Modelling the Norwegian COVID-19 Outbreak Using Complex System Models

Kur, Pawel Michal[1]
Pedersen, Fredrik Haugerud[2]
Pop, Darius Grigore[3]

Oslo Metropolitan University, Faculty of Technology, Art and Design.
Pilestredet 35, 0166 Oslo

November 6th 2020

[1] Email: s169909@oslomet.no
[2] Email: s306631@oslomet.no
[3] Email: s351662@oslomet.no

Abstract

The main purpose of our project was to apply different complex system models and evolutionary algorithms to create realistic simulations of the Covid-19 pandemic. With the use of real-world data and properties related to Covid-19, as well as using our best judgement to estimate the properties on which no accurate data was available, we were able to create somewhat realistic Cellular Automata-, Network- and Agent-based models. We have then applied evolutionary algorithms to our models to further evolve them as well as emulate best- and worst-case scenarios. Based on the results of our simulations we were able to test different pandemic containment strategies and produce our own data-based recommendations. Our results suggest that a combination of strict social distancing, widely spread face covering mask use and a strategy of early testing and isolation of infected individuals provides the best results.

Keywords

Cellular Automata, Network and Agent based models. Evolutionary Algorithms.

Table of Contents

1 Introduction

The year 2020 was dominated by the outbreak of the Covid-19 pandemic. Regardless of where you live of what your line of work is, the pandemic is very likely to have affected your daily life. As such, further study and research into the spread, global impact and methods of mitigating further consequences of the crisis are of utmost importance. While a strict total social lockdown would most likely guarantee the best results in short term, it would also lead to severe social and economic issues. Using evolutionary AI algorithms and the available computation power, could potentially provide means for creating realistic simulation models that will help us discover middle ground solutions. This project explores three different simulation approaches that were previously proposed: Network-, Cellular Automata- and Agent- based models.

1.1 Cellular Automata

Cellular Automata (CA) is a complex systems model where the state of an automaton is affected by that of its surroundings. It was first invented in the 1940s and 1950s by John von Neumann and Stanislaw Ulman, created to describe the self-reproductive and evolvable behaviour of living systems (Sayama, 2015). In layman's terms, the model is based on having a grid of cells (automata), where each cell's state is updated based on the state of its neighbouring cells. The conditions for how and why a cell's state is updated can range from very simple to complex rules, making the model very flexible in terms of complexity.

CA has been extensively utilized for modelling phenomena such as molecular dynamics, evolution of living organisms, interactions in populations and economical dynamics (Sayama, 2015), showcasing the versatility of the model's use cases.

For the sake of not bloating the report with a more technical definition, we assume the reader is either familiar with the mathematical theory behind the behaviour of Cellular Automata, or not too interested in the technical details.

1.1.1 Our Implementation of Cellular Automata

For our simulation we have implemented a Host-Pathogen version of the CA model, often used in epidemic research. Here each automaton represents an individual organism (a person) and their states reflect their physical condition (Healthy, Infected, Dead or Recovered).

Using a Moore-neighbourhood with a radius of one[4] and separate state-transition algorithms for Healthy and Infected automatons, we have a simulation where a Healthy cell with an Infected cell in its neighbourhood has a chance of becoming infected. Infected automatons have a chance of dying, and dead or recovered cells are replaced with new, healthy cells.

[4] Cell to be updated = (x_0, y_0), Neighbourhood = $\{(x_{-1},y_1), (x_0,y_1), (x_1,y_1), (x_{-1},y_0), (x_0,y_0), (x_1,y_0), (x_{-1},y_{-1}), (x_0,y_{-1}), (x_1,y_{-1})\}$

Exactly how the infection and mortality chances are calculated will be covered in chapter 3.1.

1.2 Network Model

A network (or graph) is a useful framework for representing interactions between the components of a complex system (Pellis, et al., 2015). A network consists of a set of nodes, and a set of edges connecting the nodes. An example of a network could be cities that are connected by roads. In this case the set of nodes would be represented by the cities and the edges by the roads between them.

The pandemic problem can be represented by a weighted undirected graph which additionally is complete. A weighted undirected graph is a graph where the connection between the nodes are symmetric(meaning if there is an edge from node a to node b, then there is also an edge from node b to node a) and there is a cost associated to each edge. In our specific case, the nodes are defined by the list of people that are chosen randomly from our dataset and the edges defined by the distances between people.

A complete graph is a graph where there is an edge connecting any two nodes. Traditional approaches in modelling pandemics rely heavily on degree-based mean-field theory, which means the more nodes a certain node is connected to the more likely is to infect more people. However, that might pose some limitations for the case of Covid-19. One issue might be due to distance between 2 people. This might be a factor in infection since the virus is transmitted when people get into contact with each other.

1.2.1 Our Implementation of the Network Model

The implementation is based on a SIS model (Wang, et al., 2019) In this model, a person can either be infected or susceptible to infection. We are assuming in this case a person can become infected after recovering, since there is an ambiguity involved in developing immunity against the disease. Another argument for choosing SIS over more complex models, are the scaling problems mentioned in chapter 3.2.

The initialization routines will be described in this paragraph.
Firstly, the number of nodes we use is randomly generated from the interval [1,301]. This was done to assess whether having more people will influence the development of the disease. The next step is to extract a random sample of that size from People.csv file, which will represent the nodes of the network. After we create the nodes, for each pair of distinct people, we add an edge with the cost being the distance between them. This accounts for social contact influencing the spread of the disease.

The most important part of the simulation will be the update step. As the simulation progresses, at each time a random person is selected. Depending on the person being susceptible or infected, we will compute probabilities for the person entering the other state of the model. A dice will be rolled in both cases.

In the case of the person being infected, the chances of recovery will be computed based on how long they have been infected. For a healthy person, we will select non-quarantined neighbour randomly, and if they are infected, the probability of infection will be computed based on distance and a random factor, which could account for mobility within the city. If quarantine is applied and the person selected in the step gets tested positive after being infected, all potential social interactions will be removed. This excludes the possibility of an isolated person infecting other people, which is a realistic assumption.

1.3 Agent Based Model

Agent based models (ABMs), are widely applied as a useful and very flexible method for the simulation of complex systems. In agent-based models all entities that we want to be part of our simulation is called an "agent". Each agent can be assigned a magnitude of different, individual properties. The properties can vary from general descriptive properties such as: age, height or health status, to more complex ones such as personalized movement patterns, shopping habits or social relationships.

The advantage of being able to personalize each agent is that the ABM can more accurately represent the real-life conditions of the problem we wish to simulate. The inherent benefit is that the results provided by the simulation will most likely be more accurate. The downside to this is that depending on how much more accurate and personalized we wish to make each agent and the model in general, the amount of work and data necessary to implement such a solution increases proportionally. This is an issue that becomes even more apparent if we wish to run our simulation with hundreds, or maybe even thousands of individual agents. In addition to requiring large amounts of accurate data to personalize that number of agents, such a simulation would also require a vastly more powerful hardware to run.

While the simulation is being run the agents will move freely in accordance with the pre-determined rules. When the agents "collide" or make close enough contact with each other a desired interaction will be performed between them depending on the different types of agents.

1.3.1 Our Implementation of the Agent Based Model

In our agent-based model which is based on the "Predator-Prey" model, a population of 400 is randomly divided into healthy and infected individuals. Each agent is then randomly placed on the board. Depending on if the social distancing policy is in place or not the agents may be placed in close proximity to each other. When the simulation begins, a single agent is chosen from the list of all agents and then performs a movement in a random direction. If the chosen agent adheres to the social distancing policy, it will attempt to find a location to which it can move that allows it to keep the required distance to the other agents. If after several attempts it is unable to find such a location, it will remain in its current position.

After the movement is completed (or skipped) the agent will check its surroundings for any neighbouring agents that are considered to be "colliding" with it. In the situation where it finds such a neighbour, a predetermined interaction will occur. In the case where the current agent is healthy and one or more of its neighbours are infected agents carrying the virus, there will be a chance that the healthy agent will get infected itself. If the current agent is already infected, it will look for healthy neighbours, in which case there will be a chance that it will infect them. This process is repeated in random order for every single agent in the agent list. After all agents have made their move, a "day" passes, the next step begins, and the sequence is repeated.

