The Use of Evolutionary Algorithms and Agent-Based Modelling in Decision Making for Tackling the Spread of Novel Coronaviruses Under Varying Population Density

Sara Wilson

2611955

Professor Carron Shankland & Dr David Cairns

October 2020

Submitted in Partial Fulfilment of BSc (with Honours) Computing Science

Computing Science and Mathematics

University of Stirling

Abstract

Problem:
As the Coronavirus pandemic begins to conclude, it is important to consider how best we can tackle any future pandemics, and hopefully react faster, save lives, and save money doing so. Many nations make extensive use of disease modelling technologies in conjunction with evolutionary algorithms to help them make suggestions to local or national governments on how they should best handle a pandemic outbreak. Within Scotland itself there is limited use of this technology when compared to national governments in England, which is far more densely populated; designing and implementing a disease model and appropriate evolutionary or genetic algorithm will allow us to simulate and discover some of the more effective solutions that could be used in this country to keep both infections and costs to a minimum.

Objectives:

- Design and implement a disease model using an appropriate program, which can simulate disease spread and mitigation procedures throughout a population, along with the ability to track and record infections, deaths, and other figures.
- Create a Java program which interacts with the disease model using an appropriate evolutionary algorithm to develop and refine a set of effective solutions, which keep the overall cost and number of infections to a minimum.
- Run extensive simulations of the program over a long period of time to allow it to refine the most accurate solutions possible within a reasonable timeframe.
- Examine and analyse the results gained from the program using an appropriate programming language or interface to generate graphs and charts.
- Consider the implications of the overall project in terms of its accuracy, what the results could be used for, its limitations, and what could be added or improved upon in the future.

Methodology:
The first stage of the project was to develop the disease model using the NetLogo program, by implementing the various parameters, sliders and graphs needed for an accurate model; this allows for detailed analysis of how the spread of coronavirus is affected by the different control methods and how strictly they are implemented, or when they are brought into effect.
The next stage was to create a Java program which runs the NetLogo model in headless mode, such that a genetic algorithm could be implemented to automatically run simulations of the model and try to come up with the most optimal set of solutions which keep the total cost of the procedures and the infections to a minimum.
The set of solutions found by the genetic algorithm was then analysed using the Microsoft Excel application for use in the results analysis and to conclude which setup of control parameters is most cost-effective at mitigating the spread and impact of coronavirus.

Achievements:
A disease modelling program has been created that can extensively simulate most aspects of coronavirus spread along with the various common methods that are used to mitigate it. This program works in conjunction with a genetic algorithm to run simulations of the system to generate the most accurate possible solutions, by changing the various parameter values and I/O statuses until it finds a set of solutions which keep both infections and financial costs to a minimum. This set of solutions can then be analysed using Excel to determine which ones are the most optimal.

Attestation

I understand the nature of Academic Misconduct as outlined in the University's Academic Integrity Policy and certify that this dissertation contains only original work undertaken by myself during the time of the project.

Signature Sara Wilson **Date** **16/04/2021**

Acknowledgments

I would like to offer my sincerest thanks to the following people and organisations for all their help in this project; without them this would not have been possible:
Professor Carron Shankland – Supervisor
Dr David Cairns – Supervisor and Secondary Marker
Sam Nelson – Technical Support, and for allowing me to use one of the lab PCs for installation of NetLogo.
Alex Avallone – For allowing me to look at their covid model for inspiration at the beginning of the project.
ccl.northwestern.edu – NetLogo and its associated documentation

Table of Contents

List of Figures

1 Introduction

The COVID-19 (SARS-CoV-2) outbreak has brought about unprecedented changes to everyday life and affected or taken away the lives of many people around the world. The long-term economic and educational damages are hard to estimate but there is no doubt that is has changed the way that nations consider their protections against such an epidemic. Unfortunately, coronavirus is several times more lethal than similar viruses, and many areas are ill-equipped to deal with such a large-scale outbreak whether due to tight budgets, overpopulation, or other similar factors.

For this reason, it is crucial to understand exactly how, and in what circumstances, coronavirus can spread most easily, and what can best be done to combat it. The use of a modelling tool is one of the best ways to achieve this, and in conjunction with appropriate evolutionary algorithms, the aim is to match the data as closely as possible with real-world figures, and as a result, draw accurate conclusions on which protection measures may be most viable under varying conditions of population density and public behaviour.

1.1 Background and Context

Epidemics, and indeed pandemics, are not a new thing to the world. They have been cropping up throughout human history, some far more fatal and wide reaching than others. The **Black Death** (Bubonic Plague) of 1346-1353 is perhaps the most famous of these, claiming the lives of between 75 and 200 million, including up to 60% of the European population. This outbreak was caused by a bacterium passed on by fleas riding atop rats from merchant ships arriving from Asia. Similarly, yet this time caused by a virus, was the **SARS-CoV** outbreak between 2002 and 2004, killing around 774 people worldwide.[1] It is this virus, which is of particular interest to this project, as it shares many close similarities with the SARS-CoV-2 virus, belonging to the same family.

While SARS-CoV was highly fatal but with a low transmission rate, SARS-CoV-2 has almost the opposite features; it is far less deadly, but far more transmissible. Despite the lower mortality rate, the rapid spread of the disease throughout the world has meant it has taken hold of multiple vulnerable groups such as those in care homes, retirement buildings and weaker family members, by means of transmission from more healthy, sometimes asymptomatic carriers. As a result, over 1.1 million people have died with coronavirus since the beginning of the outbreak, a far higher number than other similar, more deadly outbreaks.[2]

There are many techniques which can be used to curb the spread of coronavirus. These include but are not limited to; PPE, social distancing measures, imposed lockdowns, and self-isolation of those who are more at risk or suspected of infection. When government groups are attempting to decide which measures to take to tackle such an outbreak, it is important to have an accurate simulation of how such measures can affect the spread. NetLogo offers many facilities for creating detailed models of disease spread and this means that a model can be built to perfectly suit the needs of the research being tackled in this project with a high level of detail. Once the model is complete, implementing genetic algorithms is key to matching the simulations up with real-life data and therefore producing realistic results rather than rough estimates.

As the outbreak is slowly brought under control, it is important to consider what measures could be most effective in the event of a similar outbreak happening again. Outbreaks are more widespread in modern times due to the advent of better transportation systems, so it is vital to catch the virus early before it has a chance to take hold of a population. The nature of the world in the modern day means that spread of new diseases can be extremely rapid, and therefore stemming the spread early can prevent long term economic damage and loss of life, as is currently being seen with the SARS-CoV-2 outbreak; hence I consider this research of great importance and relevance to today's issues.

1.2 Scope and Objectives

The primary goal of this project will therefore be to help develop a model through NetLogo which as accurately as possible simulates the spread of a virus like SARS-CoV-2, and using various parameters and criteria, an accurate assessment can be made on which measures are most effective in the long term. This will be tested and refined using a genetic algorithm, which will help to determine which combinations of procedures result in the minimum number of deaths, with evaluations split across varying population densities, fitted to Scottish council area population density data. The results of the training will then be evaluated using the Microsoft Excel program, to generate the charts, graphs, and summaries that are needed to make a conclusive assessment on the outcome and therefore which procedures are most effective at curbing the spread of a novel coronavirus.

A summary of the main objectives of the project in a more simplified format are as follows:

- Using NetLogo, create a model which can as closely simulate the spread of SARS-CoV-2 as possible:
    - The model will include Boolean parameters for control methods, such as the use of PPE, social distancing, and imposed lockdowns.
    - Some parameters will have an associated slider to control the effectiveness of the measure, as some are subject to public behaviour and are wildly unpredictable.
    - Members will be able to enter and leave the population (albeit at a slow rate) which will simulate the movement of the public in and out of an infected area.
    - Members will have set homes which they will return to frequently; they are more likely to return the further away they are.
    - Slider values such as infectiousness, death rate and so on are based on current Scottish NHS data at the time of completing the model.
    - **Important to note about this process** is that the model will likely undergo an iterative process, as implementation of the evolutionary algorithms will expose new problems in the model to fix.
- Integrate the model with genetic (evolutionary) algorithms to generate accurate, trained data which can be used in the evaluation process.
- Evaluate the collected data using Excel, to produce the information and formats needed for thorough analysis.
- Final assessment of the datasets during the final dissertation.

When analysing the final datasets, of particular importance to me will be how low the death rate can be kept for as little "cost" as possible, and across varying levels of population density.

The hope is that this research will be able to produce a model which would be usable in the future by governmental research teams to decide the best course of action for dealing with any future outbreaks. By matching to a certain area's population density data, a set of usable results could be obtained for that team to use for planning their local outbreak response. Through use of Excel, the data retrieved from simulations can be cleaned and analysed to produce outputs which are more straightforward to understand by the layperson than standard spreadsheets or raw data.

1.3 Achievements

This section will be a brief overview of whether the specific objectives of the project were met – and in cases where they were not met; what could be done to improve were the project to be repeated.

- Develop a disease model using an appropriate modelling tool that simulates the spread of novel coronaviruses under typical urban conditions within Scotland, with close reference to Scottish coronavirus data regarding average incubation time, death chance among age and sex groups, immunity time, and so on:
    - The model is extensively fleshed out to include most of the typical disease spread behaviours, such as propagation among closer individuals, increased chance of death with age, standard deviations on infection and incubation times, and so on – it is impossible to model every possible part of disease spread however, so other parameters that may affect spread such as wind speed and direction, air temperature and so on are not modelled. This could be added in a more detailed model but there are virtually infinite ways two individuals could interact or become involved in a disease's spread that it is extremely unrealistic.
- Incorporate a variety of control parameters to the model to mitigate the spread of the disease:
    - Six control parameters of the most common types that were put into place to control coronavirus spread – each with a compliance level and introduction threshold (% of individuals infected) slider. As above, there is a lot more potential for adding further control parameters such as keeping those entering the population when infected, isolated for a certain period, as with those returning from holidays abroad and needing to quarantine. Again, there are really a limitless number of parameters that could be added to the model; however, I feel the state of the model is sufficient and fit for purpose regarding the project.
- Integrate the disease model with an evolutionary or genetic algorithm which searches for the most cost-effective solutions with respect to infection cost, procedure cost, and computation time using a realistic cost function:
    - Evolutionary algorithm works as intended by using the cost function in conjunction with multiple generations of sets of chromosome populations, passing the more "correct" solutions to the next generation through evolution.
- Develop an appropriate and working cost function for the algorithm to use as a point of judgement for which solutions (parameter setups) are the most optimal, and should carry on to the next generation:
    - Cost function works as expected and is balanced such that both the i and m parts of the equation do not outweigh one another, meaning changes on either end have an equally valuable impact on the overall cost – the genetic algorithm is able to use this effectively to discover good solutions while also willing to look at "poorer" solutions which may be more optimal in the end. Further refinement of the cost function could help it to become more realistic

    and accurate by both including more aspects in the function itself and using costs which are more in line with those seen in the real world.

- Test and refine both the model and the algorithm to keep execution time low, thereby increasing the number of runs possible, and data collected, within the project timeframe:
    - The execution time of the NetLogo model was cut down from around 10 seconds for a 5-run average, to around 5 seconds, by tidying up the code and removing most unnecessary or redundant parts of code. Much of the speed of the evolutionary algorithm depends on how slow the NetLogo model is, and with most model execution time going towards the tracking and updating of everyone's status, the only way this could be lowered was by decreasing the population from 500 to 400, which was found to be a healthy balance between execution time and accuracy of results. It is possible that these times could be further improved with more robust code design and optimisation using more advanced techniques, but due to time constraints and a limited understanding of the backend of NetLogo, this was not possible.
- Gather and analyse the results from the evolutionary algorithm using an appropriate data analysis tool to generate charts, graphs, and figures for the purpose of deciding which of the set of "best" solutions found is most optimal:
    - The original aim was to make use of R and RStudio for data analysis, but due to both a limited understanding of R, a lack of confidence, and shortage of time, the switch was made to Excel instead, which still provided much of the functions needed for the final data analysis.

2 State-of-The-Art

2.1 NetLogo

NetLogo is a programming environment in which the user can construct, and run simulations with, detailed agent-based models using a variety of facilities.[3] It was developed at the *Center for Connected Learning and Computer-based Modelling* at *Northwestern University* in 1967 and uses a programming language **Logo3**, which heavily draws from the **Lisp** family of programming languages. It is simple to understand the basics of, largely due to the friendly nature of Logo3, but still provides the more advanced user with the ability to develop extremely complex and detailed model with relative ease.

Logo3 is an interpreted language and as such executes lines of code one at a time as they are written, without having to compile the code into a program or executable, for example. It is also modular, which allows for a wide array of extensions and add-ons to be imported into a model's framework, further expanding the possibilities and level of detail that can be achieved when constructing models.

NetLogo makes use of three primary entity types which are involved in modelling:

- The **environment**, which is simply a grid of a x b square tiles, used to project the simulation onto, such as the movement of turtles and changes of state. They are also used to track the agentset of turtles near a specific one; that is, the turtles in the squares immediately surrounding the turtle (which can be used for further analysis, i.e., spreading disease to members of the agentset).
- **Mobile agents**, also known as turtles, which move and interact with the environment and react to changes in variables or conditions.
- **Links**, which are dynamically created between turtles, and as mentioned above, primarily includes the agentset of nearby turtles but can also incorporate other aspects such as two turtles being partnered together.
- In addition, it also incorporates the **observer**, or the user themselves, who creates, controls, and monitors the simulation and its associated features.

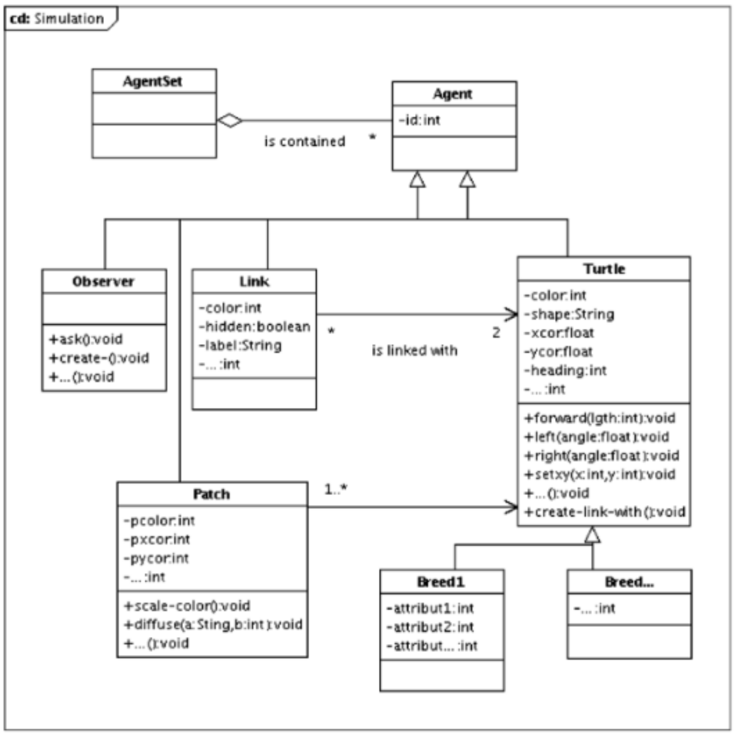


Figure 1. An example metamodel of a NetLogo model, showing the links between classes and the interactions between them.[3]

Figure 1 shows that there is a simple but effective structure to a NetLogo model; of note is the fact that there can be many turtles in one “Patch” (that is, the individual squares of the environment), and the fact that turtles(members) support inheritance, allowing us to have multiple subsets of members, for example, a subset containing all the isolating turtles, and some unique properties for them. Also, worth noting is the fact that the environment can be **continuous**; the final column of squares is “next to” the first column, so if a turtle moves off the right-hand side they will appear on the left (similar to how the old arcade game Pac-Man works).

There are 3 tabs in the NetLogo interface; “**Interface**”, “**Info**”, and “**Code**”. The **Interface** tab provides the facilities for structuring the model, allowing the user to insert buttons, slider, switches, graphs and so on to help construct a dynamic and changeable model. The **Info** tab offers the facility for documentation of what is involved with the model, instructions on how to use the model, what choices were made, and an area for credits and references to other work. Finally, the **Code** tab is where the bulk of the programming work is done; building on the content introduced in the Interface for the model, the user can take advantage of the interpreted nature of the language and pull values from the interface in real time, mid-simulation, which provides an extra layer of usefulness to the program by limiting the need for constant compiling of code and allowing variables to be changed while the simulation is running.

There are plenty of import and export features that can be used to work with existing datasets or export the created ones to another format for analysis, respectively.

The development of a NetLogo model revolves around defining several key elements:

1. The model’s **structure**, involving the creation and definition of the global variables along with the environment and the turtles within it (including their local variables).
2. An **initial state**, taken from the status of variables in the Interface tab, set by the user.

