. Notice that in the end, the Graph will take on the form of a Directed Acyclic Graph (DAG), or more specifically, it will be a tree because each node can only have one parent (unless it is the initial infection, then it would have no parent) since we can't have multiple nodes infecting the same node in our model. Also we cannot have any cycles since once a node is infectious or recovered it cannot be infected again. The algorithm used in our simulation to maintain our propagation network is shown in Algorithm 4.

Algorithm 4 Algorithm for Propagation Network Maintenance

```
procedure UPDATEP(N_infectious, N, P(N_infected, I))
    for n_i ∈ N_infectious do
        collisions ← pygame.collisions(n_i, N)
        if collisions > 1 then
            for x ∈ collisions[2...collisions.length] do
                if x.susceptible ∧ p > x.immunity then
                    x.susceptible ← False
                    x.infectious ← True
                    P.addNode(x)
                    P.addEdge((n_i, x))
                end if
            end for
        end if
    end for
end procedure
```

4.3 Preventative Measures

Definition 4.1 (Linear Threshold Decision-Based Model)**.** The linear threshold decision-based model is a model which represents how individuals might make some decision D. Each node $n \in N$ has some threshold value $t_n \in [0, 1]$. n also has neighboring nodes $n_i \in K$ which all exert some level of influence on n. If n has $|K_D|$ total neighbors that have complied with decision D, then the total influence on a node n (I_n) is as follows,

$$I_n = \sum_{k=1}^{|K_D|} n.Influence(n_k)$$

If $I_n \geq t_n$ then node n decides to comply with decision D.

4.3.1 Masks. In the mobility model, we implement the idea of masks by assuming masks reduce an individuals susceptible/infectious area by some factor M. We also assume individuals would only wear masks when they are away from their private areas.

4.3.2 Vaccines. We implement the idea of vaccinations in our simulation through the Linear Threshold Model from Definition 4.1. We do this by initially assigning a value t_n for each node $n \in N$ based on a normal distribution where $\mu = 0.4$ and $\sigma = 0.33$.

$$t_n \sim \mathcal{N}(\mu, \sigma^2)\,.$$

What defines the initial vaccinations are early adopters; those who's adoption threshold (t_n) is zero.
For the cellular automata model we assume that that each node n is aware of the vaccination decision of every node $n_i \in K$ where K represents the set of nodes that neighbor n in the grid. While in the mobility model we assume that each node n is aware of the vaccination decision of every node $n_i \in K$ in which K is the set of nodes n shares a private or public community.

In the cellular automata model, the influence a neighboring node $n_i \in K$ has on n is: $\frac{1}{|K|}$.

In the mobility model, the influence a neighboring node $n_i \in K$ has on n, depends on whether node n_i shares a private or public area with n. Assume n has $|P|$ neighbors in which it shares only a public area with, and n has $|R|$ neighbors in which it shares a private area with. If $n_i \in R$ then the influence n_i has on n is: $\frac{1}{|R|}$

If $n_i \in P$ then the influence n_i has on n is: $\frac{1}{|P|}$

When susceptible nodes get vaccinated in our simulations, they are colored yellow. Vaccinations start at some time step $t \in [0, T]$ and are assumed to be 100% effective.

5 EVALUATION

5.1 Cellular Automata

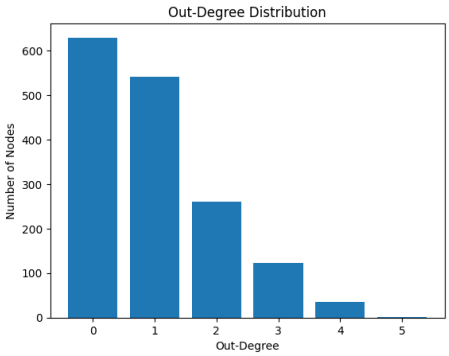

Figure 3: Out-degree distribution for CA

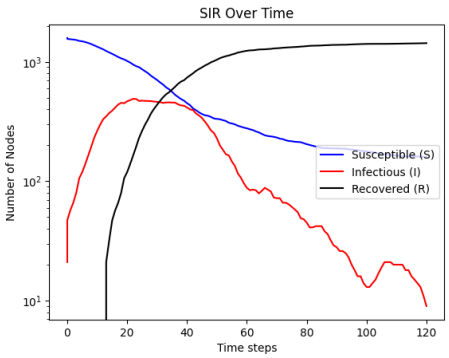

Figure 4: SIR over time for CA

5.1.1 Analysis. Figure 3 shows the out-degree distribution for the CA. As we look at more contagious nodes the number of them decreases.