2 Research

Both before and during implementation of the simulations we did research on how the Corona pandemic has behaved, and what factors into its spread. Not all the simulations implemented the research in a similar fashion but made choices according to what made sense for the model it was based on.

2.1 Virus Progression

Figuring out the virus' general lifecycle regarding what stages it goes through from a person is infected until they are recovered was a vital part in making the simulations. Our findings revealed that the virus (on average) has a 5 day incubation period, before giving it's host symptoms for 14 days, with the infected person usually being infectious a day or two before symptom onset (CDCP, 2020). On average the whole infection lasts for 20 days (CDCP, 2020), with the longest recorded case (as of October 5th 2020) being 61 days. (Osborne, 2020).

How we implemented these statistics in our simulations varied, i.e. with the Cellular Automata simulation using the average infection duration as the minimum duration before a patient has a chance to recover (and then get increasingly greater chances of recovery for each timestep) while the Agent based Simulation gives the patient a chance to recover after 15 days, with the chance increasing for each passing day.

2.2 Reinfection

When it comes to recovered patients becoming reinfected with the virus, there has been some confirmed cases where former corona patients who have tested positive for antibodies have been retested for the virus several months later and showed positive results. They have however not shown any symptoms, and their infectiousness is up for debate with the general consensus being that they are not (Stamataki, 2020).

Since there is nothing that proves or disproves reinfection, we decided that the author of each simulation should decide how to handle it. In the Cellular Automata and Agent simulations we chose to replace recovered people with new cells/agents, indicating that they cannot be

reinfected, while in the Network based simulation the node stays in the simulation and may be reinfected.

2.3 Risk Groups

When we are talking about risk groups in this report or the simulations, we mean those who are in risk of becoming mortally ill from contracting the Coronavirus, and generally consists of the elderly (ages 65+) (Krüger, 2020) and those with cancer, diabetes or heart- or lung-disease (Stranden, 2020). Excluding the elderly, there are approximately 1.8 million people in Norway considered to be in a risk group (Stranden, 2020). With a total population of 5.3 million people (Statistics Norway, 2020), that is equivalent to 35% of the total population.

In the Agent and Cellular Automata based simulations this is reflected by giving each cell/agent a random age and a 35% chance of being in a health-related risk group when they are initialized. If they are over 65 years old or in a risk group, they get a chance of dying from the disease. Those who are not in any age or health related risk group are not, as the chances of healthy, relatively young people to die are very low (Ritchie, et al., 2020). In the Network model it did not make sense for a node to die, and the whole concept was dropped.

2.4 Preventive Measures

In our simulations we choose to incorporate isolation for people with symptoms, social distancing and face masks as measures to prevent virus spread. Concrete data on the effects of these preventive measures and how many people follow them were either lacking or inconclusive, making values related to these prime targets for our evolutionary algorithms to play around with.

We did however manage to find sources indicating that social distancing reduces your chances of getting infected by up to 50% (Matrajt & Leung, 2020) and wearing a facial covering mask reduces your infection chances by up to 65% (Kushman, 2020).

In terms of how many are always wearing a mask or able to practice social distancing while in public the data was lacking. Except for an article where the Norwegian Institute of Public Health had noted how many percent of public commuters where wearing a mask one afternoon in Oslo (22%) (Dommerud & Schwencke, 2020), we were not able to find anything conclusive.

2.5 General Statistics

The Norwegian Institute of Public Health's website (NIPH, 2020) provided live updated data and was used as our primary resource on values like how many are infected, dead and recovered from the virus, as well as other datapoints which gave us a better insight into the virus' behaviour.

2.6 Obtaining Population Density

To make the network model as realistic as possible, we pulled real people's addresses from 1881.no using a self-made webcrawler,that is in file scraper.py. The results are written to a csv-file called People, and the dataset contains about six thousand entries from the Oslo-area. Using this data, we can get an approximation of the population density in different parts of the city.

This approach has several advantages, such as including probability that people are infected during commutes. However, some invalid results (such as companies) are also included in the results, but it should still be a decent approximation.

3. The Simulations

This chapter will be detailing core functions, values and the general flow and thought process behind the simulations. Note that this is not an in-depth technical documentation, which is found by reading doc-strings and comments in the source code.

All three simulations were based on sample code given from the PyCX-library. The core concepts in the PyCX-simulations have been expanded upon in all the simulations, as the samples simply serve as an introduction to working with complex system models in Python. The GUI logic has been somewhat altered to fit the more complex tasks of our simulations but remain mostly unaltered.

3.1 Cellular Automata based Simulation

The CA simulation's purpose is to model the spread of the Corona Pandemic in Norway on a linear scale in regards to the number of infected and dead people, meaning at the end of a simulation run it achieves the same numbers as we have in the real world, but does not take into consideration how the spread rate has been very turbulent during the whole pandemic. Most of the values in the simulation are based on real world research and data, with some exceptions to be mentioned later.

The simulation is using real-world data for how many infected and dead people there should be after 245 days (we pulled data from FHI on November 2nd and consider March 2nd to be the start of the outbreak), where one step represents one day. After 245 steps the simulation restarts with new values depending on what the evolutionary algorithms have calculated.

3.1.1 Core Classes and Functions

Most central in the CA simulation is the Person class, which naturally represents a single individual. A person object is used to keep track of that individual's health state, their infection, age, if they are in a mortal risk group and other factors related to whether they are following infection preventive measures.

Infection is also a class keeping track of attributes related to a person's infection in the form of its duration, if it is in an infectious stage and/or lethal to its host (determined by whether

the host is in a risk group). Every person always holds a reference to an Infection object, but the Infection object is never updated (progressing) before a Person's health state is set to being Infected.

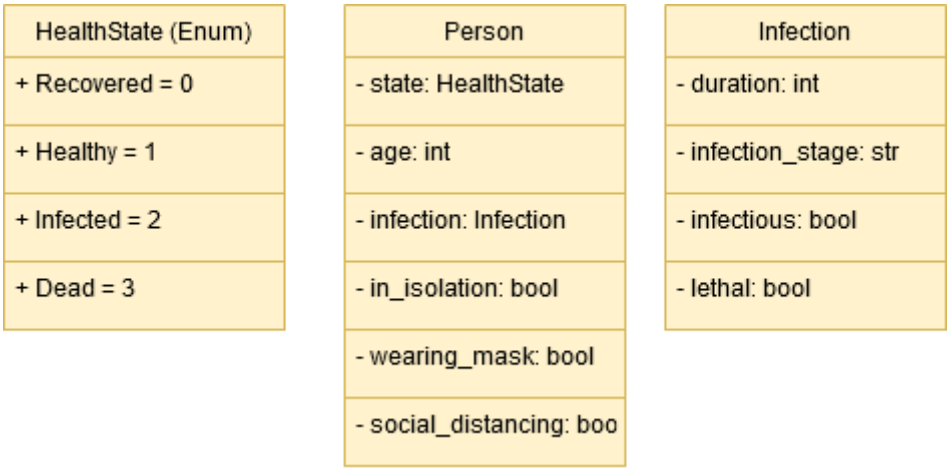

Figure 1: Person, Infected and HealthState classes in the CA Simulator

The simulation is based around three main functions, initialize, update and observe. In initialize, an array representing the simulation's current state, is filled with person objects where every person generated has a 1% chance of starting out as infected. All other values needed to run and keep track of the current simulation run's results are also initialized. In the observe function, the current simulator state is passed to pylab functions used to generate the graphical aspects of the simulation.

The update function is a tad more complex, as it iterates through the whole stateConfig array and checks every person object's health state. Dead and Recovered people are replaced with new person objects, while healthy people go through the algorithm for checking their Moore-neighbourhood for infected people. After considering the person's attributes for infection prevention, a random number is compared with the person's chance of becoming infected to determine if they are to become infected. Already infected individuals are being checked for whether they should recover or die, where those who has an Infection with the lethal attribute will be killed. Those who has an Infection which has lasted longer than the average duration of the virus will be given a chance to recover, where the chance of recovery increases the longer they stay sick.