3. The **behaviour** of the agents and the environment; this is the bulk of the code and controls the flow of the simulation up to a specified endpoint.
4. And the model's **outputs**, through raw data, graphs, visual feedback, and so on.

Also worth noting is that the use of **GIS** (Geographic Information System) data is supported through the **extensions [gis]** package; this allows for mapping data to be fed into a simulation and could be used in a variety of ways, such as using a heatmap of a country's population to distribute turtles relative to those values, which could be useful in a disease-spreading simulation in order to fit simulations to a specific region, without having to hard-code the positions and behaviours of the turtles. Much more information on the features and uses of NetLogo can be found at the software's website, and this information was crucial in helping to develop this project's model to its fullest and avoid any potential pitfalls or mistakes.[4]

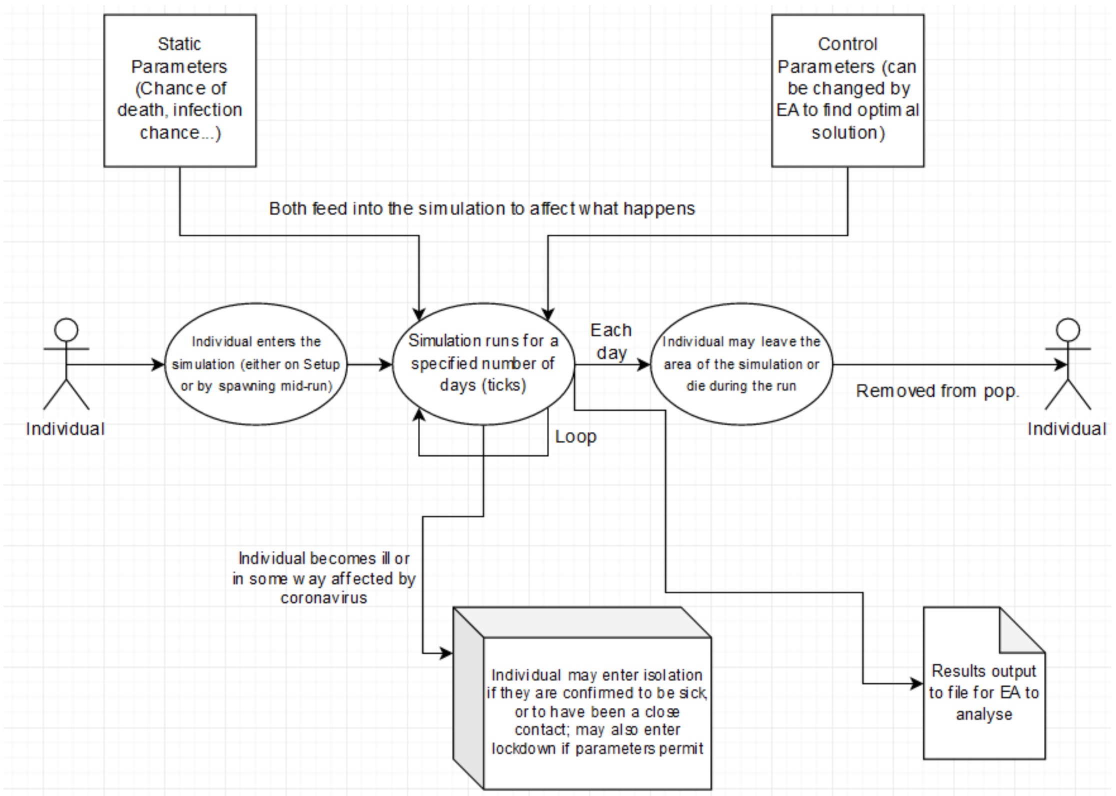

Static Parameters (Chance of death, infection chance...)
Control Parameters (can be changed by EA to find optimal solution)
Both feed into the simulation to affect what happens
Individual
Individual enters the simulation (either on Setup or by spawning mid-run)
Simulation runs for a specified number of days (ticks)
Each day
Individual may leave the area of the simulation or die during the run
Removed from pop.
Individual
Loop
Individual becomes ill or in some way affected by coronavirus
Individual may enter isolation if they are confirmed to be sick, or to have been a close contact; may also enter lockdown if parameters permit
Results output to file for EA to analyse

Figure 2. A simple diagram showing the general flow of the program, and a turtle (entity)'s typical pathway through a simulation instance.

NetLogo was chosen for this project as it makes for a perfect environment in which to construct a model of disease spread and mitigation methods; there are a few pre-built models which can serve as a foundation for building a much more complex model, and as mentioned earlier, it is relatively straightforward to get a grasp on the language itself and begin building more advanced systems. The ability to see visual feedback on the performance of the model using graphs and the environment window itself means that testing and refining of the model becomes much simpler than through comparative non-visual environments. The use of GIS data in the model is simple to implement and as a result means that, while this project will focus on population densities in Scotland, it could theoretically be used anywhere simply by swapping out the GIS data used.

2.2 Evolutionary Algorithms

Evolutionary algorithms are based upon the theory of evolution proposed by Charles Darwin. There are 2 primary points:[5]

- Populations are exposed to environmental pressure, which results in natural selection, where only a select group live on to create the next generation.
- This is known as survival of the fittest and causes an overall increase in the average fitness of the group, or their ability to survive the environmental pressure on them; in other words, their level of adaptation to the pressures experienced.

In terms of modelling, costs and cost functions are used to simulate the environmental pressures – lower costs are more preferred. A selection of the fittest population is used in the next generation for the creation of the new population of entities; this is achieved through use of the fitness function, which must be carefully designed according to the model and the desired outcome to pick the most ideal candidates to solve the given problem.

According to Câmara and Soni, [5], [6] the general steps involved in the execution of an evolutionary algorithm are as follows:

- **Initialization** – an initial population is created; it is typically random but in cases where there is a general expected outcome the population can be created manually with this in mind. It is crucial to have a wide range of solutions available as it represents a large and varied gene pool rather than a homogenous group.
- **Selection** – members are evaluated according to the fitness function being used; as mentioned earlier this function is heavily dependent on the data and environment being worked with and must be thoroughly developed. Using the function, the fitness of all members of the population are evaluated and a portion with the highest fitness are selected to create the next generation. There are two subprocesses involved with this stage:
    - **Crossover** – new individuals in the population are created, based on the idea of reproduction, by combining genetic material (or properties) of the selected individuals; this should result in a higher fitness than the previous generation.
    - **Mutation** – to introduce randomness to the system, small random changes to a new entity's genome (or features) are made; if the change does not increase fitness then it will be removed by the selection process. Without mutation, every combination reachable in the system would already be present, and as such mutation is a crucial step to focus on.
- **Termination** – the end stage of the algorithm, typically when a maximum runtime has been reached, there is a lack of resources such as time and money, or a performance threshold has been reached – the point at which new generations fail to increase the fitness by a certain value per cycle. Again, this is set by the programmer and as such requires careful assessment.

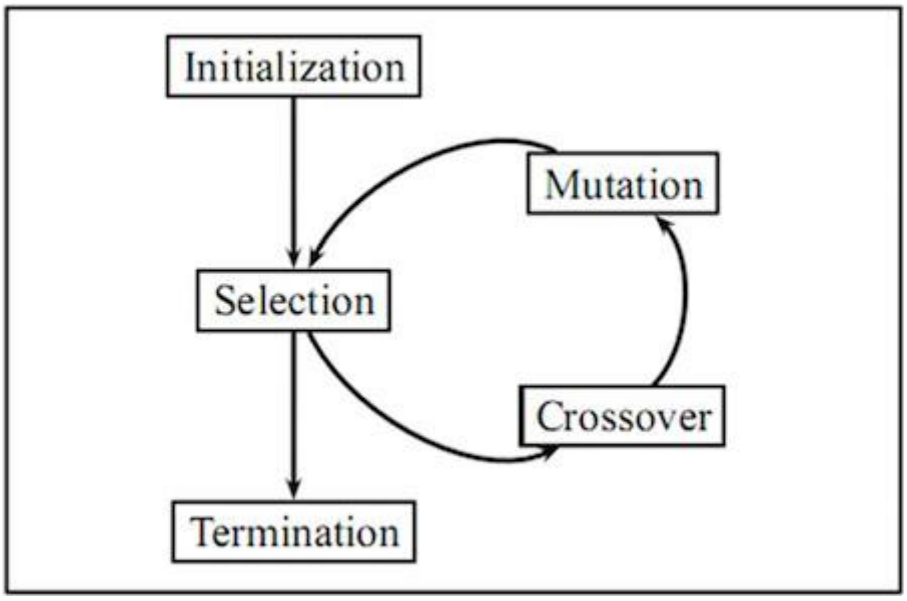


Figure 3. Diagram showing basic process flow in an evolutionary algorithm.[6]

It is possible to use multiple fitness functions in an evolutionary algorithm. Using more than one results in a set of optimal solutions, known as the **Pareto frontier**. The frontier contains the set of solutions that are equally optimal; from there it is usually the role of the decider (the individual running the tests) to select which solution is the most optimal by taking into consideration the desired outcome and the context of the problem. While the algorithm can generate this set of possible solutions, it cannot pick the most effective solution; some solutions are more viable in certain situations, for example, it may not be economical to invest heavily in test and trace methods if the R number can also be brought below 1 by introducing social distancing, for example. In this situation it is up to the decider to make an informed decision, but the algorithm has at least cut the number of solutions down to a manageable amount that this process is relatively straightforward.

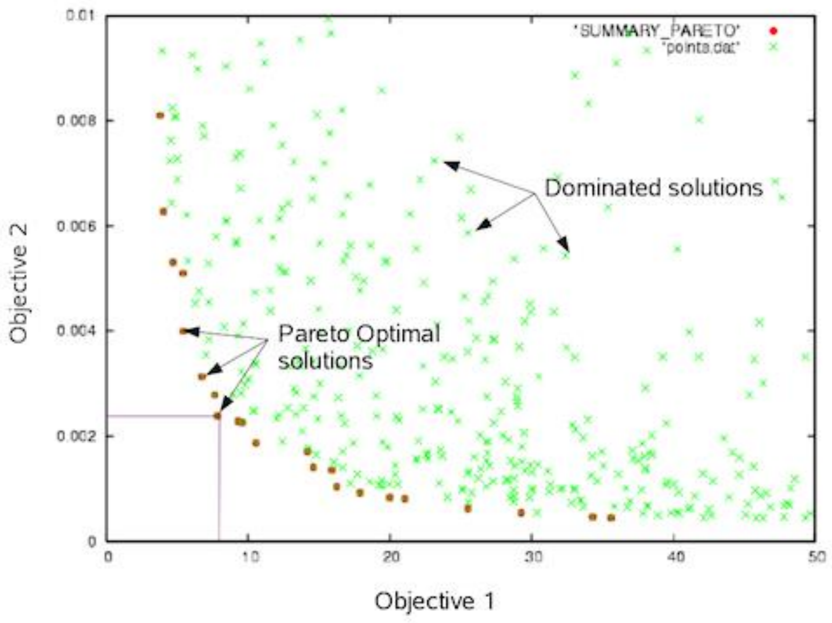


Figure 4. Example scatter graph of solutions, with Pareto frontier solutions highlighted.[6]

There are a few limitations of evolutionary algorithms that it is important to be aware of:[5]

- Sometimes it is not always obvious whether the solution is a **local** or **global extremum**; in the case of a local extrema, it may not be the best solution, simply the best for that scenario.
- Evolutionary algorithms are far less effective if there is a loss of **diversity**; as a result, the fitness function must be able to evaluate both the **solution** and the **problem** accurately to make correct decisions about which members to keep. Similarly, mutation must be correctly set up to avoid becoming stuck in local extrema and being unable to get accurate results.

In both above cases, making use of **hyper-mutation**, or a period of extreme mutation above the norm, could help to break free of the local extrema, but this is not always possible.

Overall, evolutionary algorithms make for an effective method of retrieving data from a model; provided we match the model to real-world data, in this case population density, the evolutionary algorithm will be able to find the optimal combination of measures and variable values to keep the deaths to a minimum, and at the lowest cost.

2.3 Previous Research in This Area

Before proceeding with the project research, it is important to identify and discuss the work of others in the area, to identify a sector which either has not been touched or can be expanded upon. In this section there will be some brief analysis of related research performed by others, and an identification of where this project can fit in among other research.

Firstly, it is worth considering the appropriateness of agent-based models in such an investigation, as perhaps others may be more effective. An analysis performed by *Cuevas*[7] in 2020 suggested that agent-based models are much more effective than mathematical models when simulating disease spread, as the latter considers the population to be homogenous, or having identical characteristics, whereas the former can incorporate a wide variety of characteristics and variables to suit the needs of the investigation. It points out that to have an effective agent-based model, careful attention must be paid to every aspect of the model, from the variables and parameters used to the testing and refinement performed to ensure there are no unusual results. In addition, it must be kept in mind that more complicated models are as a result more computationally intensive than simpler ones, so optimisation is important.

A similar investigation by *Maziarz et al*[8] in 2020 points out that it is impossible to accurately model every single aspect and as a result, simulations are never truly 100% accurate, rather a fairly accurate estimate of the expected outcome. During the project this must be considered when matching to real world data; working to within a 95% CI would be expected. This paper also identifies that the best agent-based models of disease spread use a combination of spread and control simulation, as is the goal with this project.

Another important aspect to consider is the effectiveness of control methods on the spread of COVID-19. Research conducted by *Chu et al*[9] and published by *The Lancet* on June 1st, 2020, after researching 172 observational studies from 16 different countries, found that physical distancing of one or more metres reduced the risk of infection by around 10.2%, compared to standing near an infected person; this reduced risk was found to steadily increase as the susceptible and infected patient were further away from one another. Similarly, the use of face masks and eye protection reduced the risk of infection by around 14.3% and 10.6% respectively, with the most protection being offered by industry grade masks such as N95 respirators.

In a related field, research performed by *Davies et al*[10] on June 20th, 2020, also published in *The Lancet*, concluded that with no prevention or control methods to mitigate the spread of COVID-19, there would be a projected total of 23 million cases and 350,000 deaths in the UK by the end of 2021. The R number, or the average number of new cases generated by a single person, could only be brought below 1 through use of lockdown periods – none of the other prevention and control methods were sufficient to prevent the spread of the virus, even in combination with one another. This paper suggested that with the strongest possible lockdown procedures the UK could expect around 120,000 total cases, and 50,000 total deaths, and concluded that frequent periods of lockdown would likely be necessary to prevent the NHS from exceeding capacity.

Taking both papers into account, it is not entirely clear exactly which combination of control methods would offer the lowest risk of spread and keep deaths to a minimum; the first paper suggests that many control methods are roughly equally as effective at controlling the spread, while the second suggests that only lockdowns can push the R number below 1 and help eliminate the virus. Hence, I believe this project will be useful in plugging this gap by identifying which methods are most effective, and at the lowest cost (in terms of both computational time and financially, with respect to infections, deaths, and procedure costs), to suggest the most viable course of action.

One paper of particular interest comes from the Proceedings of the Royal Society, published by *Handel et al*[11] in 2007. It concludes that there is a certain balance which must be struck between too weak and too strong, in terms of control measures. If measures are too weak, it can result in many deaths as the infection sweeps the population, but also a large amount of immunity build-up. On the other hand, measures which are too strong, for example introducing a lockdown too early in the outbreak, can mean that almost none of the population gain immunity, and when emerging from lockdown, the infection will be able to take hold very quickly. Therefore, the optimal method for controlling disease spread needs to strike a balance between infections, deaths, and immunity build-up while keeping the number of susceptibles as low as possible. This balance is what will result in our Pareto Front mentioned earlier, as when plotted on a graph, we will be able to see a range of solutions, some which keep the cost low and infections fairly high, and vice versa, and some which may strike that right balance between the two, which would be the most likely candidates for a chosen "best solution", although there is no true "best" solution, it is merely based on the judgement of the individual or professional body analysing the results.