Figure 4 shows the number of susceptible, infected, and removed nodes at each time step. It demonstrates that when the number of susceptible nodes and removed nodes are equal, the number of infections peaks. However after time step 14 the infectious nodes drop drastically as expected.

5.1.2 Analysis: Vaccination. Without vaccinations, 90% of the population was infected; however, with vaccinations, only 59% was infected. Figure 5 reveals that vaccines results in a significant reduction in the number of cases. This decline occurs significantly faster, and as a result, the virus dies out much quicker. It also demonstrates

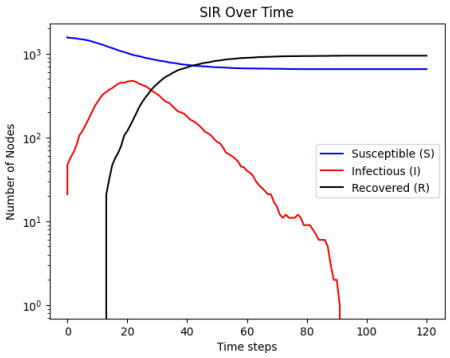

Figure 5: SIR over time with vaccinations

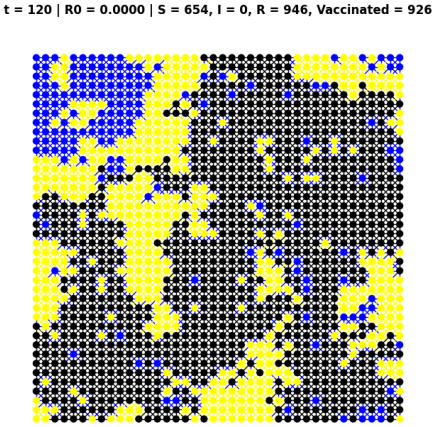

Figure 6: Final iteration of CA with vaccinations. Yellow nodes are vaccinated.

that after a certain point, the number of susceptible nodes stops decreasing greatly. This pertains to when the $R0$ value has dropped below 1.

At the top left of figure 6, it captures a perfect example of herd immunity. The vaccinated nodes stop the propagation of the virus and protect the unvaccinated nodes.

5.2 Agent-Based Mobility Model

5.2.1 Simulation. In this simulation we create 99 nodes, 33 of each risk level, with the associated properties from Figure 2. In the simulation we use the SIR model defined in Definition 2.1. We set 1 of the nodes with medium risk to be initially infectious, then we run the simulation until $|N_{infectious}| \equiv 0$. We run the simulation on a 1200×900 pixel 2-D cartesian area. Our simulation runs at 100FPS and finishes after running 7421 time-frames.

5.2.2 Analysis. At the end of the simulation, 28/33 high risk nodes were infected, 28/33 medium risk nodes were infected, and 15/33 low risk nodes were infected, resulting in a 72% infection rate by the end of the simulation.

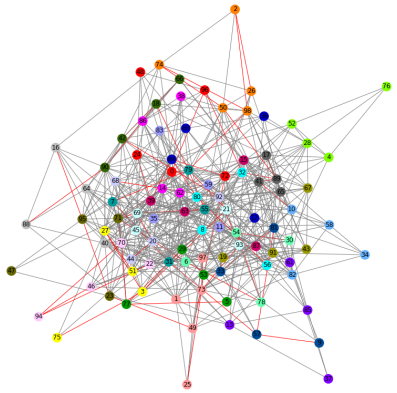

Figure 7: Final Contact Network. The color of the nodes represents the private residence each node belongs to. Each edge represents a contact between two nodes. Red edges represent a contact that resulted in transmission.

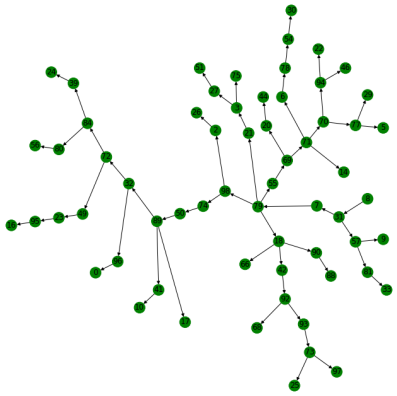

Figure 8: Final Propagation Network. Each edge represents transmission from one node to another.