3.1.2 Evolutionary Algorithms

While most of the values in the simulation are based on real-world data, those determining whether a person gets infected or dies are not. While they have been tuned to not make the simulation's results go way off course from the real world, they do not make the simulation produce total infected, dead and recovered individuals which are completely on par.

We also have other variables of which there are little to no data. These are how many percent of the population who are using face-covering masks, and how many are practicing social distance. These questionable values were prime targets for being toyed with or adjusted, and

to do this we implemented two different evolutionary algorithms: one to adjust the simulation's infection and mortality chances, and another one for running through different scenarios of how the pandemic would have evolved if we had ben better (or worse!) at implementing preventive measures.

The adjustment algorithm is a genetic algorithm, which purpose is to fine tune the infection and mortality chances by looking at the previous run of the simulation and comparing the achieved number of infected and dead with real world statistics. The algorithm keeps track of the highest and lowest chance-values the simulation has worked with in previous runs, and if the produced results are not within an adequate range (we currently allow a deviation of 10% from real-world statistics) it sets the chance value to (MaxChance + MinChance)/2 to make it eventually converge at a value which (almost[5]) always produces adequate results.

Adjusting the simulation each time is somewhat unnecessary unless there have been made changes to the simulation since last time the adjustments were run. In the current iteration of the simulation the default values for Infection- and Mortality-Chance have been set to approximate values calculated by the adjustment algorithm, and the adjustments themselves have been disabled.

The scenario algorithm changes the values for whether isolation is mandatory for those who are showing symptoms, and how many percent of the population are using face covering masks and practicing social distancing. After an initial run using values based on what we could find from research done on the Norwegian populace and their habits (enacted mandatory isolation, 22% using masks and 30% able to maintain social distance), both values are set to 0% and mandatory isolation to False. Then, for each run of the simulation the mandatory isolation variable is toggled between True and False. On every second run the social distance percentage is incremented by 10%, while on every fourth run the mask usage percentage is increased by the same amount. This results in the simulation being run with all possible combinations of [True/False], [0 – 100%], [0 – 100%].

3.1.3 Running the Simulation

The simulation is run from the simulator_functions.py file, and when first started you will be greeted by the initial state drawn up and a control panel.

[5] Due to the random nature of the simulation, i.e. we produce random numbers to determine if someone gets infected, dies or recovers, there has been edge cases where even the adjusted chance-values results in inadequate simulation runs.

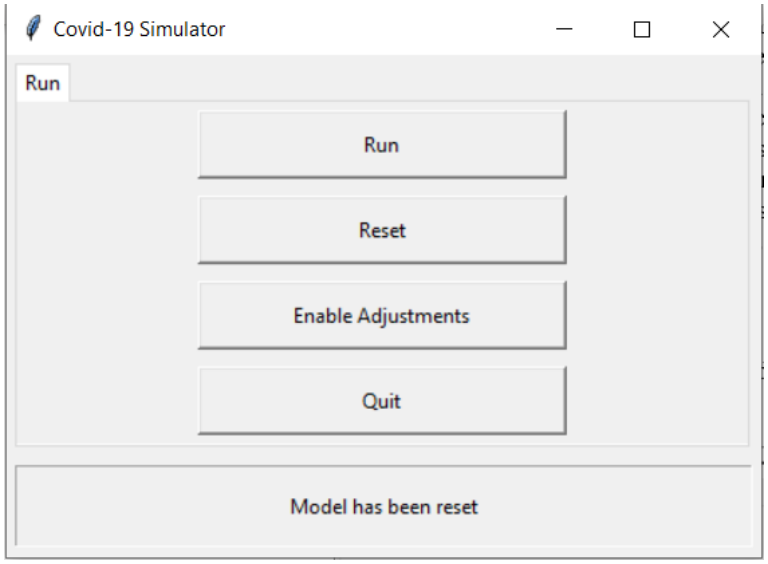

Figure 2: Control Panel for the CA Simulation

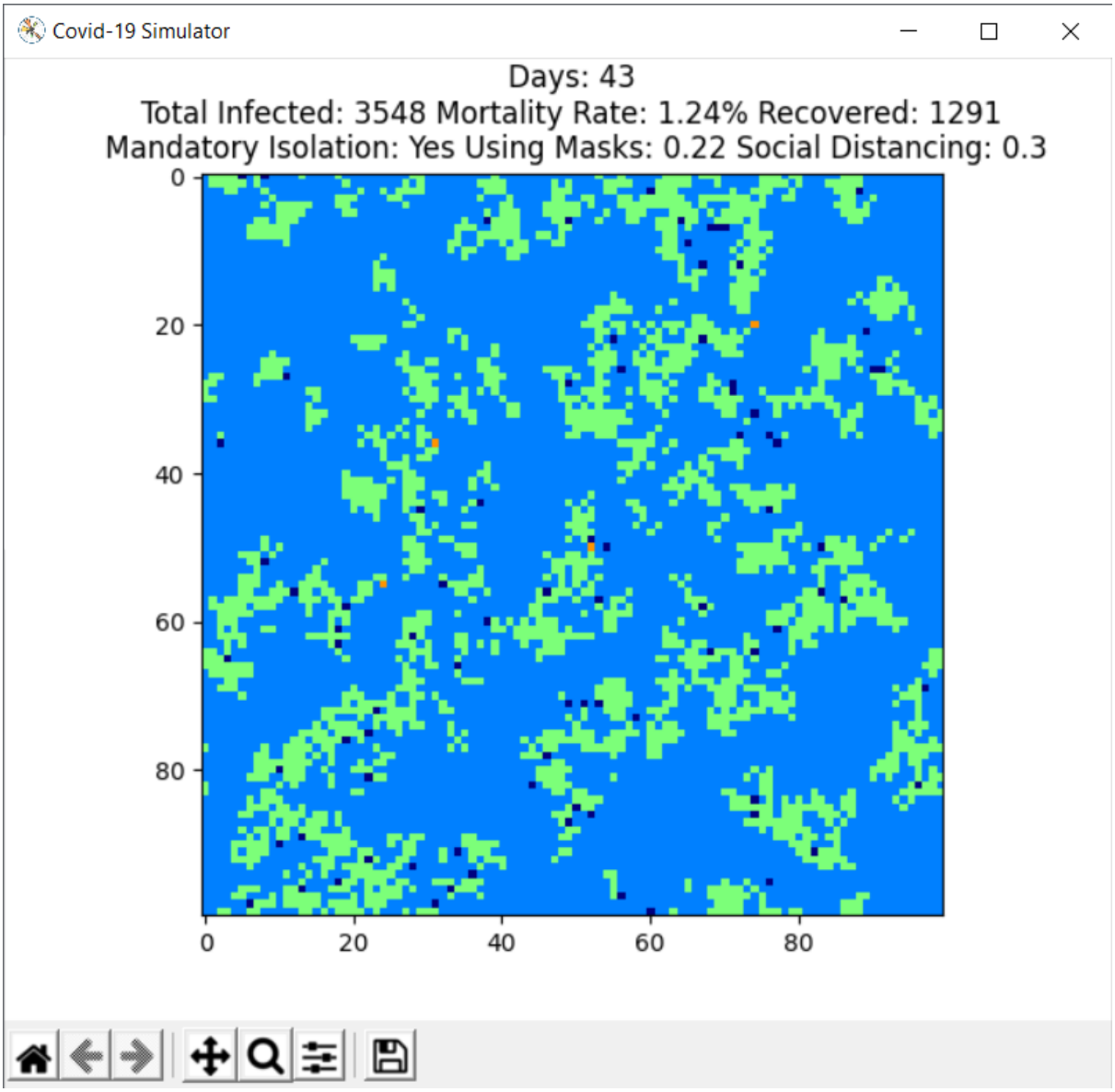

Figure 3: The CA Simulation. Blue = Healthy. Dark Blue = Recovered. Green = Infected. Red = Dead.