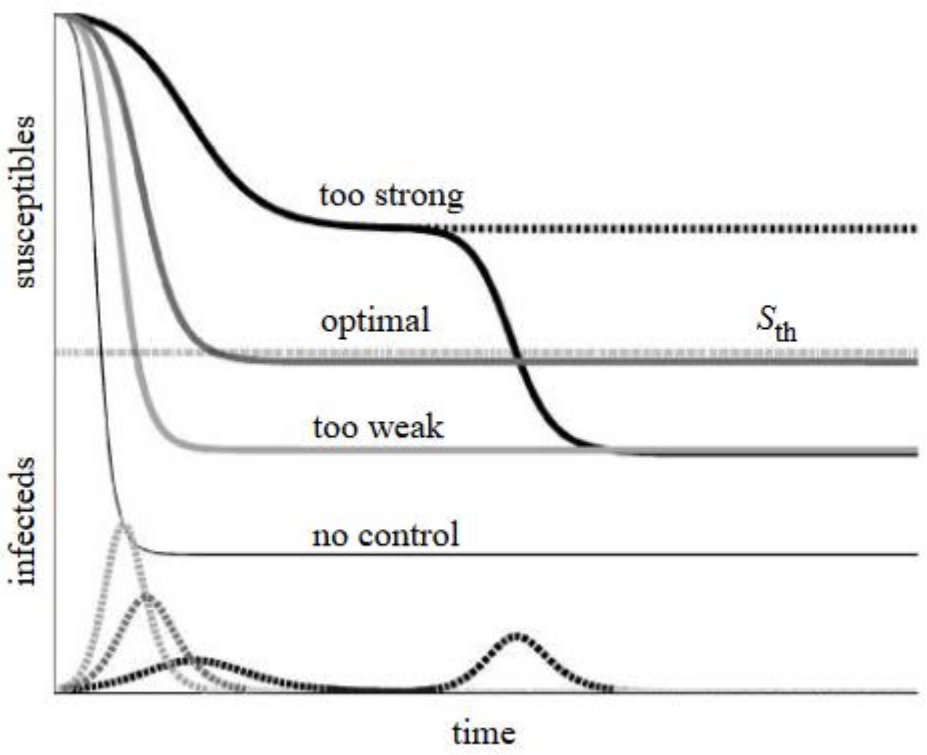


Figure 5. Graph showcasing the change in total cases with a weak, optimal, and strong approach.[11]

This paper also highlights that it is important to consider the willingness of the population to comply with control measures, and this is something which should be considered in the model; not least because of the stochasticity it introduces, but also because of the accuracy in simulating real-world behaviours; indeed, there are a large group of individuals who were seen to be unwilling to comply with measures, and the number has only increased as the pandemic has continued.

Finally, a *BMJ* paper written by *Sridhar & Majumder*[12] and published on the 21st of April 2020 assessed that some global governments throughout many recent pandemics have become overly reliant on models for planning a response. It points out that given the constantly evolving nature of COVID-19, there is a continual need for new and more accurate models, and as a result it is crucial to consider historical data of previous related outbreaks when creating the model or making judgements. There are many factors which could not be modelled in this project, including long-term economic and social damages, job loss, homelessness, and so on; so as mentioned earlier it is impossible to create a perfect model and this must be considered during the project analysis.

While all the papers investigated have incorporated or researched in some manner the work done in this project, none have combined these factors in conjunction with an investigation on how population density affects the spread of disease, and the effectiveness of control measures. As a result, this investigation fits into a new niche that has not yet been considered with regards to COVID-19 response, and in those areas which have been investigated, the model from this project could help to expand upon them – as touched on earlier, the manner in which this model is constructed should lend itself well to matching with data from other countries meaning a possible use in the future if the virus resurfaces, or as suggested, it becomes endemic similar to seasonal Influenza.

It is useful to quickly touch on other potential methods of parameter estimation besides the use of a genetic algorithm, although it is not directly related to the project. One important example is the use of the “least squares” method.[13] Here we wish to identify the difference between the observed data and the figures generated by the biological model. We then search through all the possible values of the parameters (slider and button values) to find the setup that keeps this difference to a minimum. By calculating the sum of all the squared errors, or differences between the observed and model data, we can plot the sum of squared errors against various values of β which are plugged into the model, to identify which setup gives us the lowest sum of squared errors. This can be repeated in much the same way as an evolutionary algorithm to focus in on an “ideal” setup. As this did not offer a significant advantage over something more familiar like the evolutionary algorithm, this method of parameter estimation was ignored, but it is good to be aware of its existence.

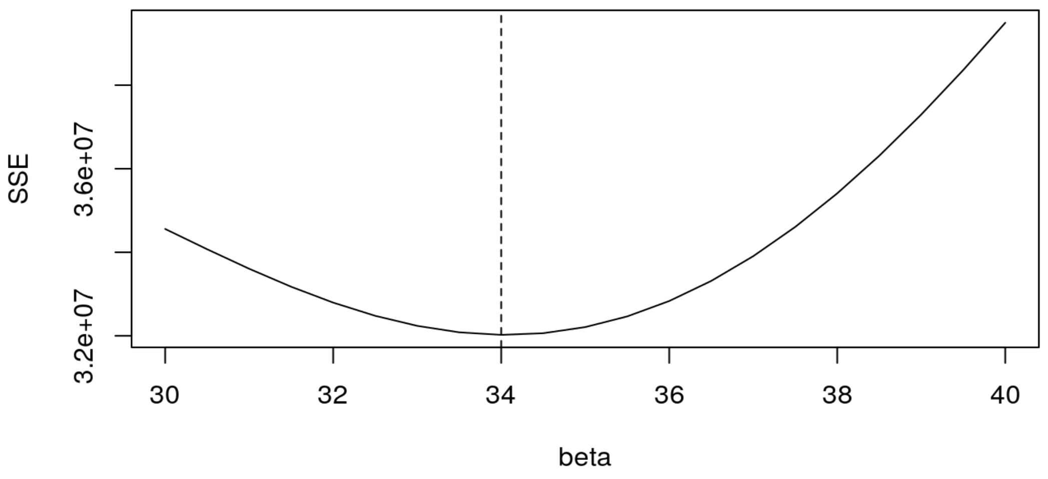


Figure 6. Graph of β value for the model against sum of squared errors – 34 is the most optimal value in this case. [13]

We can also use feature-based parameter estimation[13], whereby we look at the observed data to find notable “landmark” features of the data and change the model’s parameters until we get the same behaviour. This is considerably easier than the least squares method, but also less accurate, particularly when dealing with a wide range of variables; we cannot accurately judge what changes in the observed environment caused the landmark features, and so we could end up creating a model that is completely off from real-world behaviour.

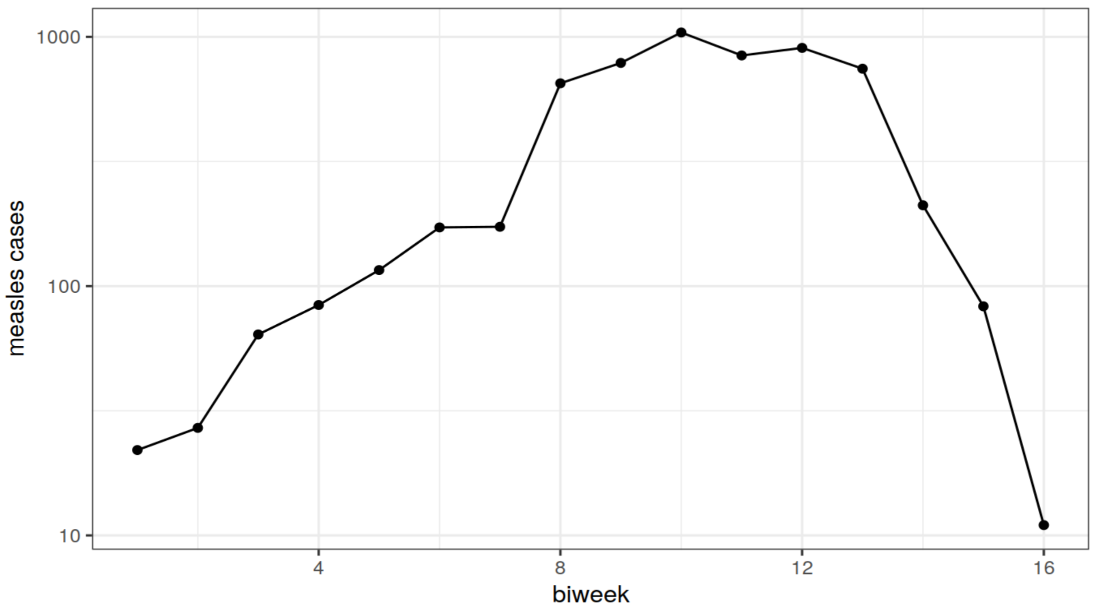


Figure 7. Example line graph highlighting some of the features we may want to match to – the peak at biweek 10, sharp increase at biweek 7, and so on. [13]

3 Problem description and analysis

This section will go into greater detail about the process involved in this project, including the various techniques used to manage the flow of work and remain focused on the goal.

The primary objective of this project is to design and create an agent-based model which is capable of simulating both the spread of COVID-19 and the effectiveness of control measures on limiting the spread. This model will feature an extensive set of parameters and variables which allow for full control over the simulation, and to produce results that are as representative of real-life behaviours as possible. It will then be matched against real-world data of population density in Scotland by council area to investigate how changes in such a metric impacts the spread and control of COVID-19; results obtained from the simulations should closely match up with those of Scottish Health Service data for each council area. Evolutionary algorithms will be used to produce the most optimal solutions at the lowest cost using the model. This 'cost' will be used in a cost function for the evolutionary algorithm, where a lower cost indicates a more optimal solution; it must consider the cost of allowing members of the population to become infected, as well as the (likely a proxy) financial cost of implementing control procedures, as well as how strictly they are enforced, i.e., compliance. Once the evolutionary algorithm has completed the search for optimal solutions, these will be thoroughly analysed using Microsoft Excel to produce detailed and accurate charts and graphs for use in the analysis and conclusion stage. In an ideal scenario each stage would take place one after the other, however this is almost always impossible, as through implementation of the genetic algorithm, flaws and problems with the model will be uncovered, and as a result the project is likely to follow an iterative development process of creating the model, adding the genetic algorithm, and analysis with Excel in several cycles.

The results obtained from this project will be presented in an academic sense, but in a real-world scenario would be presented to policymakers and other governmental officials, in conjunction with similar research conducted by other organisations around the country, to help them to make informed and detailed decisions about how best to tackle the spread of a pandemic, should a similar outbreak occur in the future.

3.1 NetLogo Model

3.1.1 Plan

The model is the core component of the project and will contain a wide variety of features to generate results as reflective of real-world scenarios as possible. It follows one of the more common types of disease models used, the SIR model. In these models there are three primary categories of member (or turtle as they are known in NetLogo):

- **Susceptibles** who are vulnerable to infection,
- **Infecteds** who are currently ill or infected with the virus,
- and **Recovereds**, who have been infected but are now free of it (and almost always immune).

There will be several environmental variables within the model, primarily split into two categories, those which will not be able to be changed by the evolutionary algorithm (i.e., not involved in the cost function), which includes sliders such as those for average recovery time, chance of death, and incubation period, and the other category, which will be changed by the algorithm when creating and testing populations, those being the buttons and sliders for the six control methods:

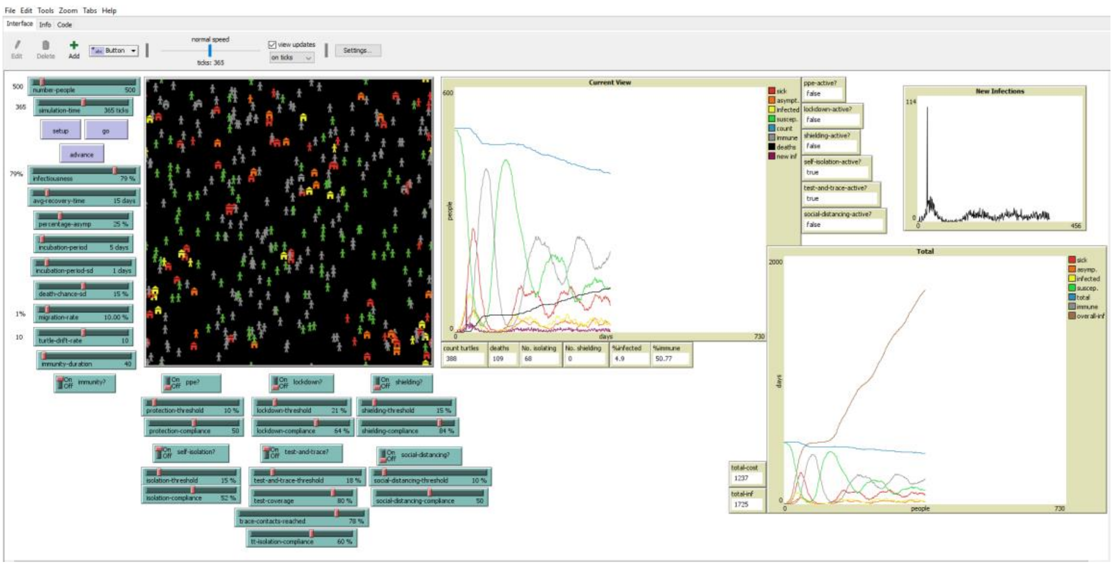
- **PPE**,

- **Test and Trace**,
- **Social Distancing**,
- **Self-Isolation**,
- **Shielding**,
- and **Lockdown**,

along with their compliance and threshold sliders.

The model will also feature some ‘monitor’ and ‘plot’ widgets for tracking of data, such as the trend of infections throughout a simulation run, and of course also to check that every aspect of the program is working as expected. It is also useful to include the ability to control the number of days (ticks) the model runs for, as well as the size of the population we would like to work with; these will both be handled with sliders too.

3.1.2 Implementation



ticks: 365

number-people 500
simulation-time 365 ticks
infectiousness 79 %
avg-recovery-time 15 days
percentage-asymp 25 %
incubation-period 5 days
incubation-period-sd 1 days
death-chance-sd 15 %
migration-rate 10.00 %
turtle-drift-rate 10
immunity-duration 40
immunity? On Off
ppe? On Off
protection-threshold 10 %
protection-compliance 50
lockdown? On Off
lockdown-threshold 21 %
lockdown-compliance 64 %
shielding? On Off
shielding-threshold 15 %
shielding-compliance 84 %
self-isolation? On Off
isolation-threshold 15 %
isolation-compliance 52 %
test-and-trace? On Off
test-and-trace-threshold 18 %
test-coverage 80 %
trace-contacts-reached 78 %
tt-isolation-compliance 60 %
social-distancing? On Off
social-distancing-threshold 10 %
social-distancing-compliance 50

ppe-active? false
lockdown-active? false
shielding-active? false
self-isolation-active? true
test-and-trace-active? true
social-distancing-active? false

count turtles 388
deaths 109
No. isolating 68
No. shielding 0
%infected 4.9
%immune 50.77
total-cost 1237
total-inf 1725

Figure 8. Screenshot of NetLogo model after a typical 365 day run.

The model includes a few Plot screens along with several Monitors to keep track of the current and final status of the model, and for easier analysis than by simply looking at flat values. One plot keeps track of the current situation of the simulation, whereas the other represents the total overall status; that is, it includes individuals which have moved out of the simulation or died at some point. Six disease control variables are included in the model, all of which are operated as a Boolean Switch with associated threshold at which the measure is introduced, as well as an additional slider to control the compliance level, such as what percentage of the population wear PPE when requested. Turtles can move in and out of the simulated area and as such new turtles can reintroduce the infection to a previously healthy simulation, adding a degree of stochasticity to the model. In that respect, there are several disease-related variables too, such as the infectiousness, probability of survival, average recovery time, and incubation period. All the above variables in combination serve to provide a detailed and accurate simulation of real behaviours which can be tweaked in line with current NHS figures.[16] None of the variables seen along the left-hand side of the model can be changed by the genetic algorithm and as such only the six control variables can be changed to try and find an optimal solution. Variables along the left-hand column of the model and their values are as follows:

Name	Default Value	Lower Bound	Upper Bound
number-people	400	0	5000
simulation-time	365	0 ticks	730 ticks
infectiousness	79%	0%	99%
avg-recovery-time	15 days	0 days	99 days
percentage-asymp	25%	0%	100%
incubation-period	5 days	0 days	100 days
incubation-period-sd	1 day	0 days	8 days
death-chance-sd	15%	0%	30 days
migration-rate	10%	0%	90%
turtle-drift-rate	10	0	20
immunity-duration	40	0	1000

Turtles are spawned into the environment at random, and at a random age and sex, and a set percentage are made infected, with the rest healthy. The outcome of this randomisation is that the stochasticity of the model is greatly increased, and more accurately reflects a real-world population with varying ages, sexes, and locations, rather than one homogenous group. As a feature of the model, an individual's age and sex has a direct impact on their risk of death from infection; the age and sex spread of the population is proportional to the most recent census data[16] for Scotland, broken down into several age group categories for ease of processing. Each tick, the turtles pick a random direction to move one square. As the outbreak progresses, infected individuals will eventually become sick after the incubation period, with a chance of becoming asymptomatic. Healthy individuals are considered susceptible to infection, and when they encounter a sick turtle, they have a chance of being infected, based on the value of the infectiousness variable. If turtles recover, they become immune (unless immunity is disabled) and this makes up the general flow of the simulation; control measures such as lockdowns and PPE will minimise the risk of transferring the virus and therefore reducing deaths. There is also a component to keep track of the homes of individuals; as turtles move further from their location, they are more likely to return home, meaning that individuals interact with their "neighbours" more often than other turtles in the simulation. An analysis from *Paul E. Smaldino*[14] illustrates the impact of movement radius on disease spread via his own simulations:

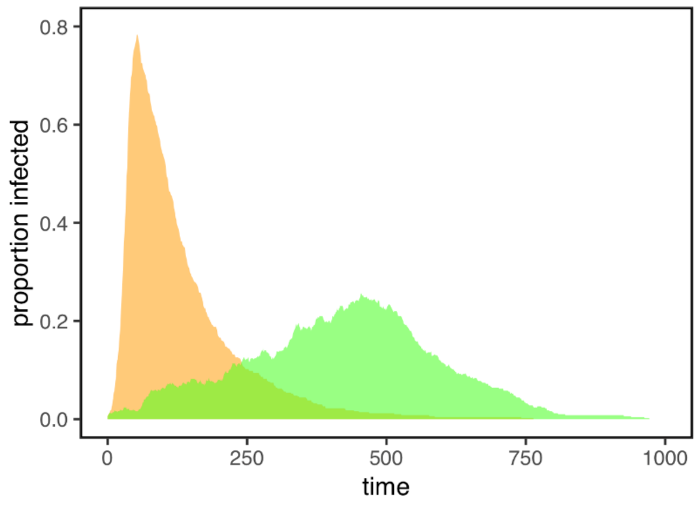


Figure 9. Graph illustrating the difference in infections between a highly mobile (orange) simulation and a less mobile (green) one.[14]

To go into more detail about some of the aspects of the model; the colours used in the display of individuals (turtles) and houses (for those isolating) are the same as seen in the graphs on the model – red for sick patients, orange for asymptomatics, yellow for infecteds, green for those who are susceptible to infection, and grey for those with an immunity to the disease. Additionally, it is important to be aware of how the six control parameters work on a fundamental level. When PPE is switched on, a percentage of the population, dictated by the compliance value, will wear face coverings and other protective equipment – this cuts their infection chance down by 80%. This change of infection chance is achieved by taking a random value from 1 to 100 for each turtle; if it is lower than the compliance value, that specific turtle complies, and wears a face covering in this case. Lockdown causes the given percentage of compliant turtles to confine themselves within their houses until the infection rate drops below the given threshold, at which point they can resume their usual activities; it is a similar case for shielding, but only influences those who are identified as "at risk", in the case of the model these are individuals over the age of 74. Self-isolation functions in a similar manner too; if a turtle has been asked to self-isolate, either due to being sick, being tested positive, or being identified as a close contact of another positive case, then they will isolate themselves for a set period to wait off the effects of the virus, provided they are compliant.