During the simulation we stored the number of nodes in each SIR state, and the R_0 value every 100 time-frames. From this analysis we can see that the R_0 value peaks at 3.0 when the time stamp (ts) is 1700, then starts to decline and reaches 0 at ts = 6800 (Fig.10). The number of infectious nodes also rises until it peaks at ts = 5000, then start to decline and reaches 0 at ts = 7421. (Fig.11).

5.2.3 Analysis: Vaccination. When implementing vaccination in the simulation, the early adopters for vaccination were first introduced at ts = 3200. From there we update the vaccination cascade every 600 time-frames. In the final analysis, of this model 69% of the population decided to get vaccinated in the end. This resulted in 38/99 nodes getting infected, which is 33 less nodes infected compared to the unvaccinated model (a 46% decrease). The cascade representing nodes deciding to get vaccinated in the simulation,

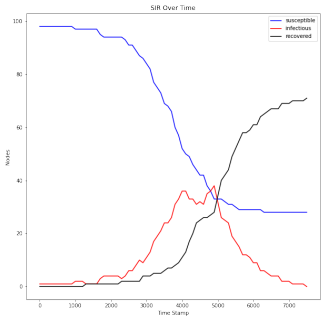

Figure 9: SIR over time.

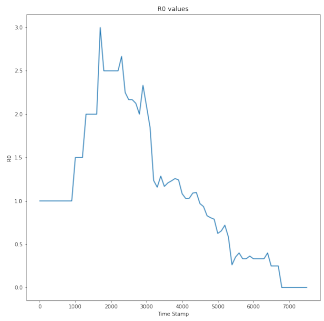

Figure 10: R_0 for each timestamp

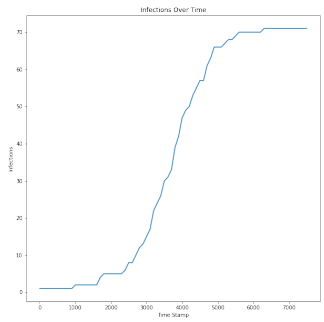

Figure 11: Number of infections over time

with the linear threshold model (Definition 4.1) is shown in Figure 19.

We can see from Figure 14 that the R_0 value for the unvaccinated model is an upper bound on that of the vaccinated model. We also see from Figure 13 that the infections over time plateaus a lot earlier for the vaccinated model compared to the unvaccinated model. This represents that we reach herd immunity a lot sooner in the vaccinated model.

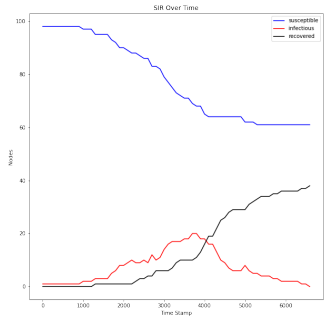

Figure 12: SIR over time for the vaccinated model

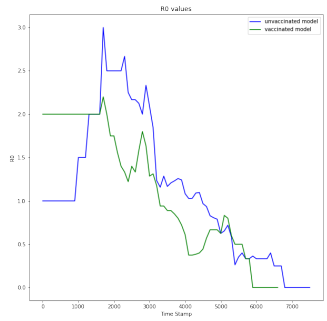

Figure 13: Comparison of R_0 values between the unvaccinated model and the vaccinated model.

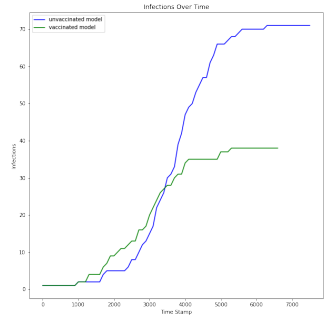

Figure 14: Infections over time comparison between vaccinated and unvaccinated model.

5.2.4 Analysis: Masks. We introduced masks in our masked model at ts = 3200. In this model individuals only wore masks when outside their private residences, we also assumed that if a node wears a mask, then that reduces its susceptible/infectious area by 30%. In the final analysis, the implementation of a public mask mandate resulted in 61/99 nodes getting infected, which is 10 less nodes

infected compared to the unmasked model (a 14% decrease in total infections).