Running the simulation is very straight forward. Run, Reset and Quit buttons does exactly what their names suggests, while the Enable Adjustments-button enables the adjustment algorithm, which will then disable itself once it achieves satisfying results. Two things to note:

1) Enabling adjustments will disable the scenario algorithm until disabled.
2) Use adjustments before completing the first run of the simulation, otherwise there might be some buggy behaviour. Currently there is no logic implemented in the GUI to prevent you from doing this.

When the simulator has run through all scenarios, it will terminate and draw up graphs with the results from all the simulation runs, displaying the fitness scores of each run, and graphs displaying how many were infected relative to how many percent of the populace were wearing masks and practicing social distancing.

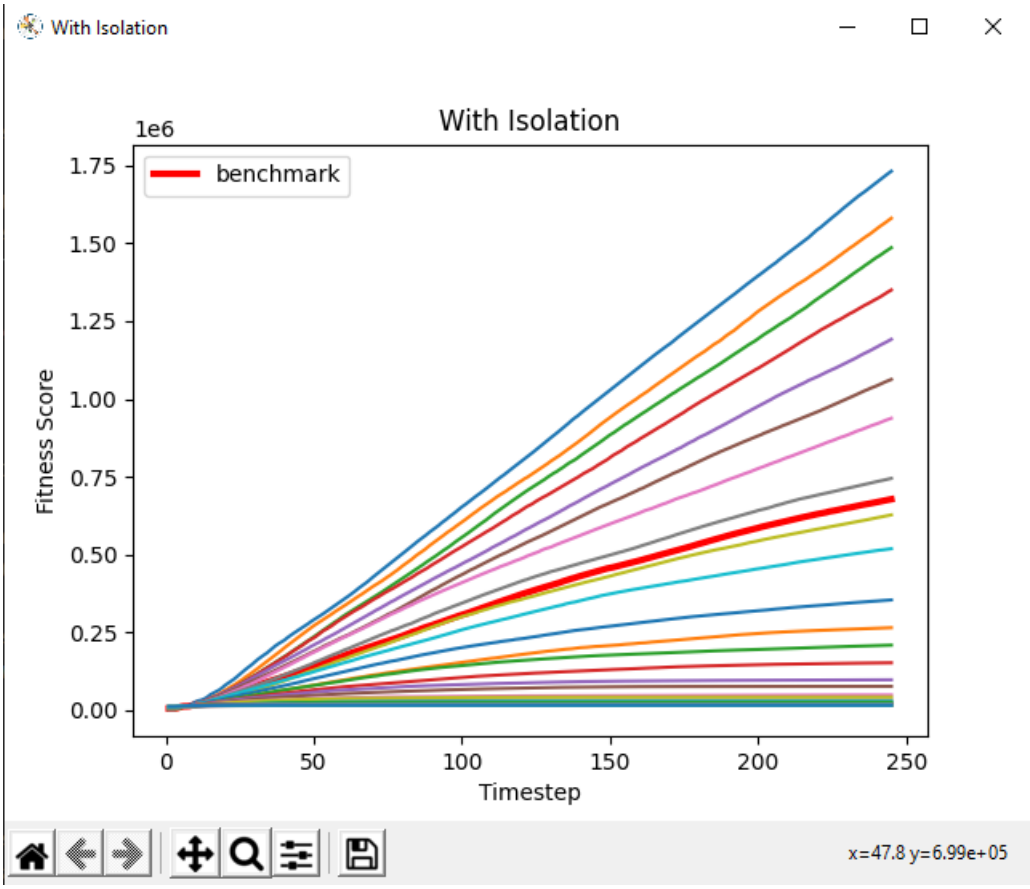

Figure 4: Graph Displaying Fitness Scores for simulation runs where mandatory isolation was enabled. Lower score is better, benchmark is run based on real-world values.

3.2 Network based Simulation

The approach is based on using locations of people residing in Oslo as described in Chapter 2.6. The limitations of the approach will be described below, while section a) and b) will describe the essential tools used in the simulations.

There are certain limitations in creating a realistic simulation of the virus, mainly related to state-of-art available computing power. A complete graph contains n(n-1)/2 edges, where n is the number of nodes. Trying to simulate the behaviour of a world as small as containing 1000 people seemed to run out of memory on a machine having 16 GB RAM. The result is that some variables that might be important for the simulations will have negligible impact on the maximum usable scale.

A trade-off can be made to reduce memory and not using a graph to store the distance between people and recomputing them at each step. This can prove to be a better solution for very large computing power that as an instance is the availability of these resources at Google, but for the scope of this project that will result in infeasible amounts of time taken to eradicate the disease in the simulated world. The bigger the world we ran the simulations in, the more time it took to heal everyone. Even for a world bigger than 500 people, there was not enough time for this project to extract meaningful results.

Alternatives can include reducing the size of the graph and only consider people that are in close vicinity. However, this excludes the possibility of some people infecting others in public transport as an instance.

Some steps to address these problems to a small extent involved using the MVC pattern[(Gamma, Helm, Johnson, & Vlissides, 1994)].The file NetworkModel.py contains GraphController and Visualizer classes that will separate the data layer from the view. In case of very large graphs, the image generated is non-intuitive for humans. The file SimulationsWithoutVisualizers.py is responsible for running a big number of simulations and reporting the results without creating graphics. In order to visualize "small words", the files SimulationVisualizerQuarantineApplied.py and SimulationVisualizerWithoutQuarantine.py will run while showing the graphs described in section 3.2.2.

a) GraphController

The GraphController class in the NetworkModel.py represents the data associated with the "world".

a.1) Variables of GraphController

The table below contains the most important variables used in the GraphController class with the corresponding explanations:

Variable name	Explanation
detection_probability	The probability for an infected person to be detected. It is suggested that not all people who are infected are tested (Giattino, 2020).
distance_graph	The graph where we keep the distances between the people in kilometres. Used to restore edges in the relationships_dynamics_graph when a person is healed.

relationships_dynamics_graph	The graph of distances for non-quarantined people.
infection_probability	The probability that someone starts as infected in the simulation.
quarantined_people	A set that keeps track of the quarantined people during each step of the simulation.
distances	An array containing all the distances between people.
time_elapsed	The number of steps that have been done so far.
time_infected	A mapping between the people and the time they got infected.
world_size	The number of people to randomly choose from the population.

There are some variables that might be important for the development of the disease that were omitted and the table below shows them and explanation for not being **included:**

Variable name	**Explanation**
mortality_rate	The mortality rate of the disease. As of November 3rd, 2020, in real world this is around 2.5%, which makes it negligible for the size of our simulations. There is also an uncertainty for this number, since there could be people that are people that got the disease but not tested (Giattino, 2020).
mask_reducer_probability	Research papers have shown that wearing a mask reduces the infection probability by 65% (Kushman, 2020). However, this can be equivalent to the random factor introduced when computing the probability of someone getting infected when they have been in
r_number (infection rate)	This is an often mentioned number related to Covid-19. The number could have been used to assess how realistic the model. However, due to the scale of the system we decided not to use it.

a.2) Methods in GraphController

The table below includes the important methods in the GraphController class and explanations

Method name	Explanation
initialize	Initialize the world. This involves creating the 2 graphs and computing distances between people in km.
update_world	Execute one step in the simulation. The function enables us to select whether we apply quarantine or not on the quarantine parameter. A random person is selected. If they are infected, we compute the probability of them getting recovered. There are three possible ways considered in the UsefulMathematicalFunctions class. Research suggests that it takes about 14 days for a person to recover and we try to include that. The most accurate solution seems to be that the probability of infection is logarithmically affected by time. Otherwise, we select a random person that is not quarantined and check if they are infected. If that is the case, we calculate the probability of the non-infected person to get infected by weighted sum of the distance and a random factor(which can be movements or people wearing masks).If we apply quarantine, additionally we roll a dice whether the person is tested or not.
create_nodes	Add the people to the graph. A random sample of the world_size is extracted.
create_people_mapping_to_coordinates	Read the csv containing the people and transforming it to a dictionary containing the coordinates where they live.
create_edges	Create an edge between each pair of nodes with the cost being the distance in kilometres.

b) UsefulMathematicalFunctions

This class includes mathematical functions that are used in the GraphController class. It is contained as well in the NetworkModel.py file. The table below illustrates them and explain the context they are used in:

Method name	Explanation
distance_linear_normalizer	In order to compute the probability of infecting someone and including the distance, we create a mapping of the original distances in km to the range

	[0,1], using a normal distribution.
calculate_distance_haversine	Given two pairs of latitude and longitude, this function applies the haversine formula (spk578, 2017) and returns the distances in kilometres between them. This is useful when creating the edges of the graph.
recovery_probability_with_inverse_logarithmic, recovery_probability_with_inverse_linear	These functions assume that the longer the person has been infected, the more likely they get recovered. The most realistic corelation seemed to be logarithmic.