Testing and tracing have perhaps the largest impact on the system; when this is switched on, it remains on for the rest of the simulation, and largely follows the real-world test and trace system; individuals can be tested for coronavirus, and the higher the "test-coverage" value, the more likely the system is to pick up a positive case. These positive cases are then analysed to find their most recent contacts (included as a set of neighbours which is re-evaluated for everyone on each day), and these contacts are then asked to self-isolate. Again the "trace-contacts-reached" value controls how successful this getting in touch is, and the compliance controls how many are likely to follow this advice. This altogether creates a vast network of communication between different individuals and goes a long way towards stemming the spread of the virus in a particular section of the modelled area before it can spread to others.

Based on the age of the individual turtle, there is some standard deviation on the values selected in the model for factors such as death chance, recovery time, and so on. These deviations are again drawn from Scottish NHS data and further develop the accuracy of the model in terms of realistic simulation of individual characteristics and behaviours. As stated earlier, it is impossible to create a perfect model, but it is possible to come reasonably close with just a small handful of variables introduced, as with this model. The model is then matched against real-world data for Scotland’s coronavirus infection trends so that the evolutionary algorithm can assess the most cost-effective methods of combating the spread in each area; this is covered in the next section.

It is important to consider why NetLogo, or any modelling tool, is useful for this project. When we consider just the I/O values of the 6 parameter switches, we have an extremely small search space of only 64 possible combinations (2^6), which is easy for a human to search through to find the “best” solution in conjunction with the cost function used in the genetic algorithm. However, once we consider the slider values for thresholds (and especially if we also include those for compliance), then this search space increases massively to well over 100^6 for the sliders alone, giving us over 1,000,000,000,000 possible combinations. This is of course impossible to search through in any reasonable time, let alone by a human, so this is why we need to make use of the evolutionary algorithm to speed things up; the NetLogo model runs headlessly and allows for quick changes to variable values before running again, straight from the program running the algorithm, so there is no need for human intervention to set up parameters for each run, and to have to keep an eye on operations at all times, for example. This greatly increases the workflow of the project when compared to standard human workflow.

3.1.3 Evaluation

The first, and most critical stage of evaluation of the model, was to perform data matching against appropriate real-world data. The Scottish Government’s coronavirus data for the first 30 days of tracking were used against the averaged results of many simulations of the model, with all procedures turned off; the 30-day averages of the number of infections were normalised along with the initial 30-day data for the observed set and plotted against one another, using the equation $i = \frac{i - minValue}{maxValue - minValue}$, where i is the current infection count for a specific day. Normalisation is important as it allows us to keep the two sets of values in line with one another, as a proportion of their highest possible values. This means that if we have a situation with double the usual population, the average infection counts will still be represented in the same way as if we used a smaller one, as a proportion from 0 to 1. The lowest infection value in the 30-day set will produce a normalised value of 0, with the highest yielding a value of 1. The Euclidean distance between the two data sets was found using the equation $d(p, q) = \sqrt{(p - q)^2}$, where p and q are the sums of the 30 values for each data set. By doing this, we can evaluate whether the model follows the same general behaviour as the real-world data, even if it may come up with figures that are many times lower than reality. After finding the Euclidean distance between both sets of data, and plotting the normalised values, the following graph is produced:

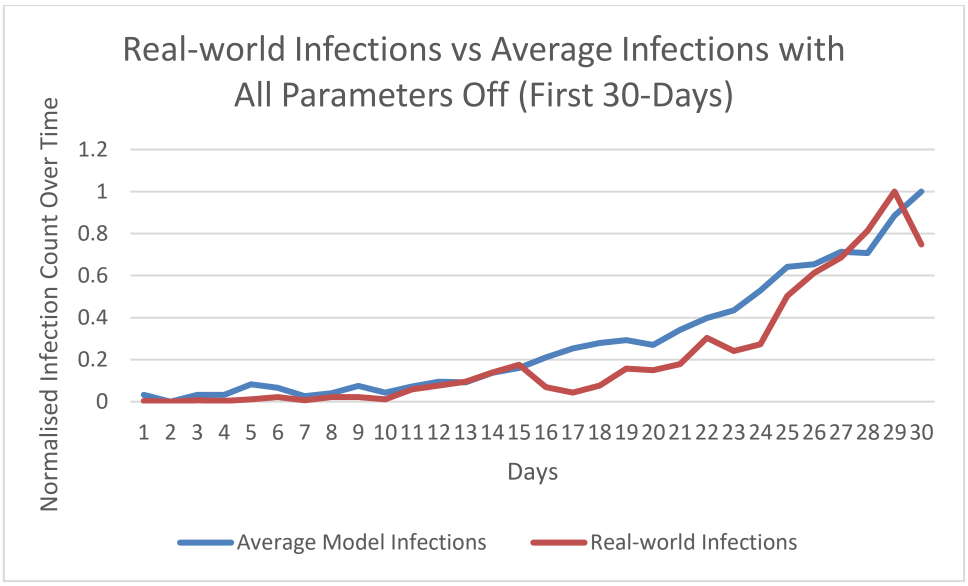


Figure 10. Graph of real-world infection data vs average simulation infection data over the first 30 days with no control parameters enabled, normalised to a range of 0-1.

Here we can see that the two trend lines are highly similar in terms of their values; both infection numbers begin to escalate at around day 10 and reaching a peak at the end of the graph – day 29 and day 30 in the case of the real-world data and simulation data, respectively. The Euclidean distance of this dataset was 0.632976, a low value, given that the worst we could indicating that the model does for the most part accurately simulate the uncontrolled spread of coronavirus.

Another similar experiment was performed, this time of the first 70 days with all control parameters turned on; here we would expect two more disjointed lines, as in the real-world, all procedures were not introduced roughly simultaneously, but more staggered out over a longer period(Shielding and self-isolation began on the 16th of March, social distancing, and face coverings two weeks later, but Test and Protect did not launch until mid-September). It is important to consider that testing became widely available in mid-February in Scotland, so we would expect to see a large spike in infections at around 1 month into the data. Comparing the first 70 days of real-world data with those of multiple simulations, we get this graph, as above:

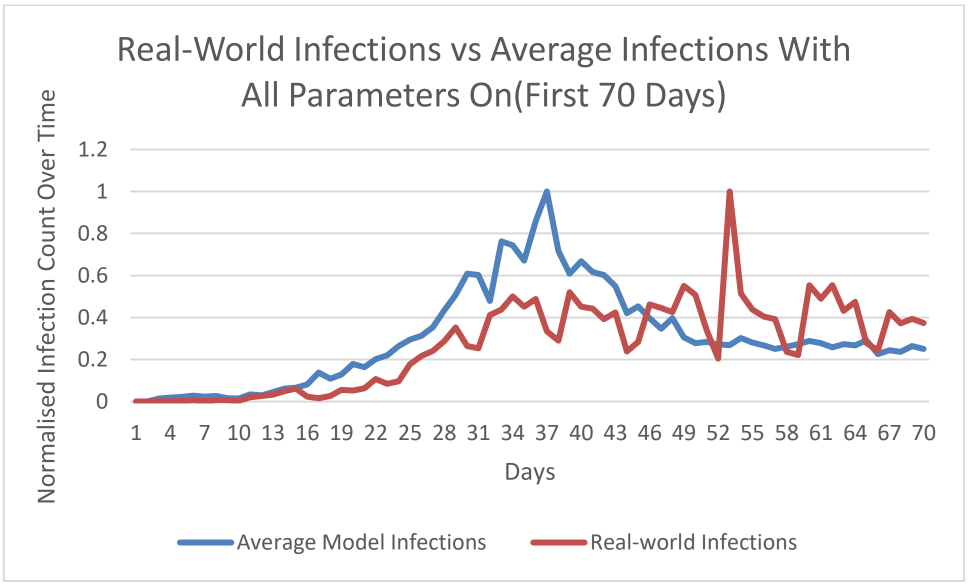


Figure 11. Graph of real-world infection data vs average simulation infection data over the first 30 days with all control parameters enabled, normalised to a range of 0-1.

We can see that the lines follow a similar pattern to one another up until around day 46, at which point we should expect to see the model trend line continue along the same axis rather than decrease, indicating that there is a deviation in the way the model handles the spread when compared to the real-world data. In reality, reported daily figures always show a significant drop on a Monday, followed by a spike on Tuesday as the weekend's figures are caught up on. Therefore, we can see the sharp peaks and troughs with the real-world infection trend line, and not with the simulation, as this is not implemented into the model. If we take this into account and average out the peaks and troughs, we expect a line fairly like that seen with the simulation, albeit slightly higher due to a higher rate of infection. We can therefore conclude that this experiment with all parameters on is also accurate at simulating the spread. Here the Euclidean distance is 1.660696, which is much higher than the previous experiment, however considering the issue with Monday data reporting, this is roughly the range of result we would expect, however, we must take into account that the simplicity of the model in comparison to the real-world situation could be having an impact on the results, particularly due to the unpredictability of real individuals, means that the results for the model tend to follow a fluctuating sine wave type pattern, rather than a more unpredictable one. With a greater level of detail and stochasticity we could potentially see results closer to those observed values, however. Hence, we can conclude that the model accurately simulates the behaviour and spread of coronavirus throughout a population, even when considering various control procedures.

Aside from the data matching, there are other aspects to evaluate regarding the completeness and accuracy of the model. As it is impossible to model every possible real-world scenario and function, it is therefore impossible to create a 100% accurate model. There was a plan to allow multiple individuals to live within one home, and therefore there would be a greatly increased risk of disease spread within a household compared to other places, but due to both time constraints and the fact that the simulation runs in day-long steps, it was not possible to model the individuals returning home at the end of the day after work, for example.

Also of consideration was the implementation of a vaccine system for the model, whereby after a set amount of time, a vaccine for the disease would be created, which could be administered to the population to provide long-term (many years) or short-term (one year) immunity. This would have added another layer to the model by having seven control parameters and could result in some interesting findings with the genetic algorithm; however, given the amount of time it takes to research, create, and test a vaccine, this would not be a suitable parameter to include in the model, particularly as a standard run occurs over only 365 days, and places the focus more on short-term disease control solutions. For this reason, we could argue that after 365 days, the vaccine would be ready, and therefore the infection rate would drop dramatically, so perhaps the vaccination program could be implemented in a future experiment if this was repeated and more time was available – the genetic algorithm may discover that it is more economically viable to keep infections higher for the sake of only having to invest heavily in vaccines and nothing else, for example.

Overall, however, I feel that this model does a very accurate job of simulating the spread and control of a disease like coronavirus, particularly in terms of incorporating the random element that can be seen with real-world disease behaviour, by means of the different age and sex groups, as well as the varying standard deviation on the chance of infection and death from the disease. With more time the model could become more and more complex to further increase its accuracy to real-world scenarios, however for the purposes of this project, the current model is a complete and accurate solution which works well with the Java program and the genetic algorithm in helping to find a cost-effective solution.

3.2 Implementation of Evolutionary Algorithm

3.2.1 Plan

This is the most time-consuming section of the project, largely due to inexperience in this field, but this provides a lot of scope for development of new skills and testing of ideas. Evolutionary algorithms are a must for this project as it is not possible to manually iterate through every possible combination of values; were we just focusing on the I/O values of the six parameters this would not be a problem, since there would only be 2^6 or 64 combinations, but since we are also considering the slider values for the threshold at which we introduce each procedure, there are effectively infinite combinations that we can have (well over 1,000,000,000,000). Hence, we need to make use of an algorithm in conjunction with an effective cost function to narrow down and find a set of possible solutions.

The evolutionary algorithm that will be used is a subset of the field, called a genetic algorithm. This closely imitates the behaviour of real-world evolution; each generation, we have a set of possible solutions (populations), which each have a unique setup for the parameter values. In the first generation this is largely random, however, after each population in the generation has been tested, and its score calculated, the lowest-scoring ones are selected to move onto the next generation – the exact number selected depends on the tournament value. The higher the tournament value, the more solutions from the population are chosen to "breed" to produce new solutions, with characteristics of the two parents; this is how we get the "evolution" aspect of the algorithm and continually work towards a "good" solution. These solutions are carried over into the next generation, which is populated with the same number of populations as before, but this time they are based on the parent solutions. Some degree of mutation is introduced within these solutions, which simulates real-world evolutionary mutation, and helps to prevent the algorithm from becoming stuck in a trough which it cannot get out of, even if there may be a more effective solution elsewhere. The lower the tournament size, the less chance we have of entering a hill-climbing situation, which does not accurately simulate evolutionary behaviour.

Regarding the cost function (or score) used with the algorithm, it needed to factor in both the cost of implementing the control parameters, as well as the inherent cost of allowing someone to become infected. For each population of a generation, we run it five times to obtain an average value for infections, deaths, total populations, and so on for each day. For each day, there is an associated cost of i + m, where i is the infection cost for the day and m is the cost of the procedures currently being used. This will give us an array of daily costs, which we can sum together to get the final cost. Because we want to keep the two costs in proportion to one another, to prevent one from becoming more heavily weighted (valuable) than the other, they will both be normalised relative to their worst-case scenarios; that being if everyone got infected, and if all procedures were on, respectively. This should give us a final cost of between 0 and 2, if we divide the total cost by the number of days we simulated for at the end. The exact implementation of the cost function will be expanded on in the next section since there is liable to be many iterations of the cost function over time while the most appropriate and accurate one is found.

Once the cost function and evolutionary algorithm are set up and refined to a good standard, the final stage is to run the algorithm multiple times to allow it to come up with several optimised solutions which we can compare in the final analysis.

Regarding the output of the solutions:

- Each run will produce a tab-separated text file with the values of the genes (both procedure I/O statuses and slider values), along with the total infection count, cost, and number of deaths.
- The cost will be split into 3 sections; the infection cost i, the procedure cost m, and finally the total overall cost after all calculations.
- All this data together should provide all the necessary information to draw conclusions on which procedure setup is best.

There is no need to include the total number of immune patients, or asymptomatics, as while it could be useful in some ways, it is not necessary for the purposes of the project in drawing conclusions on the most effective method for keeping both infections and costs to a minimum; deaths were included as this is factored in to the overall cost function, and we may find that some setups are willing to let more people die to save money for example.

3.2.2 Implementation

The basis for the evolutionary algorithm came from a practical on evolutionary algorithms and hill-climbing behaviour in a previous module. This was built upon and adapted to suit the needs of the project by making use of a one-dimensional chromosome with 12 entries, one for each aspect of the disease model that we want the evolutionary algorithm to be able to experiment with. Initially, only 6 were used, and these represented the I/O status of the switches for the 6 control parameters. By switching these on and off with based on the population's chromosomes, the evolutionary algorithm can get a (very limited) range of solutions which can be compared. At this stage we of course find that the evolutionary algorithm has no real way of comparing the solutions, so we must introduce the cost function.