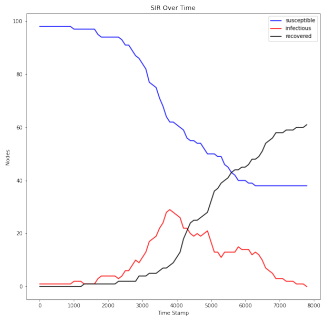

Figure 15: SIR over time for the masked model

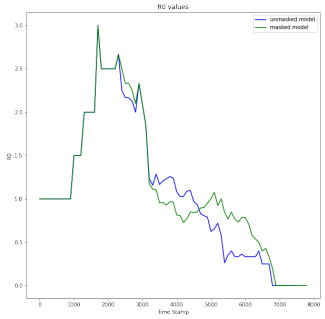

Figure 16: Comparison of R_0 values between the unmasked model and the masked model.

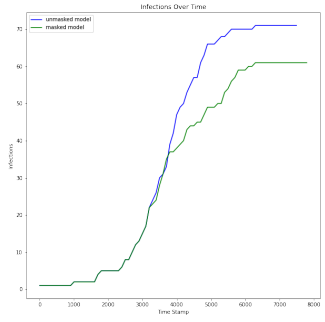

Figure 17: Infections over time comparison between masked and unmasked model.

We can see from Figure 17 that the R_0 value for the masked model is very similar to that of the unmasked model. However if we look at Figure 16, we see that the infections over time plateaus a little earlier for the masked model compared to the unmasked model. This represents that we reach herd immunity a little sooner in the masked model.

5.2.5 Analysis: Quarantine. If we wanted to identify what contacts would still remain from our contact network T, if a quarantine was implemented, this would result in a contact network in which each node only has contact to the nodes in its private residence. The best way to identify the private communities each node belongs to, is by removing the edges with the lowest weights, or in other words, removing the edge between the nodes that have been in contact with one another for the shortest amount of time.

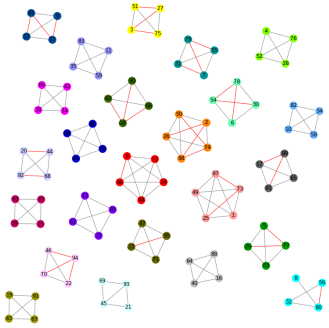

Figure 18: Community detection using lowest weight edge removal.

As shown in Figure 18, this technique is very efficient at identifying the private residences of each individual in our population.

6 CONCLUSION

In conclusion, we create two different simulations that have different levels of mobility. The first model, which is a CA model, has stationary nodes. The second model, which is a Mobility model, consists of nodes moving within a 2-D Cartesian area within their private and public communities. We construct a contact network and a propagation network from these simulations, then we contrast the results of these simulations. We then take on the challenge of implementing preventative measures in our models, such as public mask mandates, and vaccinations. We vaccinate nodes through the linear threshold model, then we analyze how the vaccinations and mask mandates have affected the resulting propagation network.

6.1 GitHub

6.1.1 Cellular Automata. Visit https://github.com/Vishwa061/epidemic-sim for the full code on our Cellular Automata.

6.1.2 Mobility Model. Visit https://github.com/milandrapic/COVID-19-Final-Simulation-Networks for the full code on our Mobility Model.