3.2.1 Evolutionary Algorithm

It is useful to know good strategies in order to eradicate the disease while maintaining the society as open as possible. Therefore, the file EvolutionaryAlgorithm.py uses an evolutionary approach to find best strategy in order to achieve the goal. The algorithm will search for an optimal value of detection_probability (this can be realistically achieved by the country increasing the testing capacity as an example or deciding to perform less tests) and whether quarantine or non_quarantine will have a better impact. The detection probability being increased is irrelevant in our algorithm when we decide to not apply quarantine. At each step, we set up a new random detection probability and then try to run a scenario based on that. Moreover, we also run the scenario when not applying quarantine. The third scenario is run with the old detection_probability. If the new detection probability yields a better result, we update the detection probability.

In each evolutionary step, multiple simulations are executed to find an accurate enough result as described in chapter 3.2.2.

Out of 100 steps, the algorithm found it is better to use quarantine in 65 of the cases and to generally choose a higher detection probability.

3.2.2 Running the simulation

There are 2 classes of different scenarios that are implemented by the code:
whether people are quarantined or not. In terms of execution flow, when applying quarantine, all adjacent edges will be removed for some nodes, as illustrated by Figure 6. In the case of not applying quarantine, the graph will remain complete throughout the process. The figure below illustrates the initialization

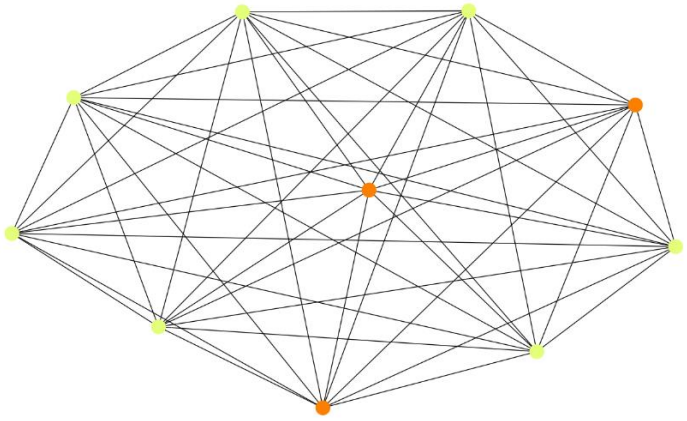

Figure 5: A graph representing the initial state of a randomized world with size 10

If a node is green that means they don't have the virus. In case of the orange color, it means the person has been infected. The placement of the nodes signifies how far they are from each other in relative terms.
When an infection is detected and thus the person quarantined, all the edges adjacent to the node will be removed, as highlighted below by the orange node in the middle:

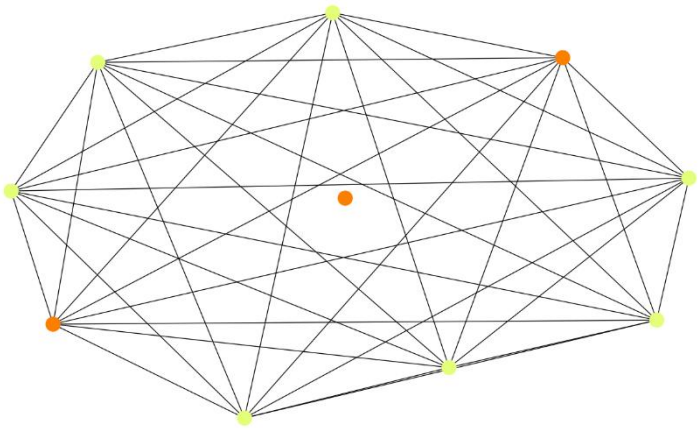

Figure 6: The representation of the graph after some changes occurred. The orange node on the left-hand side was infected and not detected, so he remained connected to the other people. The orange node in the center was healed and then got reinfected,that time the infection being detected.

When the person recovers, the former connections will reappear:

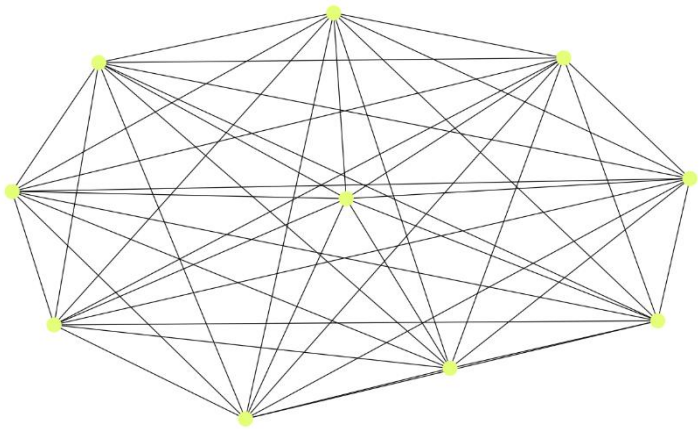

Figure 7: The representation of the graph when the disease is eradicated. The world gets in a state where everyone is healed.

The last image also illustrates the end of the simulation. However, since the model is based on probabilities, sometimes everyone will get recovered in an unrealistic amount of time (such as 0 steps). To get an accurate estimate it can be useful to take averages of multiple simulations. We ran 1 million simulations for multiple world sizes and we found that taking average of 3000 samples provides a very close estimate to the average of the whole set. The averages depend on the size of the world, and as instance for a world of size. The above results were achieved after running the file TimeAnalyzer.py.

3.3 Agent based Simulation

3.3.1 Main Features

The main features of our agent-based model include:

Age and risk group assessment
Each agent gets assigned an age value between 18 and 100. Agents of the age 65 or older will be a part of a risk group and will have a higher chance of dying from the virus. This chance gets progressively higher the older the agent is. In our model agents who are not part of a risk group cannot die of the virus.

Social Distancing Policy
As a part of our agent-based model simulation there is an option to enact a social distancing policy. Social distancing is one the most efficient and effective ways to severely limit the spread of the Covid-19 virus. If a sufficient distance is maintained between agents, the virus will have no possibility to spread effectively and find new hosts. While a 100% participation rate in social distancing would be the ideal scenario, it is also very unrealistic. Our simulation takes into consideration that there always will be a percentage of the population who is either

unable or unwilling to social distance. We were unable to find concrete real-world data that would give us an accurate estimate of the ratio between people who are following the social distancing guidelines and the ones who refuse or are unable to do so. In our simulation we used our best judgement and set the amount of population not participating in social distancing at around 30%. In our model people who are distancing themselves will never come close enough to another person to be in danger of contracting the virus, and can only be infected by an infected agent who breaks the social distancing policy.

Mask Wearing Policy

In addition to being able to enact a social distancing policy we are also able to run a mask wearing policy. While face covering masks do not reduce the death rate of the virus and are not as effective as strict social distancing, they do scientifically lower the probability of getting infected or infecting others. Face covering masks are especially important when keeping a proper social distance is not possible. In our agent-based model individuals wearing masks are 65% less likely to contract the virus from an infected neighbour. As with the social distancing policy, the simulation takes into account that not everyone will agree on wearing a mask.