The cost function went through a lengthy process of development which was to be expected; it is extremely difficult to develop an accurate one without an iterative process of trial-and-error. The first step was to consider what elements to incorporate into the cost function; the decision was made to split it into two parts; the infection cost, i, and the procedure cost, m. Initially we used the equation i-m, under the belief that the genetic algorithm would attempt to find the best combination of the two which would cancel one another out – if i ended up being 500, with an m of 1000, for example, the final cost would be -500, so the algorithm would try to increase some procedure thresholds to bring it back closer to 0. This did not work as expected, and simply resulted in the algorithm turning everything to its most expensive setting, as a very low negative number is of course much "better" than a value close to 0. To this end, we took the absolute value instead, where if we got a negative number, the value would be flipped back to a positive number. This showed some improvement, however we still found that the algorithm would favour one part of the equation over the other; preferring to keep the cost of procedures low at the expense of a high number of infections. This led to deciding to add the two components together, rather than subtract them; this meant they were in tandem with one another, and as such changes to either i or m's cost had a direct impact on the overall cost, and the worthiness of the solution. At this point we began to see more expected behaviour from the evolutionary algorithm; the main issue now was settling on a reasonable equation to use.

The final equation settled on for the cost associated with a specific day of the simulation can be seen below, and through a series of short tests of the results obtained, was found to be the most suitable with respect to the parameters we want to experiment on. To keep the two costs in line with one another, we calculate a maximum, worst-case-scenario cost for both i and m, and the actual cost is then represented as a ratio of the max. This helps to normalise the two values, and in cases where the procedure cost might hugely outweigh the infection cost, they are still converted to similar values, preventing the genetic algorithm from ignoring one in favour of the other.

$$eqnI = \big((averageDailyInfections[d] * averageDailyWage) + (averageDailyDeaths[d] * deathCost)\big)$$

$$maxI = \big((averageDailyPopulations[d] * averageDailyWage) + (averageDailyDeaths[d] * deathCost)\big)$$

$$normI = \frac{eqnI}{maxI}$$

$$\begin{aligned} eqnM = (((&isPPEOn * (1 - (\frac{ppeThreshold}{100})) * ((\frac{ppeCompliance}{100}) * averageDailyPopulations[d]) * ppeCostPerDay) + \\ &(isLockdownOn * (1 - (\frac{lockdownThreshold}{100})) * ((\frac{lockdownCompliance}{100}) * averageDailyPopulations[d]) * lockdownCostPerDay) + \\ &(isSDOn * (1 - (\frac{sdThreshold}{100})) * ((\frac{sdCompliance}{100}) * averageDailyPopulations[d]) * sdCostPerDay) + \\ &(isIsolationOn * (1 - (\frac{isolationThreshold}{100})) * ((\frac{isolationCompliance}{100}) * averageDailyPopulations[d]) * isolationCostPerDay) + \\ &(isShieldingOn * (1 - (\frac{shieldingThreshold}{100})) * ((\frac{shieldingCompliance}{100}) * averageDailyPopulations[d]) * shieldingCostPerDay) + \\ &(isTTOn * testAndTraceCost * (1 - (\frac{ttThreshold}{100})))))\end{aligned}$$

$$\begin{aligned} maxM = (((&1 * (1 - (\frac{0}{100})) * ((\frac{ppeCompliance}{100}) * averageDailyPopulations[d]) * ppeCostPerDay) + \\ &(1 * (1 - (\frac{0}{100})) * ((\frac{lockdownCompliance}{100}) * averageDailyPopulations[d]) * lockdownCostPerDay) + \\ &(1 * (1 - (\frac{0}{100})) * ((\frac{sdCompliance}{100}) * averageDailyPopulations[d]) * sdCostPerDay) + \\ &(1 * (1 - (\frac{0}{100})) * ((\frac{isolationCompliance}{100}) * averageDailyPopulations[d]) * isolationCostPerDay) + \\ &(1 * (1 - (\frac{0}{100})) * ((\frac{shieldingCompliance}{100}) * averageDailyPopulations[d]) * shieldingCostPerDay) + \\ &(1 * testAndTraceCost * (1 - (\frac{0}{100})))))\end{aligned}$$

$$normM = \frac{eqnM}{maxM}$$
$$dailyCost[d] = Math.abs((infectionWeighting * normI) + (methodWeighting * normM))$$

Where ‘d’ is the current day we are looking at from the results; typically, we run for 365 days so we will end up with an array of 365 daily costs, calculated with respect to the average daily infections, population count, and so on, on that day. Also note that **Math.abs** returns the absolute value of the equation, so we can avoid negative values, as mentioned above.

Once we have our array of daily costs, we can get the final cost:

$$totalCost = \frac{(float)DoubleStream.of(dailyCost).sum(\ \ )}{noDays}$$

Where the numerator is the sum of all the daily costs, cast to a float to prevent any rounding errors.

Now that a cost function had been settled on, some testing was needed to assess whether it was fit for purpose. Sample runs were performed of the algorithm with a generation size of 50, and the results assessed by eye to evaluate if the cost function were willing to experiment correctly and in the way we would expect, rather than becoming stuck on one solution. For the shorter chromosome with only the switches affected by the genetic algorithm, it was of course able to find a solution it was satisfied with fairly quickly, but once the chromosome was expanded to include the threshold slider values, we could see a definite improvement in the behaviour of the algorithm; with it willing to make changes that would initially be seen as suboptimal but would turn out to be more effective in the long run, thanks to the increased dimensionality of the solution space and the changes afforded by the mutations. At this point, we could observe the type of behaviour we would expect from a genetic algorithm, so we could conclude that the cost function works effectively and correctly in conjunction with it.

One other important stage was to identify which parameter setup should be used for the simulations, regarding the population size and the number of generations per simulation. This needed to be assessed with respect to the remaining time; as at this point there were only a few weeks left to finish the project, so there needed to be a trade-off between the accuracy and the number of results obtained. To briefly highlight the process used:

- Several runs were performed with a population of 100, 200, 400, and 800 in the disease model, and their best total cost and parameter setup saved and analysed.
- The best balance between execution time and accuracy of results was found at a population of 400 – increasing the NetLogo model’s population even slightly results in a huge increase in computational overhead regarding the tracking of the individual’s current state, position, and all their relationships with neighbours.
- Similarly, runs were performed with varying genetic algorithm populations, from 5 to 40, doubling each time.
- The results obtained here (using a 400 population in the disease model) indicated that the best trade-off between accuracy and timeframe could be found between 30 and 40.

The full table of results can be seen below:

Model Population	GA Population	Time	Total Cost of Best Solution	Total Infections of Best Solution
400	5	392.452637	0.9504386	3153.49927
100	30	636.997314	0.10747383	44.20002
400	10	776.315369	0.8017283	5589.80225
200	30	1206.08044	0.06407711	166.69987
400	20	1768.72119	0.52982193	3162.39966
400	30	2385.31738	0.6876609	4439.99902
400	40	3346.31641	0.3285798	1606.7002
800	30	5117.99023	0.28581247	4408.79883

Based on the above results, I decided to use a population of 30 in the interests of saving time and collating more results, and to use a generation size of 20, given that in testing, we noted that the most optimal solution for a run is typically identified at around 15 generations, so it did not seem worthwhile to go for say, 40 generations, when there may only be a slight improvement over solutions found at 20 generations. It was important to get the values of the standard parameters just right to strike a balance between exploration of possible solutions, and exploitation of the most ideal ones; for this reason, we use a tournament size of 8 and mutation rate of 0.05 as this keeps a comfortable selection of solutions to breed the next population, while introducing a slight mutation on the child chromosomes/solutions, not too much for them to become vastly different and possibly worthless, however.

Finally, the cost function and genetic algorithm were ready for final solution gathering. Given additional time, it would be ideal to perform as many runs as possible, however this was unfortunately not possible. Given the average execution time of a full run was around 40 minutes, I opted to perform 30 runs of two variations of the algorithm; one set with only threshold slider changes allowed, and one with both threshold and I/O values changeable. I performed 20 runs for the variation where only switches could be changed, as due to the extremely small search space, there is little benefit to doing an extra 10 runs, when 20 gives a general indication of the poor performance of this style of searching. 30 runs should be enough for the other two, to give us a general sense of the type of solutions obtained, even if it is nowhere near the size we would ideally like. The results were stored as tab-separated text files consisting of the following:

- The solution setup (i.e., the values of the parameters)
- The average daily infections
- The total number of infections
- The average daily deaths
- The total number of deaths
- The total cost of the solution
- The parts of the total cost, i and m

With all these results collected, they could then be imported into Excel for the creation of graphs and charts for use in the final analysis.

3.2.3 Evaluation

As this was the most complex part of the project, there are a lot of areas to investigate for evaluation.

Firstly, we can look at how the cost function itself was tackled. The issue with the cost function primarily lay in finding one which was equally weighted in terms of both the infection and control method costs (to prevent one part of the cost from being outweighed by the other and completely ignored by virtue of being too high). The tendency initially was for the algorithm to settle on not turning on any control procedures, as the cost of infections was so dramatically low that there was no reason to do so. For this reason, as seen in the implementation, weightings were used to balance i and m out. Given the fact that the average cost of m in most scenarios was around 12 times higher than that for I (as the economic impact of the procedures themselves far outweighs the "cost" of allowing someone to become infected or off from work for a time), the value of i was multiplied by 12 at the end of each daily cost calculation. In combination with the normalisation, this helped to keep the two costs in line with one another and to avoid the issue of the genetic algorithm favouring one aspect of the cost over the other. Despite this issue and considering the complexity of the world within the model, I feel that the process of refining a simple measure from it went well. It was a lengthy process to come up with an effective cost function, given all the ways in which the model could potentially stray away from the type of results we would expect, however they were dealt with well. One major example is that during the initial setups of the cost function, the algorithm was willing to turn the threshold to its lowest possible value, since the cost was setup to increase as the slider percentage increased. Upon changing this to correctly **decrease** as the threshold of introduction increased, we find that the cost function works much more effectively; the algorithm must find a trade-off between how early to introduce the various procedures, and the cost associated with introducing them too soon.

In addition, some aspects of the cost function were hard to judge, as they have no quantifiable "real-world" equivalent. For example, there is no inherent "cost" associated with allowing an individual to become infected or to die, so the most appropriate metric I could come up with involved factoring in the average daily wage for a Scottish employee, and the average cost of a funeral, respectively. Since we cannot possibly account for the economic impact of each individual's absence from work due to the vast array of jobs, this fit the purpose well enough, with the cost function seeking to avoid deaths, and therefore infections, as funerals had an extremely high cost of £4000 per person, thereby keeping the algorithm in check.

There were other ways in which the cost function could be refined; one such method would have been to improve the way the cost of an infection is thought about – by factoring a much wider range of economic impacts from having an infected individual be off from work, or occupying a hospital bed, the infection cost would likely be much higher, and fall more in line with the cost of the procedures. However, due to time constraints, it was not realistic to account for all these issues, so the cost function is perhaps more generic than what could have been achieved, only having a flat cost associated with an infection, death, or out of office employee, rather than varying based on age, gender, or location, for example.

Finally, we can briefly think about how to further improve the accuracy of the model and algorithm by increasing the population used:

- The runs for this project used a very small population of 400, largely to prevent the computational overhead from becoming too large, given the sheer size of data that needs to be tracked about everyone regarding their age, gender, status, and neighbours.

- In reality, Scotland's population is around 13,635 times higher (5.454 million people), and as such, the sparse nature of our population means there are far fewer opportunities for disease spread.
- Even with the normalised proportions being used, we still cannot account for the increased stochasticity that comes with a larger population, even when not including the other unmodelled aspects such as families living together, transmission by wind direction and speed, enclosed spaces, and so on.
- For this reason, we can consider that in a future experiment, we might wish to optimise the neighbour tracking system, or omit it altogether, in favour of adding many other real-world aspects to the model and improving the accuracy of the genetic algorithm relative to the observed data.

Much consideration could be given to other methods of tackling the genetic algorithm and cost function. Due to time and resource constraints, the solution used in the project did not involve the changing of the compliance sliders. Adding these to the genetic algorithm's search space would have drastically increased the dimensionality and time taken to find effective solutions, which could not be afforded. (If looking at all 14 sliders, this would mean a search space of over 100^{14}. In the sense of presenting this project's findings to a research team or government official, they are more in control of when and in what way they begin to implement procedures to mitigate disease spread, more than they can control public compliance with said procedures, so the focus was placed solely on the I/O values and threshold sliders. Given more time, it would result in a much more complete (and accurate) solution to include as much dimensionality as possible for the algorithm to work with; the same goes for the overall flexibility and complexity of the NetLogo model itself, as mentioned in the earlier section.

One other way to improve the genetic algorithm would be to keep a store of some scenarios and their results within the program itself, such as when every slider is turned to its max value, everything is off, and so on. The final cost function made use of the current daily cost as a proportion of the worst possible cost for that day to create more a more accurate sense of the impact of certain procedures, but perhaps having more of these saved within the program, such as some prior poor solutions found in other simulations, could allow them to be adapted into the genetic algorithm itself to speed up and refine the solution-finding process.

Overall, I feel that the implementation of the genetic algorithm meets the goals of the project well, and still resulted in some accurate and useful results being generated, however there was obviously a lot of room for improvement. It is worth keeping in mind that as a relatively complex and broad project, there will always be more features to add and refinements to make, particularly in the complexity of the disease model – that could be developed for a very long time and still not be perfect; therefore, I think the algorithm implementation was highly successful. Repeating this experiment again, a larger chunk of time would need to be dedicated to this section of the project, and with familiarity with all the tools used, I feel this would be achievable, by finishing up the NetLogo model far earlier than before and leaving a larger window to work with for polishing the cost function, speeding up execution, and increasing the dimensionality of the search. The values used for the number of generations and the size of their populations, as well as the small but meaningful mutation rate, mean that the results were sensible and relevant, and the best that could be gleaned from the (relatively) small search space, while managing to avoid pitfalls such as becoming stuck in troughs or a hill-climbing situation.

3.3 Professional Computing Approach

To keep track of progress and remain on schedule for this project, particularly due to the difficulties in working from home, extensive use was made of facilities such as Microsoft Teams to keep a detailed diary of progress, including what was done each week and what was due over the following week, as well as for keeping in touch with project supervisors and ensuring the project remained on task and accurate. Microsoft Outlook was also used to keep track of relevant dates as well as ensuring regular university work was completed to the appropriate deadlines.

It is also worth considering how this project maintains a commitment to the *BCS Code of Conduct*.[15] The 4 main principles, and how they were adhered to, are as follows, and informed from the BCS website's PDF:

1. **Public Interest**
    a. This project has due regard for public privacy and safety by not incorporating any personal data in the simulation process; all information taken from the Tableau Public dashboard is fully anonymous and there is no risk of data breaches.
    d. The model should ideally be accessible and useable by all sectors and industries without requiring prior knowledge of computing concepts.
2. **Professional Competence and Integrity**
    a. By incorporating prior knowledge from coursework in combination with learning new skills this is within my professional competence.
    b. Also ties in with point c); this project is an opportunity to learn and develop new skills, not just in computing but in presentation and confidence skills also.
    c. Constant reviewing of government data ensures the model is as accurate as possible at all times.
    e. Frequent meetings with supervisors ensure that there is always an opportunity for criticism and therefore further improvement of the work.
3. **Duty to Relevant Authority**
    a. The greatest care is taken at every stage to ensure that the work produced is as professional as possible and within the interests of the University.
    b. As this work is partly owned by the University, I accept personal responsibility and a commitment to ensure work is completed to the highest possible standard.
4. **Duty to the Profession**
    a. This project does not involve any actions or data that could bring the reputation of the University into disrepute.
    b. Constant development of skills took place throughout this project.
    c. The BCS Code of Conduct, along with the reputation of the BCS, was always upheld throughout the project.
    e. Engagement with others on the course helped me to encourage my co-workers to produce the highest quality of work that they could.

4 Analysis and Conclusion

4.1 Analysis of Data

After completing the runs of the genetic algorithm, we have 3 primary sets of data to work with, one where the GA could only change the six switch values, one where only the threshold sliders could be changed, and finally, one where both switch and threshold values could be changed. This should give us a good spread of information to work with and form a basis for which setup can be considered most accurate or useful. The best of these 3 will be analysed further to assess some of the "best" solutions it offered, as well as if it is fit for purpose, and does its job to a good standard. We can also analyse a parameter sweep of the type of results we could expect with each parameter setup for the switches to see if the solution suggested by the "Switch only" dataset was what we would expect, allowing us to further see if the program is working correctly. In addition, there is also a short data set tracking the average run time for an experiment under varying population and generation value conditions – this should allow us to assess the type of overhead we can expect were we to repeat the experiment with more time, or a more powerful machine.