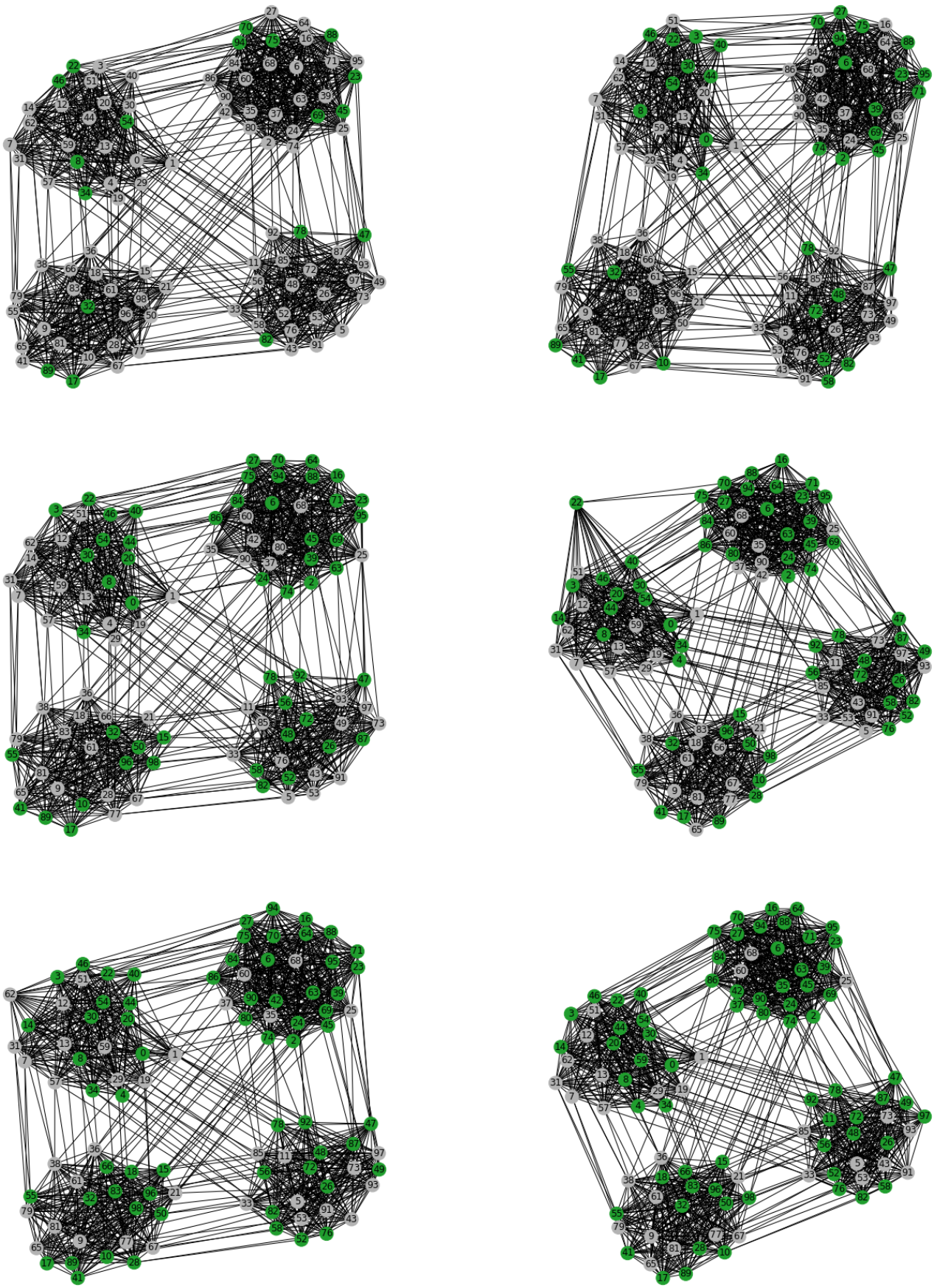

Figure 19: Shows progression of Cascade representing the nodes that decided to get vaccinated.

REFERENCES

[1] David Easley and Jon Kleinberg. 2010. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World.* Cambridge University Press. https://www.cs.cornell.edu/home/kleinber/networks-book/

[2] Naoya Fujiwara, Abhijeet R. Sonawane, Koji Iwayama, and Kazuyuki Aihara. 2015. Contribution of hidden modes to nonlinear epidemic dynamics in urban human proximity networks. arXiv:1512.01901 [physics.soc-ph]

[3] Sayantari Ghosh and Saumik Bhattacharya. 2020. A data-driven understanding of COVID-19 dynamics using sequential genetic algorithm based probabilistic cellular automata. *Applied Soft Computing* 96 (2020), 106692. https://doi.org/10.1016/j.asoc.2020.106692

[4] Danny Leung. 2021. Characteristics of businesses that closed during the COVID-19 pandemic in 2020. *Statistics Canada* 1, 3 (March 2021). https://www150.statcan.gc.ca/n1/pub/36-28-0001/2021003/article/00003-eng.htm

[5] Tilemachos Pechlivanoglou, Jing Li, Jialin Sun, Farzaneh Heidari, and Manos Papagelis. [n.d.]. Epidemic Spreading in Trajectory Networks. *National Library of Medicine* ([n. d.]). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8553416/#fm0070

[6] Yixing Wang, Hainan Xiong, Sijie Liu, Ara Jung, Trish Stone, and Leanne Chukoskie. 2021. Simulation Agent-Based Model to Demonstrate the Transmission of COVID-19 and Effectiveness of Different Public Health Strategies. *Frontiers in Computer Science* 3 (2021). https://doi.org/10.3389/fcomp.2021.642321