Isolation Policy

Each simulation step or "day", infected agents will have a progressively growing chance to be diagnosed. When an infected agent is successfully diagnosed, he will get placed into isolation. Isolated agents are locked in place and cannot move until they recover from the disease.

Recovery

After 20 days an infected agent will have progressively growing chance of recovering from the virus. The probability of death is always checked before the probability of recovering, simulating the growing chances of dying if being sick for too long.

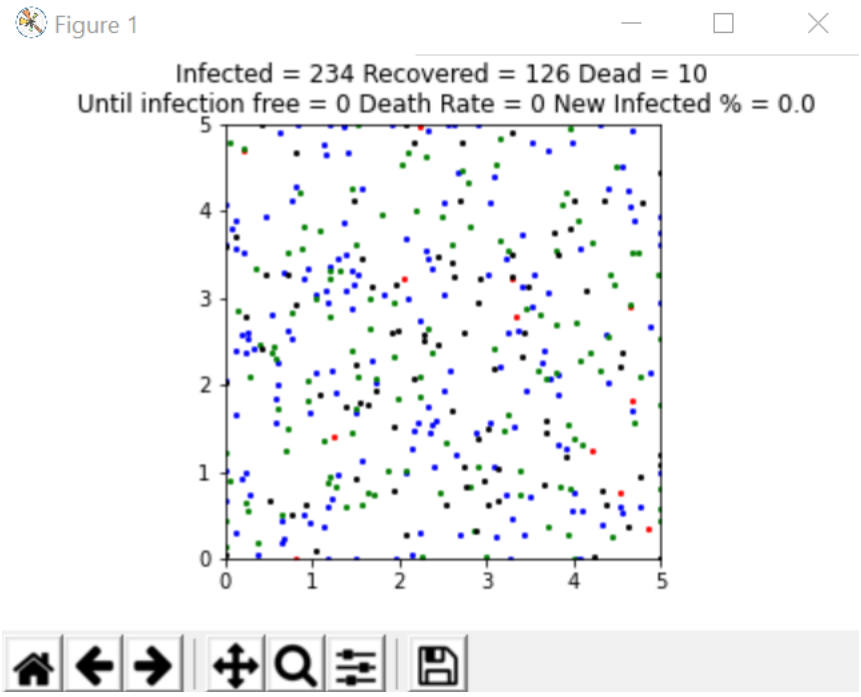

Figure 8 - A figure of the main simulation window

Agent Class

The main part of our agent-based model is centred around the "agent" class. This class describes consists the following parameters:

Variable Name	Description
Type	Describes the type of the agent. Our simulation recognizes four different agent types: 'h' – healthy. Represented with the colour 'blue' 'i' – infected. Represented with the colour 'red' 'id' – infected diagnosed. This agent is under isolationRepresented with the colour '**black**' 'r' – recovered. Represented with the colour 'green'
age	Stores the age value of the individual agent

X and Y	Stores values for the x and y coordinates of each agent
risk_group	Stores information about the agent being a part of a risk group. 0 – not in a risk group 1 – young but has a pre-existing medical condition 2 – is over 65 years old 3 – is over 75 years old 4 – is over 85 years old
Agrees_to_social_distance	stores information about the agent's willingness to follow social distancing guidelines: 1 for YES, 0 for NO.
agrees_to_wear_mask	stores information about the agent's willingness to follow mask wearing guidelines: 1 for YES, 0 for NO.
sickCount	stores information about the number of days and infected agent has been sick for.

Important Constants:

Constant Name	Description
total_population	Total amount of agents
sick_rate	Chance for an infected agent to infect a healthy agent whilke close to them
High_risk	Percentage of the population who are under 65 years old but are in the high risk group because of pre-existing conditions
Recovery_time	The time it takes on average for an infected agent to recover.
Death_rate	The main death rate coefficient. This is the number that decides

Face_mask_effectivnes	The amount wearing a face mask reduces the probability to getting infected while in contact with an infected agent

Control variables:

face_mask = True / False - Turns face mask on/off
social_distance = True / False - Social distancing on/off

adjust = True / False - Adjusts the death rate coefficient. Cannot run at the same time as the evolve function.

evolve = True / False - Turns evolution on / off. Cannot run together with adjustment function.

3.3.2 Adjustment Function

We know from our research that the real world death rate for infected people is around 2%. That's why it is important to set our simulation death_rate coefficient to a number that will give us a comparable 2% actual death rate in our simulation. The adjustment function test all the values for death_rate coefficient between 0.01 and 0.1 in 0.2 intervals and compares them to simulation death rate.

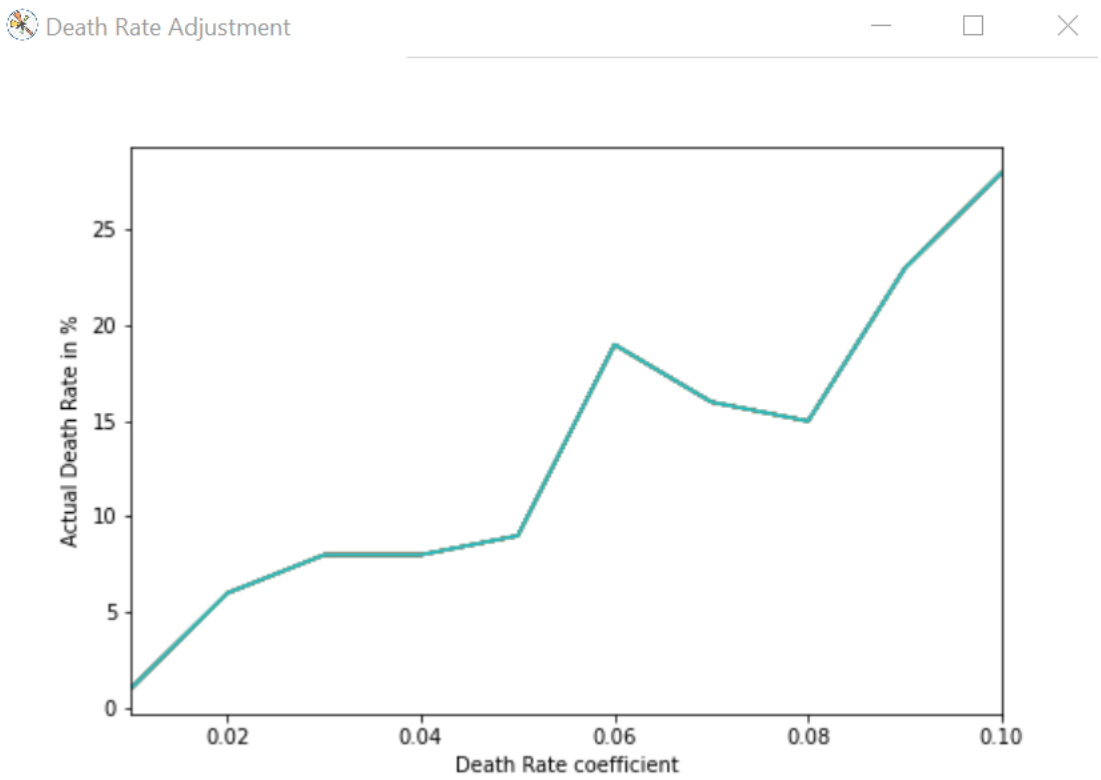

Figure 9 - A figure showing the relationship between the death rate coefficient the actual simulated death rate.

Based on the result from this figure we were able to determine that a death_rate coefficient of 0.01 will make our agent-based simulation run with death rate statistics matching the real-world data. The deviations in the graph are related to the random nature of the simulation. A larger sample size would eventually even out the graph but was not possible due to very long simulation times.

3.3.3 Evolutionary Algorithm

A simple evolutionary algorithm has been implemented in our agent-based model. The goal of the algorithm is to find the ideal values for the percentage of agents that have to wear face masks, and the percentage of agents that have to social distance, so that the pandemic can be controlled in the most efficient way.
To achieve this, we have created a fitness function that will help us compare the individual results.