4.1.1 Switches Only

Firstly, let us look at the "Switches Only" dataset. Plotting the total cost against the total infections from across the 20 runs of the genetic algorithm, we obtain the following scatter plot:

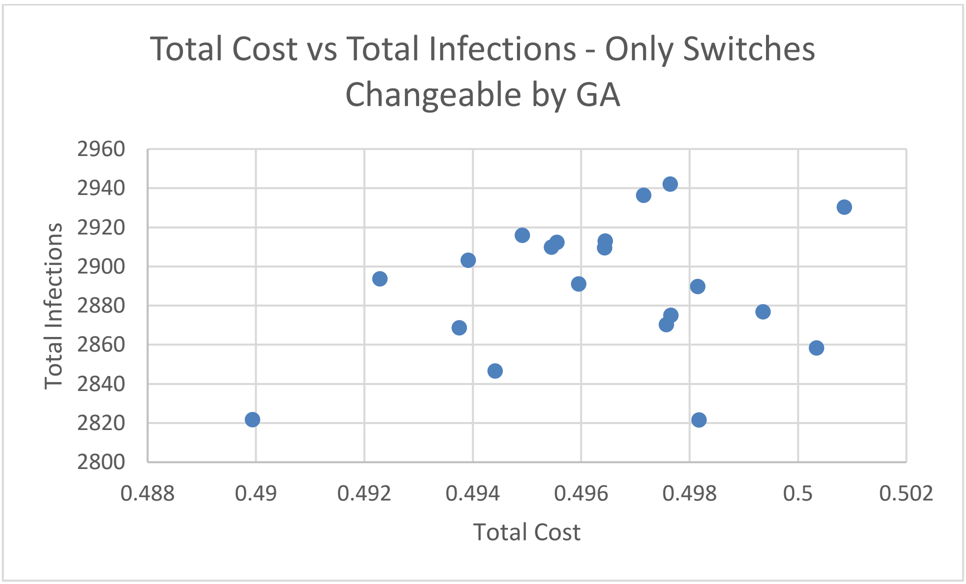


Figure 12. Scatter plot of cost against infections for the various solutions obtained by the genetic algorithm while only changing switch I/O values.

We can see clearly from this plot that there is little correlation between the total cost and the infection count when only toggling the on/off status of the switches. This indicates to us that most of the search space, and therefore improvements to be made to solutions, lie with the slider values. All 20 plotted solutions use the same combination of I/O values; 100011. In addition, the cost is kept tightly confined, with a maximum value of 0.50085 and a minimum of 0.48993, a difference of just 0.01092, again indicating that much of the improvements that could come from experimenting with the model would be seen with the slider values, rather than the on/off status of the switches. This is of course mainly due to the very small search space when only changing switch status; there are only 64 possible solutions, and the sliders have much more room to experiment with, with over 1,000,000,000,000 possible combinations. The minimum value we can achieve comes a cost of 0.49817, at an infection count of 2821.60, which is extremely high given that the population of the model is only 400 by default. Hence, we can conclude that simply changing the switches is not enough to come up with accurate or useful solutions.

In addition, if we look at the parameter sweep of all the 64 possible combinations, using 5 loops of the disease model simulation to get a more accurate average cost, we get the following top 10 results, sorted by total cost:

Solution	Total Cost
100011	0.51921016
100010	0.569595
100001	0.5858147
101011	0.6575537
101010	0.69699144
100111	0.6985221
101001	0.7215911
100101	0.73398805
100000	0.7378835
100110	0.7566692

Figure 13. Table of the top 10 switch-only solutions, sorted by total cost, taken from the 64-solution parameter sweep.

As we can see, the lowest cost solution from the set of 64 switches on PPE, Test & Trace, and social distancing, and is the same "best" solution identified in each entry of the scatter plot, despite there only being 20 plotted solutions. Given that the algorithm settled on the same switch combination in each of the 20 runs, we can say with good confidence that the genetic algorithm is working as intended, and able to find the most optimal solution consistently. Given the huge size of the search space when dealing with switches and sliders together, we cannot say with confidence that it is as effective with that dataset, but this provides a good metric for confidence in the model's ability to find good solutions, even as the search space increases.

4.1.2 Sliders Only

Next, we can analyse the “Sliders Only” dataset; again, plotting total cost against the total infections for the 30 runs, we obtain this scatter plot:

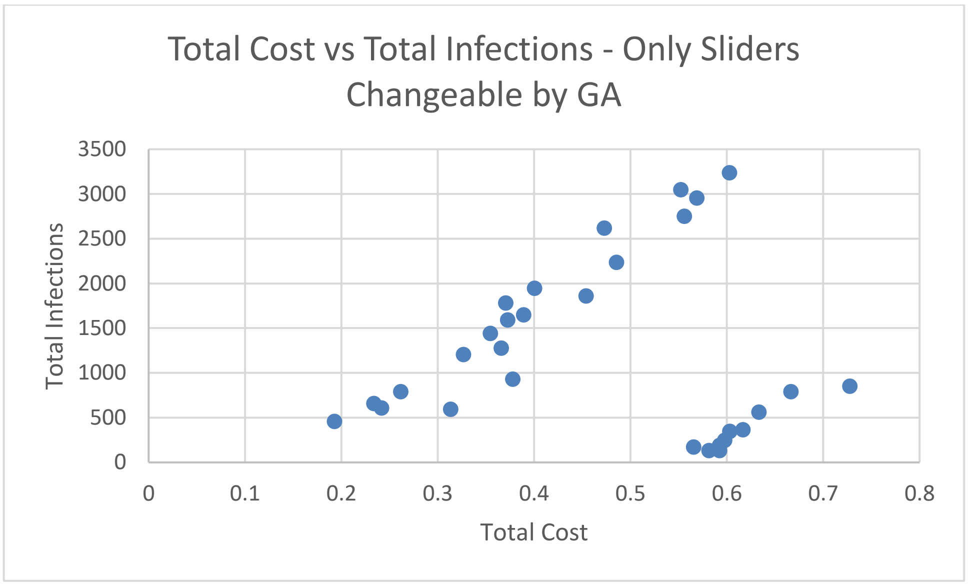


Figure 14. Scatter plot of cost against infections for the various solutions obtained by the genetic algorithm while only changing threshold slider values.

In this plot, we see much more viable solutions; there are two distinct groups here; one set indicates that as cost increases, infections also increase, suggesting that not all expensive solutions are effective, if the correct procedures are not configured well. It is however important to consider that since infections make up a part of the cost function, we would expect infections to increase with cost to some degree. The second group follows a similar trend, but at a much lower infection count – there is a sizeable cluster of solutions at around the 0.58 to 0.60 cost area. These also happen to be the best solutions in the set, which is more in line with my original expectation that an increased cost would lead to fewer infections. However, despite the best solution being found at a cost of 0.58135 and infection count of 133.20, an almost as effective solution can be found at the beginning of the first group mentioned earlier, with a cost of 0.19254 and infection count of 456.3. There is a clear point at which two solutions become equally viable; in the former, we invest a large sum of money for a very small infection count, whereas the latter uses approximately 1/3 of the cost, but results in (very roughly) 3 times more infections. We cannot reasonably conclude that one is “better” than the other in this case, but this is what we want out of this experiment, as there is no definitive “best” solution, it is mainly subjective.

4.1.3 Both Switches and Sliders

Finally, the "Switches and Sliders" dataset is the most complex in terms of the facilities offered to the genetic algorithm – plotting total cost against total infections for the 30 runs, we obtain the following scatter plot:

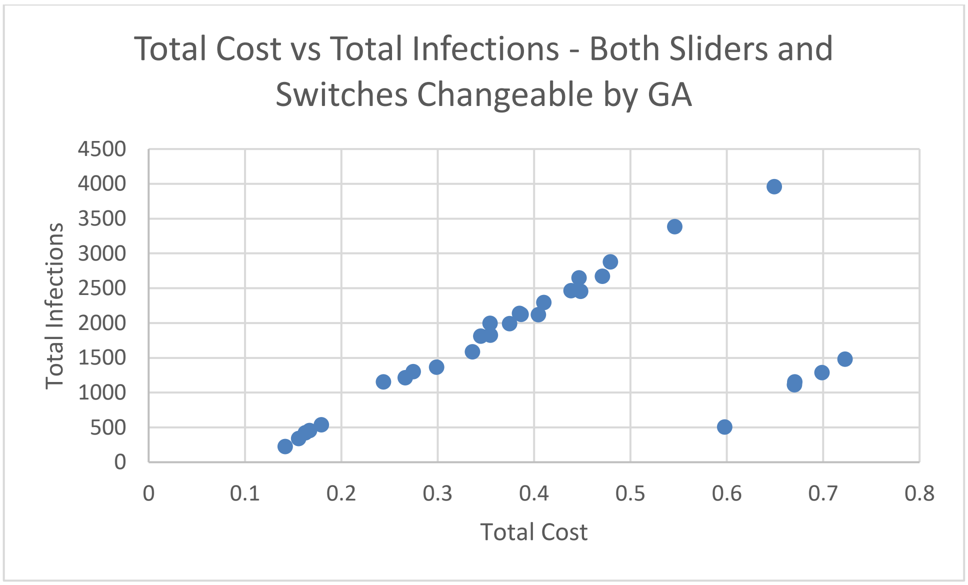


Figure 15. Scatter plot of cost against infections for the various solutions obtained by the genetic algorithm while changing both switch I/O and threshold slider values.

This is similar in nature to the plot seen with only sliders but is much more tightly spread. We still have the two distinct groups of results, but this time it is clear there is only one cluster of good results, those seen in the bottom-left of the graph, between a cost of 0.1 and 0.2. There is a good result infection count wise at a cost of 0.6 but given the wide range of solutions seen between 0.1 and 0.2, we can extrapolate this and predict that most good results will be centred around this point. The objective best solution here, with both lowest cost and infection count, totals 0.14159 and 226.10, respectively. Again, we see the trend that a higher cost does not necessarily guarantee good solutions, showing that it is more a case of how the cost is allocated than how much. This also poses a good case for the genetic algorithm and cost function, as it can home in on these low-cost, low infection scenarios often, rather than trying to only keep infections or cost down.

4.1.4 Overall Analysis

If we combine the three scatter plots together, we can further analyse the accuracy achieved with each setup:

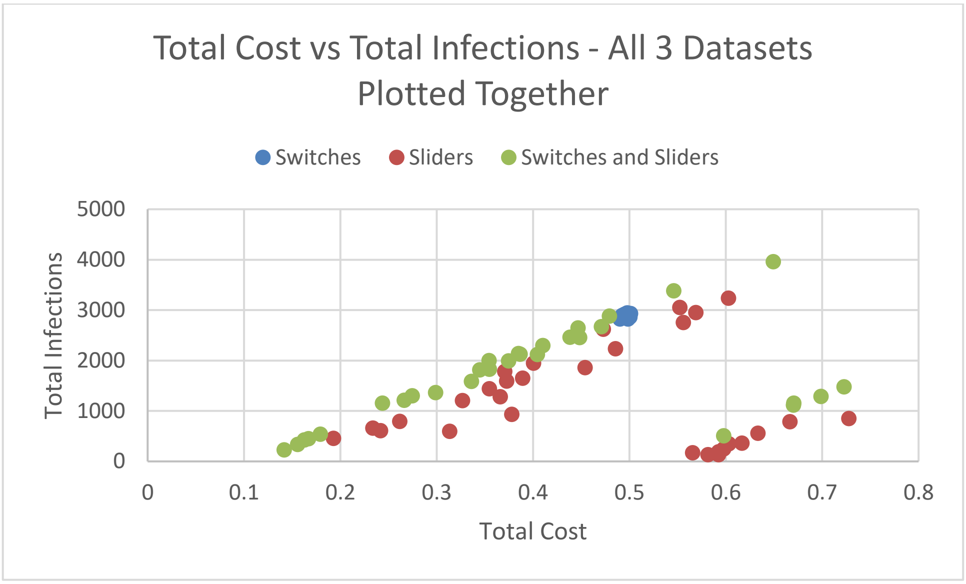


Figure 16. The three prior scatter plots plotted together for ease of analysis.

Now that we can see the three data sets alongside one another, we can see much more easily just how tightly grouped the results are for the "Switches only" data set, due to the very small search space the GA had to work with. The other two sets follow roughly identical patterns, with two distinct groups as seen previously. Generally, we find that the solutions offered by the slider-only set generally result in a lower infection count, but with a more varied spread when compared to the switch and slider group, which is more tightly spread, and has several good solutions which keep the infection count and cost to a minimum. I believe the reason we see such a varied spread in this case is due to both the small number of GA runs performed, and the more limited number of parameters and dimensionality it could work with; this likely made it more difficult for the algorithm to home in on good solutions as consistently, so consequently there is a little bit more 'randomness' in the distribution of results. The lowest overall infection count can still be seen with the slider-only set, at a cost of 0.58135 and infection count of 133.20. On the other hand, the lowest cost can now be found from the switch and slider set at a cost of 0.14159, and infection count of 226.10. These could be regarded as our two "best" solutions from all of the data collected; typically, in something like a Pareto front (which is not what we see here) we would expect there to be several solutions which strike a balance between both infection count and cost and could also be argued to be a "good" solution. However, given the positive incline on both major data sets when plotted on a graph, we do not have this halfway point. Instead, we might consider something like the "3rd best" slider-only solution, with a cost of 0.56547, and infection count of 169.40. Here we might decide that the slight increase in infections is worth the reduction in financial impact, but as mentioned earlier, it is primarily up to the researcher or individual working with the data to make these decisions. Nonetheless, I will take these three main solutions into account when giving my conclusion later in this paper.

4.1.5 Execution Speed Testing

Next, let us analyse the results from the execution speed tests. This is a smaller data set, due to time constraints owed from how long larger population and generation values take to run, but we can still see the general trend as the values increase. Plotting the population size against the total time (in minutes), where generations remain fixed at 20, we retrieve this scatter plot:

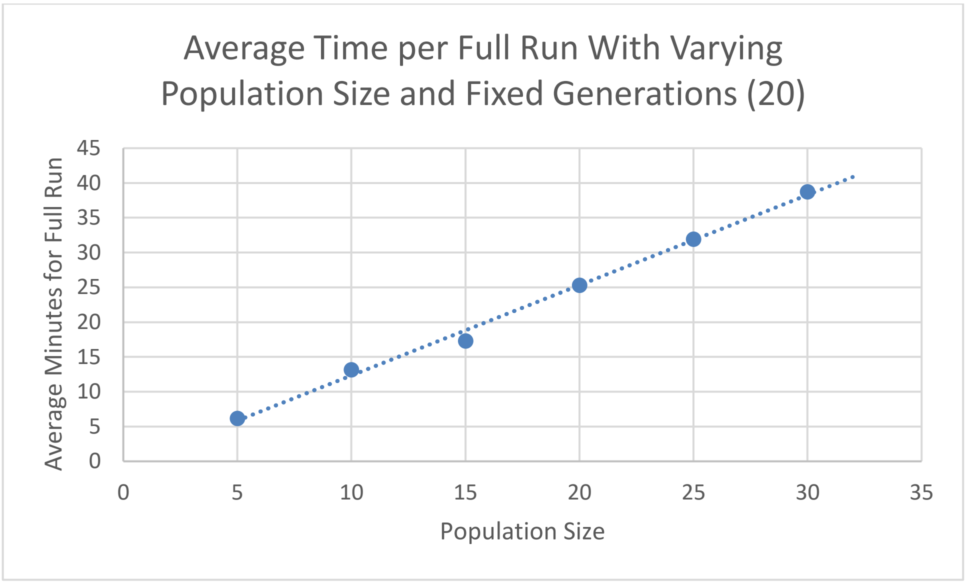


Figure 17. Scatter plot of varying population size against average execution time for a full run, with predictive trend line.

On the other hand, if we plot the generation size against the total time (in minutes), where population size is fixed at 30, we get this plot:

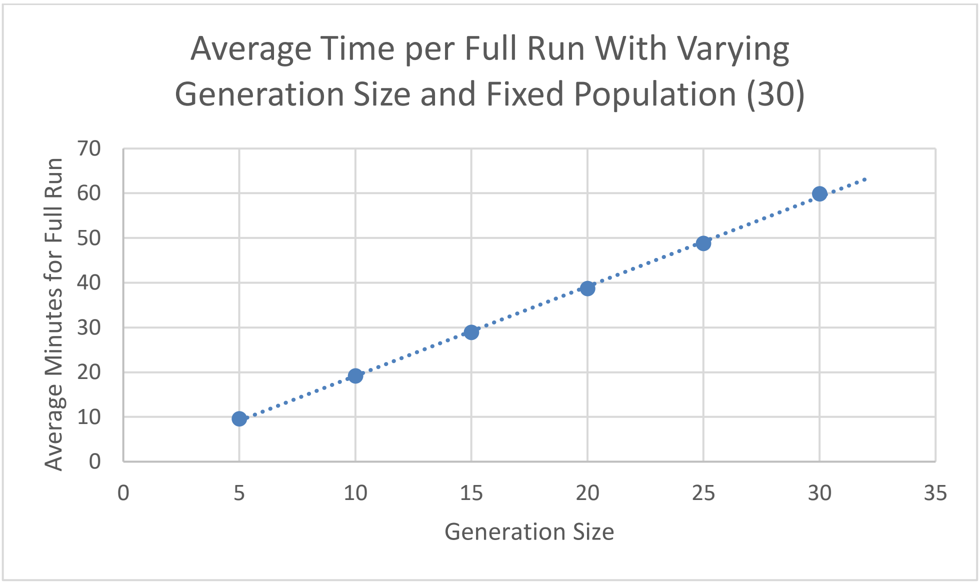


Figure 18. Scatter plot of varying generation size against average execution time for a full run, with predictive trend line.