Fitness Function

The fitness function equals to: total_population * (150 – infection_free) * (1- death_rate * 10) * (1- new_infections)*1.3 – (1000 * (mask_usage + social _distancing)

total_population - The total initial agent population is a constant large number.

infection_free - The total amount of days until infection free. The higher this number is the lower the fitness function will be. The number 150 functions as a weight.

death_rate - The death rate. Usually a relatively small number so we multiply it by 10 to give it a larger weight. The higher the death rate the lower the fitness function

new_infections - The rate of new infections. The more the virus was able to spread the lower the fitness function. This statistic is very important and carries a very large weight for the fitness value. The larger the percentage the lower the fitness value will be.

mask_usage + social_distancing - The final part of the fitness function is the subtraction of sum of the percentages of people having to wear masks and to social distance multiplied by a weight. While a 100% usage of both face masks and social distancing would undoubtably give the best results, it is also near impossible to achieve. The enforce such laws would require a lot of resources and tremendous amount of good will from the public. As such, stricter orders of using face masks and social distancing will have a negative effect on the fitness function.

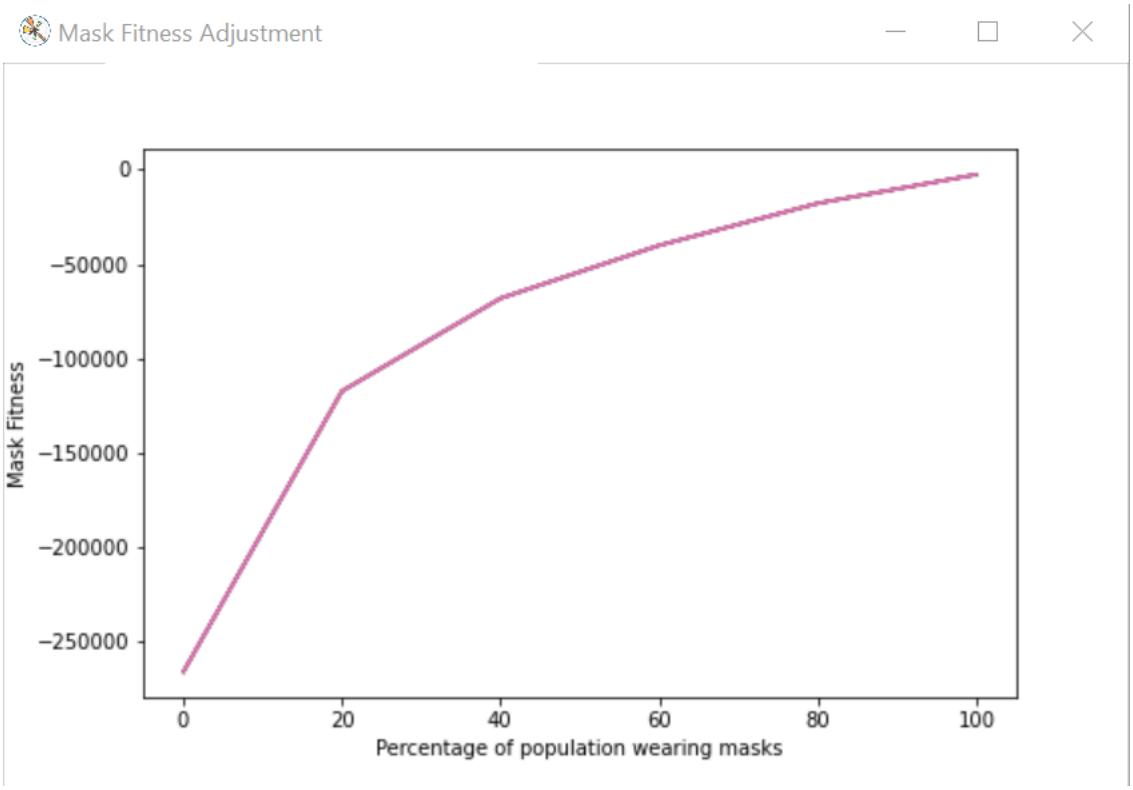

Figure 10- A figure showing the change in fitness function in relation to change in the percentage of the population wearing face masks. Social distancing is turned off. Higher is better

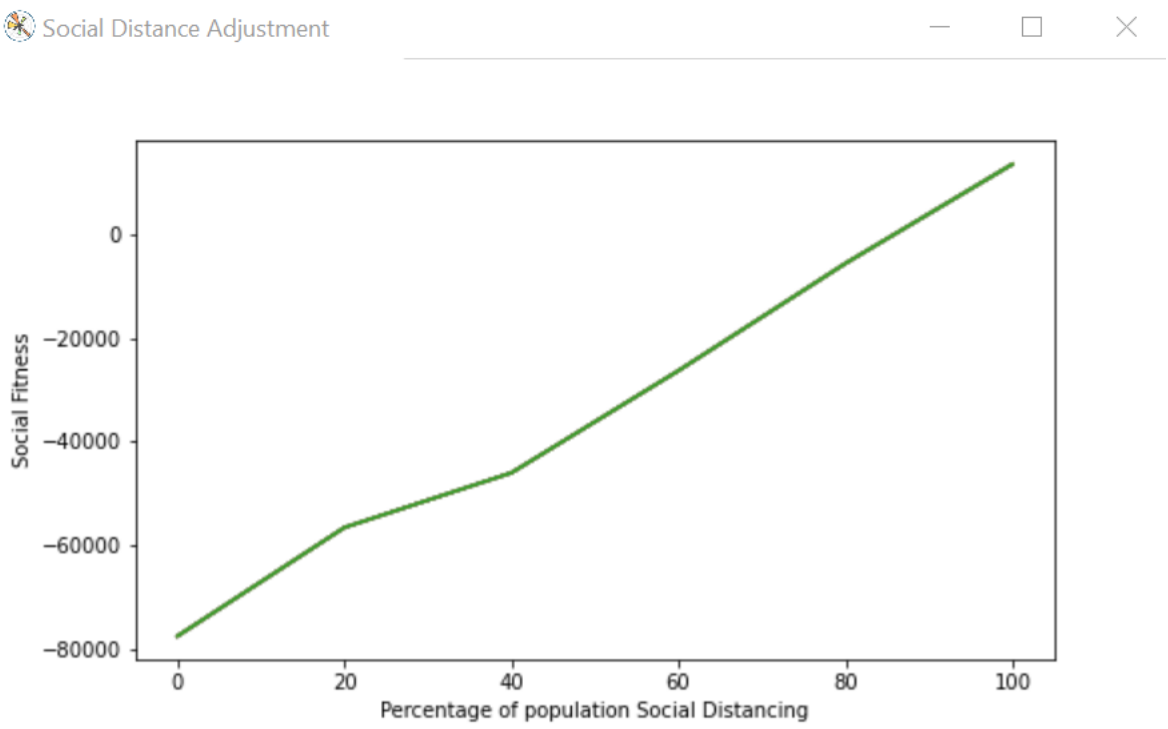

Figure 11 - A figure showing the change in fitness function in relation to change in the percentage of the population social distancing. Face mask usage is turned off. Higher is better

From the figures social distancing yields better results when it comes to lowering the infection spread rates, than just face mask usage and as such is more highly rated by our fitness function. While face mask usage significantly lowers the chance for the virus to spread, the gains get progressively smaller beyond 60% of the population participating and it gets harder to justify further investment. On the other hand, higher public participation in social distancing provides large enough of a benefit that it is worth pursuing even at the cost of growing inconvenience.

4. Results

In all three simulations we can see a clear pattern of how implementing the preventive measures (isolation, social distancing, mask usage) drastically reduces the number of infected in the simulated environments.

From the CA simulation (in the other simulations, isolation was always enabled) we can see that mandatory isolation for everyone showing symptoms had a major impact on the number of infected, going as far as reducing the number of infected by 45% when no other preventive measures were implemented. Having social distancing and mask usage mandatory for everyone also greatly reduced the number of people infected by approximately 40% when compared to no preventive measures whatsoever.