Given the limited size of both datasets, it is difficult to come to a definite conclusion about the behaviour of the runtime as the generations and populations increase, but due to the large overhead involved once we go beyond a certain value, higher value tests were skipped. Even so, we can see that there is a straight trendline for both graphs, indicating a linear increase in time as both generation and population size increases. Although we do not have much data to work with, we can assume that this trend continues on both graphs, were we to gather results for higher population and generation values, as well as by extrapolating forward on the graph. Hence, we can reasonably (although not with certainty) assume that the overhead increase is mostly linear, rather than exponential as I had initially expected it would be. I had assumed the overhead increase would be exponential due to a misunderstanding of how the genetic algorithm worked at first, thinking that as the population or generations increased, the dimensionality and number of searches required would increase in response.

4.1.6 Final Data Analysis

After looking at all the data available from the simulation runs, we have three solutions which I have selected as good. The data for them is as follows:

#	I/O	PPE	Lockdown	Shielding	Isolation	Test and Trace	Social Distancing	Total Infections	Total Deaths	Total Cost
1	100011	0	0.59	0.45	0.15	0.02	0	226.09995	3.1	0.14159
2	N/A	0.91	0	0.92	1	0.05	0.92	133.19989	2.2	0.58135
3	N/A	0.96	0	0.99	0.99	0.74	0.89	169.39993	1.8	0.56547

Figure 19. The 3 "best" solutions, identified from the complete set of solutions from the three datasets, when plotted against one another.

The first result comes from the best solution identified in the dataset for both switches and sliders, whereas the other two come from the slider only dataset, hence they do not have a valid switch value (although it is effectively using a permanent 111111 setup for the switches). Analysing the data, we can see that the lowest number of infections comes from **solution 2**, which places heavy focus on lockdown and the test and trace program, with virtually none in PPE, shielding, social distancing, and especially lockdown. With lower threshold figures comes a higher associated cost, so we can see that the GA did not feel the investment was worthwhile for this solution yet managed to get the lowest infection count of any solution obtained (133.20), albeit the highest cost of 0.58135. On the other hand, the lowest cost was associated with **solution 1**, where heavy focus was placed on PPE, social distancing, and the test and trace system, with the other three parameters not used. Here we have a much wider spread than with **solution 2**; getting a cost of just 0.14159, but many infections, at 226.10. Comparatively, **solution 3** is the closest to a "happy medium" of the solution space, with a slightly lower cost and slightly higher infection count than **solution 2**, at 0.56547 and 169.40, respectively. As mentioned previously, if we were working with a pareto front then there would be a much larger range of solutions to choose from, but this was not the case due to the simplicity of the genetic algorithm – more complex ones can strike the balance more easily.

Of the three solutions, none are objectively the "best". There is an unavoidable trade-off between financial impact, and the impact on the livelihoods, or lives of the population. It is also crucial to consider the overlap between the two; with a greater cost comes a greater timeframe for economic recovery, and economic damage could shut down the businesses or jobs of many citizens, and indirectly cause just as many deaths as the disease, through poverty, poor living conditions, and mental health damage. Therefore, there is a dilemma between protecting the population, or protecting the economy. For this reason, if I were to give a recommendation of which solution I would select, were I a government official, I would be inclined to select **solution 3** at first, since it attempts to strike a balance between cost and infections more finely than the other two. However, it is still nowhere near the "happy medium" we would like, and according to the results obtained from this experiment, it may not be completely possible. Therefore, in the interests of protecting the futures of as many people as possible, I would select **solution 1**, which sacrifices the short-term health of the population, for longer-term economic stability. This would hopefully mitigate a lot of the damage that comes with an epidemic and keep the economy strong. As mentioned previously, there is no true, perfect solution, that solves every problem.

4.2 Evaluation of Project

4.2.1 Areas of Strength

Regarding areas of strength within the project, there are several areas in which I am pleased with the outcome:

- I am satisfied that the project was completed in its entirety; given the complex nature of the topic and the amount of work involved, I am pleased to be able to present the results and findings as a complete piece of work, despite the hurdles and time constraints faced at certain points.
- Many skills were developed throughout this project, including learning new technologies and programming languages through NetLogo, developing and refining a complex system when creating the cost function to work with the genetic algorithm, and becoming more skilled with presentation and communication skills throughout this report, and the associated poster and PowerPoint presentations.
- The cost function works extremely well for the purpose it has been created for, maintaining a balance between keeping cost and infections low, while still managing to find many "good" solutions to evaluate. There were no notable outlier solutions, and comparing with the parameter sweep above, we can see that the algorithm is able to reach its best possible solutions in a timely manner.
- The disease model works impressively well in tandem with the genetic algorithm (headlessly), and as a standalone piece of work, by accurately simulating the observed spread of coronavirus within a population, when the datasets are mapped against one another. Individuals respond and act in the manner they were programmed to, all facilities work as expected, and despite the complexity of the model, each run of the model only takes around 1.5 seconds to complete with a population of 400. All the relevant variables are tracked correctly, and graphs are updated in real-time with lots of useful information for the user, or ready to be passed to an external document.

4.2.2 Areas to Improve Upon

Despite the many successes, there are many areas in which some improvements could be made to the project:

- Given more time, more runs of the genetic algorithm should be performed, as well as with a higher population and generation value, and perhaps a heightened chance of mutated chromosomes to work with via the mutation rate parameter. This could afford the algorithm the time and space it requires to find even more accurate solutions, and improve the quality of the results obtained, as well as give us more "good" solutions to compare.
- The cost function could be fleshed out to a greater degree, both by incorporating a more complex sequence of parameters within the cost (such as assuming that those who were isolating could work from home, but at a fraction of their typical workflow, or that only those within a certain age range could have a job), and by using figures

that are more in line with real-world data, which I was unable to find reliable and accurate figures for. This would improve the accuracy of the cost function and further push the results to be more in line with what we would expect from the real world.

- At one time I had hoped to make use of R and RStudio in my project, due to time constraints this was not possible, however most of the information I needed to gather could be collected through use of Microsoft Excel instead. Given more time and more confidence with R's functionality, I would have liked to use RStudio for more detailed analysis.
- Further refinement of the NetLogo model is almost certainly possible – using NetLogo's built in profiler extension I was able to cut the execution time of an average run down by around 40% through optimisation of the code, however it was still quite slow for a program of its size, so further optimisation could be possible, perhaps by improving the way in which the program stores and handles neighbour data, which had the largest computational overhead. Since test and trace is crucial to both real-world spread prevention and the disease model itself, as one of the parameters, we cannot outright remove it, given its importance.

4.2.3 Future Work

Were this experiment to be repeated in the future, there are a few things that would be useful to expand upon or introduce:

- There are a potentially limitless number of features that could be added to the disease model which could enhance the realism and accuracy of the model, including wind direction and speed, air temperature, and pollen levels (increased sneezing), as well as other features, such as having several individuals extremely close together in an enclosed space, like work, school, or home, thereby increasing the chance of disease spread. There are many other features which could be introduced, but there is a trade-off to bear in mind between complexity and computational overhead when developing a model like this.
- I would also like to experiment with feature-based parameter estimation, and the least squares method (see section 2.3), as ways of working towards finding good solutions, and compare the advantages and disadvantages of each when used with this type of modelling project, looking at time taken, accuracy, and scope for development.
- Finally, it would be interesting to investigate how the accuracy and usefulness of the disease model changes based on the different scenarios it could be used in. For example, could we use the same model, with similar degrees of accuracy in their results, for both a densely populated area such as Glasgow or New York City, and a sparsely populated area, like a rural village in Wales? I believe it would be interesting to answer this question and gain a full understanding of the different applications of a model such as this (albeit perhaps more detailed, as discussed above).

References

[1] “WHO | Cumulative Number of Reported Probable Cases of SARS.” https://www.who.int/csr/sars/country/2003_07_11/en/ (accessed Oct. 20, 2020).

[2] “WHO Coronavirus Disease (COVID-19) Dashboard | WHO Coronavirus Disease (COVID-19) Dashboard.” https://covid19.who.int/table (accessed Oct. 21, 2020).

[3] F. Amblard *et al.*, “Introduction to NetLogo,” in *Agent-Based Spatial Simulation with NetLogo*, vol. 1, Elsevier Inc., 2015, pp. 75–123.

[4] “NetLogo 6.1.1 User Manual,” 2019. https://ccl.northwestern.edu/netlogo/docs/ (accessed Oct. 19, 2020).

[5] D. Câmara, “Evolution and Evolutionary Algorithms,” in *Bio-inspired Networking*, Elsevier, 2015, pp. 1–30.

[6] “Introduction to Evolutionary Algorithms | by Devin Soni | Towards Data Science,” 2018. https://towardsdatascience.com/introduction-to-evolutionary-algorithms-a8594b484ac (accessed Oct. 19, 2020).

[7] E. Cuevas, “An agent-based model to evaluate the COVID-19 transmission risks in facilities,” *Computers in Biology and Medicine*, vol. 121, p. 103827, Jun. 2020, doi: 10.1016/j.compbiomed.2020.103827.

[8] M. Maziarz and M. Zach, “Agent-based modelling for SARS-CoV-2 epidemic prediction and intervention assessment: A methodological appraisal,” *Journal of Evaluation in Clinical Practice*, vol. 26, no. 5, pp. 1352–1360, Oct. 2020, doi: 10.1111/jep.13459.

[9] D. K. Chu *et al.*, “Physical distancing, face masks, and eye protection to prevent person-to-person transmission of SARS-CoV-2 and COVID-19: a systematic review and meta-analysis,” *The Lancet*, vol. 395, no. 10242, pp. 1973–1987, Jun. 2020, doi: 10.1016/S0140-6736(20)31142-9.

[10] N. G. Davies *et al.*, “Effects of non-pharmaceutical interventions on COVID-19 cases, deaths, and demand for hospital services in the UK: a modelling study,” *The Lancet Public Health*, vol. 5, no. 7, pp. e375–e385, Jul. 2020, doi: 10.1016/S2468-2667(20)30133-X.

[11] A. Handel, I. M. Longini, and R. Antia, “What is the best control strategy for multiple infectious disease outbreaks?,” *Proceedings of the Royal Society B: Biological Sciences*, vol. 274, no. 1611, pp. 833–837, Mar. 2007, doi: 10.1098/rspb.2006.0015.

[12] D. Sridhar and M. S. Majumder, “Modelling the pandemic,” *The BMJ*, vol. 369. BMJ Publishing Group, Apr. 21, 2020, doi: 10.1136/bmj.m1567.

[13] Aaron A. King, “Introduction to inference: parameter estimation” https://kingaa.github.io/clim-dis/parest/parest.html (accessed Apr 03, 2020)

[14] “COVID-19: Modeling the Flattening of the Curve | Paul E. Smaldino,” 2020. http://smaldino.com/wp/covid-19-modeling-the-flattening-of-the-curve/ (accessed Oct. 19, 2020).

[15] “BCS, THE CHARTERED INSTITUTE FOR IT CODE OF CONDUCT FOR BCS MEMBERS.”

[16] “COVID-19 in Scotland – Daily Dashboard” https://public.tableau.com/profile/phs.covid.19#!/vizhome/COVID-19DailyDashboard_15960160643010/Overview (accessed Oct. 20, 2020)

5 Appendices

5.1 Disease Model Data Matching (Normalised Values - All Parameters Off)

Day	Model	Real
1	0.032895	0.00365
2	0	0
3	0.032895	0.007299
4	0.032895	0.00365
5	0.082237	0.010949
6	0.065789	0.021898
7	0.026316	0.007299
8	0.039474	0.021898
9	0.075658	0.021898
10	0.042763	0.010949
11	0.072368	0.058394
12	0.095395	0.076642
13	0.092105	0.094891
14	0.138158	0.138686
15	0.161184	0.175182
16	0.210526	0.069343
17	0.253289	0.043796
18	0.279605	0.076642
19	0.292763	0.156934
20	0.269737	0.149635
21	0.342105	0.178832
22	0.398026	0.30292
23	0.434211	0.240876
24	0.529605	0.273723
25	0.641447	0.50365
26	0.654605	0.613139
27	0.713816	0.686131
28	0.707237	0.813869
29	0.884868	1
30	1	0.748175
Euclidean Distance	0.632976	

5.2 Disease Model Data Matching (Normalised Values - All Parameters On)

Day	Model	Real
1	0	0.001289
2	0.001462	0
3	0.01462	0.002577
4	0.019006	0.001289
5	0.02193	0.003866
6	0.027778	0.007732
7	0.023392	0.002577
8	0.026316	0.007732
9	0.016082	0.007732
10	0.01462	0.003866
11	0.035088	0.020619
12	0.02924	0.027062
13	0.045322	0.033505
14	0.061404	0.048969
15	0.065789	0.061856
16	0.081871	0.024485
17	0.137427	0.015464
18	0.109649	0.027062
19	0.128655	0.055412
20	0.179825	0.052835
21	0.163743	0.063144
22	0.201754	0.106959
23	0.22076	0.085052
24	0.26462	0.096649
25	0.295322	0.177835
26	0.312865	0.216495
27	0.353801	0.242268
28	0.432749	0.287371
29	0.508772	0.353093
30	0.608187	0.264175
31	0.602339	0.253866
32	0.479532	0.412371
33	0.763158	0.438144
34	0.744152	0.5
35	0.671053	0.45232
36	0.858187	0.488402
37	1	0.333763
38	0.716374	0.289948
39	0.608187	0.520619
40	0.668129	0.45232
41	0.616959	0.44201
42	0.602339	0.391753

43	0.549708	0.423969
44	0.421053	0.238402
45	0.453216	0.283505
46	0.397661	0.462629
47	0.346491	0.445876
48	0.397661	0.426546
49	0.305556	0.550258
50	0.277778	0.507732
51	0.283626	0.340206
52	0.27193	0.204897
53	0.269006	1
54	0.302632	0.514175
55	0.280702	0.438144
56	0.267544	0.404639
57	0.251462	0.393041
58	0.261696	0.237113
59	0.273392	0.221649
60	0.289474	0.554124
61	0.277778	0.489691
62	0.258772	0.554124
63	0.273392	0.431701
64	0.267544	0.474227
65	0.292398	0.279639
66	0.226608	0.243557
67	0.244152	0.425258
68	0.236842	0.372423
69	0.26462	0.39433
70	0.25	0.373711
Euclidean Distance	1.660696	

5.3 Genetic Algorithm Results (Only Switches in Search Space)

#	I/O	Avg. Daily Infections	Total Infections	Avg. Daily Deaths	Total Deaths	Total Cost	Infection Cost	Control Procedure Cost
1	100011	7.95397	2903.2002	0.09397	34.30003	0.49391	0.38893	0.10499
2	100011	7.98109	2913.09961	0.09452	34.50004	0.49644	0.39144	0.105
3	100011	7.92795	2893.7002	0.0937	34.20003	0.49228	0.38729	0.10499
4	100011	7.86411	2870.40015	0.09808	35.80002	0.49757	0.39257	0.105
5	100011	7.88219	2876.99951	0.09808	35.80001	0.49935	0.39435	0.105
6	100011	7.92082	2891.09985	0.09562	34.90002	0.49595	0.39095	0.105
7	100011	7.98904	2916.00073	0.09342	34.10003	0.49491	0.38991	0.105
8	100011	7.87726	2875.20068	0.09781	35.70002	0.49765	0.39265	0.105
9	100011	8.04521	2936.50146	0.09452	34.50002	0.49715	0.39216	0.10499
10	100011	7.97261	2910.00146	0.09425	34.40003	0.49545	0.39046	0.105
11	100011	7.73041	2821.59985	0.1	36.50002	0.49817	0.39316	0.10501
12	100011	8.06082	2942.19995	0.09425	34.40003	0.49764	0.39265	0.10499
13	100011	8.02849	2930.39966	0.09644	35.2	0.50085	0.39585	0.105
14	100011	7.97918	2912.39966	0.09397	34.30003	0.49555	0.39056	0.10499
15	100011	7.73096	2821.80103	0.09507	34.70002	0.48993	0.38493	0.10501
16	100011	7.97178	2909.69946	0.09507	34.70004	0.49643	0.39144	0.105
17	100011	7.79891	2846.60107	0.09507	34.70003	0.49441	0.38939	0.10502
18	100011	7.85973	2868.7998	0.09589	35.00003	0.49375	0.38876	0.10499
19	100011	7.83123	2858.3999	0.09918	36.20001	0.50034	0.39533	0.10501
20	100011	7.91726	2889.79883	0.09671	35.30002	0.49815	0.39315	0.105