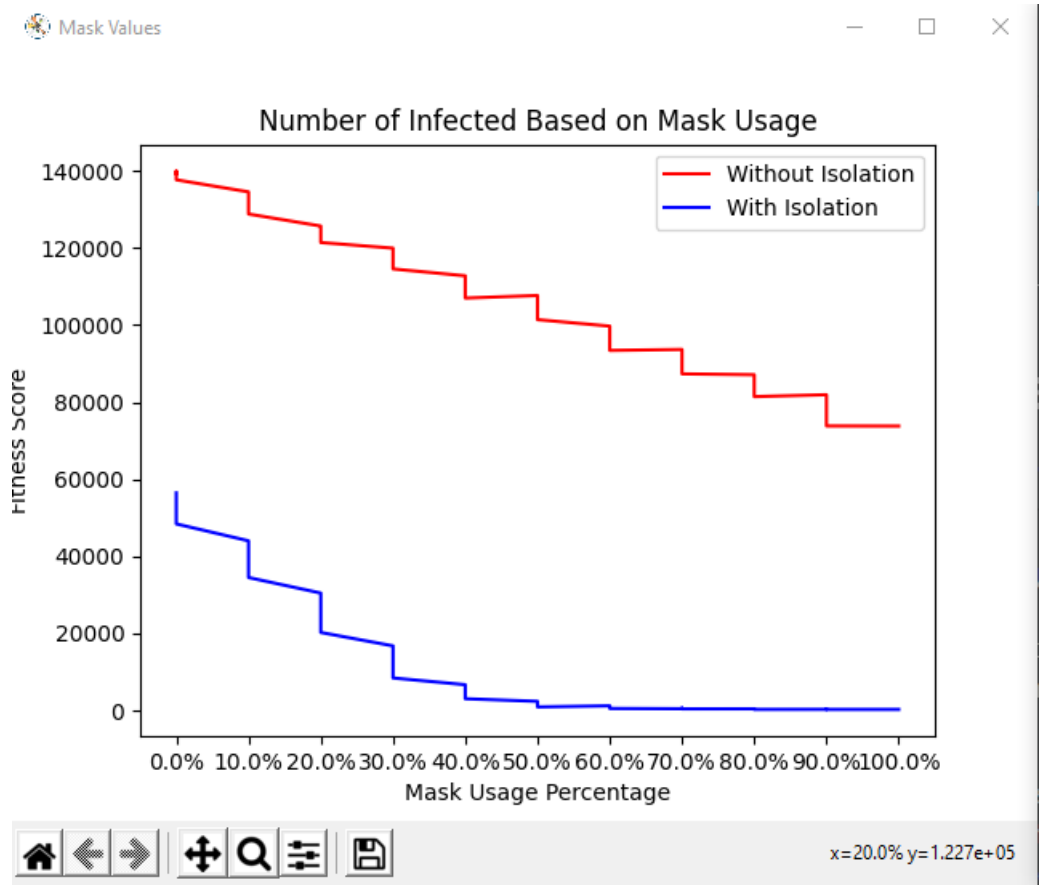

Figure 12: Graph of number of infected relative to how many percent of the population are using masks. Note how mandatory isolation drastically reduces number of infected.

In the Agent based simulation, we can see that social distancing is more effective at preventing infection than wearing masks, as social distancing completely eliminates chances of infection due to contact with other agents isn't possible. Mask usage however only reduces the chances of infection upon contact.

These results are of course very naïve due to the restrictions and lack of complexity in the simulations, but should be a good indication in regard to these measures playing a vital role in stopping the virus from spreading.

5. Conclusion

This project was intended for us to learn about the theoretical and practical aspects of complex system models, by implementing simple versions of Cellular Automata, Network and Agent models to produce somewhat realistic simulations. In that regard the project has been a success as we have learned a great deal about the strengths and weaknesses of working with each model in a standalone setting.

We consider the simulations to be good based on our lack of prior experience with complex system models, but have noted how a combination of the different models would result in more accurate simulations as each model (or our knowledge of how to work with them) by itself is somewhat restricted in terms of what can be simulated.

6. References

CDCP. (2020, October 27). *Interim Clinical Guidance for Management of Patients with Confirmed Coronavirus Disease (COVID-19)*. Retrieved from cdc.gov: https://www.cdc.gov/coronavirus/2019-ncov/hcp/clinical-guidance-management-patients.html

Dommerud, T., & Schwencke, M. (2020, September 4). *Folkehelseinstituttet har talt hvor mange som bruker munnbind i kollektivtrafikken. Dette fant de.* Retrieved from aftenposten.no: https://www.aftenposten.no/norge/i/9vjaXw/folkehelseinstituttet-har-talt-hvor-mange-som-bruker-munnbind-i-kollek

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.*

Giattino, C. (2020, August 24). *How epidemiological models of COVID-19 help us estimate the true number of infections*. Retrieved from ourworldindata.org: https://ourworldindata.org/covid-models

Krüger, L. (2020, March 22). *Bør eldre frykte korona bare fordi de er eldre?* Retrieved from nrk.no: https://www.nrk.no/norge/bor-eldre-frykte-korona-bare-fordi-de-er-eldre_-1.14940204

Kushman, R. (2020, July 6). *Your Mask Cuts Own Risk by 65 Percent*. Retrieved from ucdavis.edu: https://www.ucdavis.edu/coronavirus/news/your-mask-cuts-own-risk-65-percent/

Matrajt, L., & Leung, T. (2020, August). *Evaluating the Effectiveness of Social Distancing Interventions to Delay or Flatten the Epidemic Curve of Coronavirus Disease*. Retrieved from cdc.gov: https://wwwnc.cdc.gov/eid/article/26/8/20-1093_article

NIPH. (2020, November 5). *Daily report and statistics about coronavirus and COVID-19*. Retrieved from fhi.no: https://www.fhi.no/en/id/infectious-diseases/coronavirus/daily-reports/daily-reports-COVID19/

Osborne, H. (2020, October 5). *Woman With COVID Shed Virus for 61 Days in Longest Case of Its Kind Recorded*. Retrieved from Newsweek.com: https://www.newsweek.com/viral-shedding-covid-longest-period-contagious-1536387

Pellis, L., Ball, F., Shweta, B., Eames, K., House, T., Isham, V., & Trapman, P. (2015, March). *Eight challenges for network epidemic models.* Retrieved from sciencedirect.com: https://www.sciencedirect.com/science/article/pii/S1755436514000334

Ritchie, H., Ortiz-Ospina, E., Beltekian, D., Mathieu, E., Hasell, J., Macdonald, B., . . . Roser, M. (2020, November 3). *Mortality Risk of COVID-19*. Retrieved from ourworldindata.org: https://ourworldindata.org/mortality-risk-covid

Sayama, H. (2015). *Introduction to the Modeling and Analysis of Complex Systems.* Opensuny.

spk578. (2017, October 5). *Distance on a sphere: The Haversine Formula*. Retrieved from esri.com: https://community.esri.com/groups/coordinate-reference-systems/blog/2017/10/05/haversine-formula

Stamataki, Z. (2020, September). *Coronavirus reinfection – what it actually means, and why you shouldn't panic*. Retrieved from theconversation.com: https://theconversation.com/coronavirus-reinfection-what-it-actually-means-and-why-you-shouldnt-panic-144965

Statistics Norway. (2020, April 1). *Fakta om Befolkningen*. Retrieved from ssb.no: https://www.ssb.no/befolkning/faktaside/befolkningen

Stranden, A. L. (2020, March 24). *Hvem er egentlig i risikogruppen for korona?* Retrieved from forskning.no: https://forskning.no/sykdommer-virus/hvem-er-egentlig-i-risikogruppen-for-korona/1659901

Wang, X., Song, B., Wei, N., Liu, R. P., Guo, Y. J., Niu, X., & Zheng, K. (2019). Group-Based Susceptible-Infectious-Susceptible Model in Large-Scale Directed Networks. *Researchgate*.