5.4 Genetic Algorithm Results (Only Threshold Sliders in Search Space)

Run	PPE	Lockdown	Shielding	Isolation	Test and Trace	Social Distancing	Average Daily Infections	Total Infections	Average Daily Deaths	Total Deaths	Total Cost	Infection Cost	Control Procedure Cost
1	0.96	0.01	0.93	0.93	0.02	0.79	2.15945	788.19971	0.02548	9.3	0.66638	0.10309	0.56328
2	0.93	0	0.91	0.97	0.01	0.88	0.67068	244.80002	0.00822	3	0.59753	0.03233	0.5652
3	0.01	0.97	0.94	0.91	0.01	0.01	2.16493	790.2002	0.02849	10.4	0.26154	0.10773	0.15381
4	0.01	0.96	0.89	0.82	0	0.03	3.50082	1277.80103	0.05123	18.70002	0.36563	0.18394	0.18169
5	0.97	0	0.93	0.91	0.57	0.84	0.99479	363.10004	0.01233	4.5	0.61648	0.04844	0.56804
6	0.01	0.81	0.85	0.94	0.03	0.02	2.54849	930.20001	0.03534	12.90001	0.37761	0.12947	0.24814
7	0.01	0.94	0.96	0.94	0.76	0.03	3.95068	1441.99976	0.05123	18.70002	0.35442	0.19724	0.15718
8	0.05	0.95	1	0.91	0.1	0.07	8.09014	2952.8999	0.10932	39.90001	0.56858	0.41912	0.14945
9	0.01	0.89	0.91	0.73	0.02	0.01	1.62849	594.40015	0.02137	7.79999	0.31342	0.08074	0.23268
10	0.99	0	0.87	0.93	0.18	0.98	0.36548	133.39986	0.00685	2.5	0.59234	0.02113	0.57122
11	0.99	0	0.86	0.95	0.89	0.96	0.52164	190.40001	0.00466	1.7	0.59219	0.02252	0.56967
12	0.81	0.01	0.97	0.97	0.85	0.99	1.53534	560.3999	0.01863	6.8	0.63307	0.07424	0.55883
13	0.96	0	0.92	0.96	0.85	0.94	0.95123	347.19998	0.01041	3.8	0.60291	0.04414	0.55877
14	0.04	1	0.93	0.99	0	0.7	7.16986	2617	0.09315	34.00003	0.47261	0.36474	0.10787
15	0.05	0.94	0.95	0.98	0	0.08	8.3526	3048.69971	0.09507	34.70002	0.55215	0.40339	0.14876
16	0.03	0.97	0.99	1	0.09	0.02	4.88164	1781.80029	0.06384	23.30003	0.37027	0.24527	0.125
17	0.96	0	0.99	0.99	0.74	0.89	0.46411	169.39993	0.00493	1.8	0.56547	0.02127	0.5442
18	0.01	1	1	0.85	0.26	0.04	5.33288	1946.49963	0.0663	24.20004	0.40034	0.26362	0.13673
19	0.03	0.9	0.94	0.94	0.06	0.05	6.12	2233.7998	0.07671	28.00004	0.48518	0.3039	0.18128
20	0.77	0.01	0.89	0.9	0.59	0	2.32767	849.59961	0.03479	12.70001	0.72741	0.12211	0.60531
21	0.07	0.98	0.78	0.98	0.07	0.03	8.8663	3236.19971	0.11014	40.2	0.60245	0.4454	0.15705
22	0.91	0	0.92	1	0.05	0.92	0.36493	133.19989	0.00603	2.2	0.58135	0.01988	0.56147

23	0	0.94	0.94	0.94	0.02	0.01	1.66767	608.70038	0.01918	7	0.24144	0.07889	0.16255
24	0.02	0.98	0.92	0.89	0.02	0.02	3.29616	1203.09961	0.04795	17.50002	0.32654	0.1721	0.15444
25	0.02	0.99	0.9	0.9	0.46	0.03	4.5189	1649.39941	0.06849	25.00003	0.38909	0.24283	0.14625
26	0.05	0.99	0.79	0.9	0.01	0.01	7.53672	2750.9021	0.10247	37.40001	0.55577	0.38937	0.1664
27	0.02	0.95	0.95	0.92	0.01	0.02	4.36247	1592.2998	0.05616	20.50002	0.37256	0.21749	0.15507
28	0.02	0.9	0.9	0.92	0.04	0.06	5.09096	1858.20032	0.07041	25.70004	0.45375	0.26202	0.19173
29	0.01	0.97	0.98	0.96	0.01	0	1.25014	456.29993	0.0137	5	0.19254	0.05822	0.13432
30	0.01	0.96	0.94	0.98	0.15	0.01	1.80438	658.59985	0.02274	8.3	0.23366	0.08836	0.14529

5.5 Genetic Algorithm Results (Both Switches and Threshold Sliders in Search Space)

Run	I/O	PPE	Lockdown	Shielding	Isolation	Test and Trace	Social Distancing	Average Daily Infections	Total Infections	Average Daily Deaths	Total Deaths	Total Cost	Infection Cost	Control Procedure Cost
1	100010	0.04	0.84	0.52	0.34	0.06	0.6	7.25041	2646.39844	0.08986	32.80006	0.44654	0.36114	0.0854
2	100010	0	0.62	0.96	0.92	0	0.32	3.31808	1211.10034	0.05041	18.40002	0.26603	0.17793	0.0881
3	100011	0	0.55	0.24	0.66	0.54	0	1.46685	535.39978	0.01808	6.6	0.17899	0.07065	0.10834
4	100010	0.01	0.02	0.08	0.75	0.08	0.14	5.86	2138.90015	0.07726	28.20004	0.38463	0.29684	0.08779
5	100011	0.02	0.93	0.02	0.05	0.89	0.03	5.00082	1825.3009	0.06329	23.10003	0.35433	0.24919	0.10514
6	100011	0.03	0.58	0.4	0.73	0.76	0.04	5.80849	2120.09961	0.07918	28.90005	0.40433	0.2995	0.10482
7	100110	0.03	0.87	0.39	0.98	0.03	0.08	6.28493	2293.99976	0.08274	30.20004	0.4099	0.31932	0.09058
8	100011	0.09	0.68	1	0.31	0.02	0.06	10.83918	3956.30103	0.13342	48.69997	0.6492	0.5477	0.1015
9	100010	0.05	0.8	0.65	0.96	0.12	0.91	9.26576	3382.00293	0.11288	41.19999	0.5459	0.46187	0.08403
10	100011	0.01	0.24	0.04	0.31	0.09	0.07	6.72466	2454.49927	0.0874	31.90005	0.44806	0.3404	0.10766

11	100001	0.01	0.84	0.57	0.52	0.39	0.02	3.56794	1302.2998	0.04	14.60001	0.27426	0.16804	0.10622
12	100001	0	0.32	0.3	0.32	0.26	0	1.16137	423.90009	0.01397	5.1	0.16262	0.05558	0.10705
13	100010	0	0.54	0.77	0.91	0.04	0.43	4.96302	1811.50098	0.06849	25.00004	0.34434	0.25592	0.08842
14	100011	0.01	0.5	0.24	0.69	0.06	0.05	4.34	1584.10022	0.06219	22.70003	0.33571	0.2272	0.10851
15	100011	0	0.92	0.24	0.3	0.48	0.01	0.92521	337.70004	0.01288	4.7	0.15552	0.0468	0.10872
16	010000	0.51	0.01	0.65	0.56	0.47	0.63	1.38822	506.69998	0.01699	6.2	0.59761	0.06715	0.53047
17	100011	0.03	0.83	0.13	0.78	0	0.06	5.45781	1992.09973	0.06712	24.50004	0.37439	0.26756	0.10683
18	010000	0.3	0.02	0.34	0.55	0.21	0.39	3.52	1284.79919	0.04438	16.20001	0.69874	0.17392	0.52482
19	100001	0.04	0.39	0.04	0.92	0.09	0	5.81809	2123.60107	0.06822	24.90004	0.38641	0.28208	0.10433
20	100010	0	0.75	0.54	0.86	0.01	0.86	3.15808	1152.69971	0.03945	14.40001	0.24354	0.15532	0.08822
21	100010	0.02	0.06	0.72	0.05	0.12	0.33	7.32301	2672.89893	0.10384	37.9	0.47058	0.38449	0.08608
22	010000	0.33	0.01	0.35	0.15	0.94	0.71	3.05151	1113.79993	0.03425	12.50001	0.6699	0.14416	0.52574
23	100010	0	0.59	0.58	0.13	0.07	0.69	5.46849	1995.99988	0.06548	23.90002	0.35427	0.26617	0.0881

24	100001	0.01	0.52	0.59	0.04	0.23	0	1.23945	452.40027	0.01507	5.5	0.16681	0.05988	0.10693
25	100011	0.04	0.12	0.38	0.86	0.25	0.03	6.74986	2463.70044	0.08192	29.90005	0.43819	0.3324	0.10579
26	100011	0.01	0.27	0.94	0.04	0.05	0.03	3.74356	1366.39966	0.05041	18.40001	0.29866	0.18984	0.10882
27	010000	0.47	0.02	0.99	0.77	0.33	0.14	4.04822	1477.6001	0.05123	18.70002	0.72251	0.19944	0.52307
28	010010	0.83	0.01	0.72	0.43	0.16	0.81	3.16027	1153.5	0.03068	11.20001	0.67038	0.14147	0.52891
29	100001	0.04	0.01	0.24	0.3	0.81	0.05	7.88356	2877.5	0.08877	32.40004	0.47897	0.37608	0.10288
30	100011	0	0.59	0.45	0.15	0.02	0	0.61945	226.09995	0.00849	3.1	0.14159	0.03121	0.11038

5.6 Model Execution Speed Tests

10 pop, 20 gen	
Generation	**Time**
1	38.117836
2	38.117836
3	38.923141
4	38.654705
5	38.654705
6	39.460011
7	40.26532
8	39.996883
9	39.996883
10	39.996883
11	40.26532
12	39.996883
13	39.996883
14	39.728447
15	39.996883
16	39.728447
17	39.191578
18	38.386269
19	38.386269
20	38.923141

Total (secs)	786.784323
Total (mins)	13.11307205
Population	10
Generations	20

20 pop, 20 gen	
Generation	**Time**
1	73.282883
2	73.551315
3	74.088188
4	74.625053
5	74.625053
6	75.698799
7	77.040977
8	78.114716
9	77.57785
10	74.088188
11	74.625053
12	74.088188
13	75.698799
14	77.040977
15	77.040977
16	77.30941
17	77.040977
18	76.772537

19	76.772537
20	76.772537
Total (secs)	1515.855014
Total (mins)	25.26425023
Population	20
Generations	20

30 pop, 5 gen	
Generation	**Time**
1	110.595406
2	111.937584
3	114.621941
4	116.500984
5	120.259087
Total (secs)	573.915002
Total (mins)	9.565250033
Population	30
Generations	5

30 pop, 10 gen	
Generation	**Time**
1	111.400711

2	114.890373
3	114.621941
4	115.964119
5	117.037857
6	116.500984
7	117.037857
8	116.232552
9	115.964119
10	113.279762
Total (secs)	1152.930275
Total (mins)	19.21550458
Population	30
Generations	10

30 pop, 20 gen	
Generation	**Time**
1	110.863846
2	119.185341
3	109.253227
4	110.05854
5	113.01133
6	118.916908
7	118.648468
8	118.380035

9	117.843163
10	114.890373
11	114.085068
12	117.57473
13	117.57473
14	115.427246
15	114.890373
16	118.648468
17	119.990646
18	120.259087
19	116.232552
20	115.695679
Total (secs)	2321.42981
Total (mins)	38.69049683
Population	30
Generations	20

15 pop, 20 gen	
Generation	**Time**
1	55.02927
2	53.687092
3	50.734303
4	51.002735
5	50.734303

6	51.271172
7	51.271172
8	51.002735
9	52.076477
10	51.808044
11	51.539608
12	52.344913
13	51.808044
14	51.808044
15	51.808044
16	51.808044
17	51.539608
18	52.076477
19	51.539608
20	51.539608
Total (secs)	1036.429301
Total (mins)	17.27382168
Population	15
Generations	20

25 pop, 20 gen	
Generation	**Time**
1	92.87867
2	93.147102

3	94.489281
4	94.489281
5	94.489281
6	94.489281
7	94.757713
8	94.489281
9	93.952408
10	93.952408
11	93.952408
12	93.952408
13	93.952408
14	94.757713
15	96.636765
16	97.978943
17	100.126427
18	100.126427
19	100.6633
20	101.468605
Total (secs)	1914.75011
Total (mins)	31.91250183
Population	25
Generations	20

30 pop, 15 gen	

Generation	Time
1	110.863846
2	112.742889
3	112.742889
4	113.279762
5	114.085068
6	114.085068
7	115.427246
8	116.769424
9	116.500984
10	116.500984
11	117.843163
12	118.380035
13	118.916908
14	118.648468
15	118.916908
Total (secs)	1735.703642
Total (mins)	28.92839403
Population	30
Generations	15

30 pop, 25 gen	
Generation	**Time**
1	111.669151

2	111.132278
3	111.400711
4	110.863846
5	111.132278
6	111.400711
7	113.01133
8	113.279762
9	115.158813
10	114.890373
11	117.037857
12	117.843163
13	119.722214
14	120.259087
15	120.527519
16	121.332825
17	121.332825
18	119.722214
19	120.795952
20	120.527519
21	121.601265
22	119.990646
23	121.332825
24	120.795952
25	119.722214

Total (secs)	2926.48333
Total (mins)	48.77472217
Population	30
Generations	25

30 pop, 30 gen	
Generation	**Time**
1	110.863846
2	113.279762
3	115.964119
4	115.158813
5	114.353508
6	115.695679
7	117.306297
8	118.380035
9	118.916908
10	117.843163
11	120.795952
12	120.527519
13	120.259087
14	119.990646
15	120.527519
16	119.453781
17	121.064392
18	120.527519

19	121.332825
20	121.064392
21	121.064392
22	121.332825
23	122.675003
24	123.480309
25	123.211876
26	123.480309
27	123.748749
28	123.748749
29	123.480309
30	123.748749
Total (secs)	3593.277032
Total (mins)	59.88795053
Population	30
Generations	30

5 pop, 5 gen	
Generation	**Time**
1	18.790482
2	18.253611
3	18.253611
4	18.522047
5	18.522047
6	18.522047

7	18.253611
8	18.522047
9	18.522047
10	18.522047
11	18.522047
12	18.522047
13	18.253611
14	18.522047
15	18.253611
16	18.522047
17	18.253611
18	18.522047
19	18.253611
20	18.522047
Total (secs)	368.830323
Total (mins)	6.14717205
Population	5
Generations	20

5.7 Parameter Sweep Results (Sorted by Total Cost)

Note: the order of switches in a solution is as follows: *PPE, Lockdown, Shielding, Isolation, Test & Trace, Social Distancing*.

Solution	Total Cost
100011	0.51921016
100010	0.569595
100001	0.5858147
101011	0.6575537
101010	0.69699144
100111	0.6985221
101001	0.7215911
100101	0.73398805
100000	0.7378835
100110	0.7566692
101000	0.81714845
000011	0.8579145
100100	0.85887223
101111	0.86320543
101110	0.89196855
101101	0.9019226
001011	0.93608004
110011	0.96983325
101100	0.98093873
110010	0.99204016
000001	0.992562
000010	1.031517
000111	1.034139
000000	1.0341856
110001	1.0367824
001001	1.0811596
001010	1.1000978
001111	1.1033086
110000	1.1335793
111011	1.136676
000101	1.1461351
110111	1.169578
111001	1.171652
111010	1.1779543
000110	1.1895338
110110	1.2204105
010001	1.2211437
010011	1.2219784

001000	1.2221944
110101	1.2236242
001101	1.2422248
000100	1.2497965
001110	1.2595264
111000	1.2764542
010010	1.2884358
110100	1.2911018
010000	1.2965683
111111	1.3216484
011011	1.334326
011001	1.3513821
111101	1.3612144
010111	1.3644173
011010	1.3776051
010101	1.3788754
111110	1.3860581
001100	1.3972414
010110	1.4044402
111100	1.4062084
010100	1.4636979
011000	1.463775
011111	1.5011889
011101	1.5249575
011110	1.5308486
011100	1.5902685