A Model-Based Approach To Assess Epidemic Risk

Hugo Dolan

July 31, 2020

Abstract

In recent years, the extensive development of the transportation infrastructure has radically changed how connected our world is. In today's "small-world" we can travel around the globe in a matter of days, if not hours. This has important implications on international security, especially in regard to the potential spread of pandemic diseases. The recent COVID-19 outbreak has forced many countries to take drastic measures to contain and slow down the spread of the virus. Due to the urgency of the situation, a number of countries have immediately reacted by imposing lockdowns and closing borders. While these measures have been successful in temporarily confining the epidemics, this immediate and chaotic response has blurred the actual role played by the topology of the infrastructure network on the spread of the virus. The main goal of this paper is to create a model-based framework that can inform decision making regarding flight connection closures as a means to slowing down a potential pandemic without causing excessive economic damage. In particular, we introduce a new framework to study networks of international flights as potential vehicles for the spread of pandemic diseases. First, we propose an in-depth analysis of the OpenFlights network dataset, which describes a large number of flight connections between airports. Then, we use this real infrastructure network to create a model for the simulation of epidemics. Our model combines existing graph diffusion processes and SEIRS compartmental models to study the effect of international travel on the spread of a disease within communities, both locally and globally. We run a number of simulations to characterise possible real scenarios, and to test the sensitivity of our model. Finally, we use our framework to explore several mitigation strategies on the network, and employ genetic algorithms to optimise one such strategy in order to minimise the spread of disease whilst preserving global commerce.

1 Introduction

In recent years, the extensive development of the air transportation infrastructure has radically changed how connected our world is. In today's "small-world" we can travel around the globe in a matter of days, if not hours. This has important implications on international security, especially in regard to the potential spread of pandemic diseases. The recent 2019-nCov outbreak has forced many countries to take drastic measures to contain and slow down the spread of the virus. Due to the urgency of the situation, several countries have immediately reacted by quarantining infected individuals / regions and cancelling flights. It may be argued that these preventive measures are often taken without full awareness of the effective pandemic risk, or without a formal modelling framework.

The goal of this project is to study a network of international flight routes and how its topology may play a role in the spread of disease. An essential aspect of this project is the development of a statistical network model that can take into account the flight routes data and the distribution of the infected population. The outcome of the project is to give a model-based quantification of pandemic risk, and to identify effective interventions within the network. Due to the large size of the dataset (3425 Nodes) and due to the complexity of the data (37,594 Edges), the project will involve extensive numerical simulation and optimisation approaches.

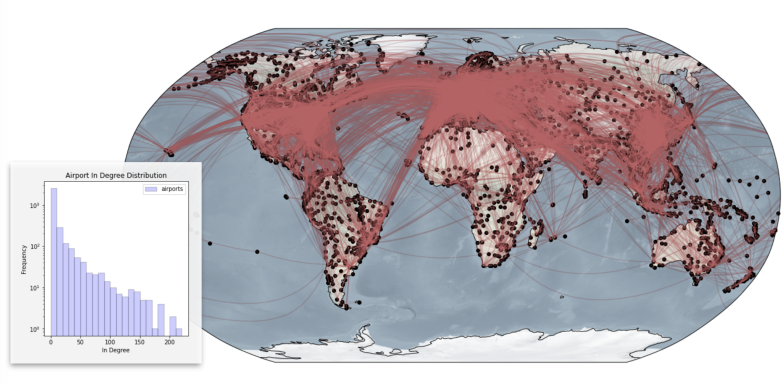


Figure 1: Open Flights Network Degree Distribution & Visualisation.

In Section 2 we first examine the topology of the Open Flights network via a selection of summary statistics to identify which nodes may be important to the network's connectivity. Furthermore we fit a Block Stochastic Model to the network in order identify latent community structures. Finally we employ a Percolation Algorithm to ascertain the resilience of the network. These probes of the network topology will highlight the key challenges in devising subsequent mitigation strategies. In Section 3 on completion of our exploratory analysis we then seek to model the evolution of the epidemic through the flight network by combining a network diffusion model (from the domain of applied mathematics) and SEIRS model (from epidemiology) to create a hybrid model capable of capturing the propagation of the virus through the network as well as the dynamics of local community spread. Finally we discuss the key parameters and assumptions of the model. In Section 4 our initial approach is to visualise the unmitigated spread of a virus through the network, given parameters estimated from real world data in addition to several heuristics. Subsequently we explore the evolution of the unmitigated spread over time and perform a sensitivity analysis on key parameters of the simulation. We then proceed to select metrics which capture the overall behaviour of the system. Having done so we then implement and test several mitigation strategies and utilise chosen metrics to assess the impact of the proposed strategies on the spread of the virus. Using these findings we create then utilise a Genetic Algorithm to find the near optimal choice of parameters for the chosen mitigation strategy. Finally we examine the behaviour of this strategy when employed on theoretical epidemics with different origins across the globe.

2 Network Topology

In this analysis we utilise the Open Flights dataset, it contains information on 3425 airports globally, including a database of 37,594 routes between airports aggregated by airline. The dataset is transformed into an adjacency matrix with nodes representing airports and edges representing routes between them. In Figure 1 we present the network visually and on initial inspection it is clear that the network exhibits extremely high degree of connectivity, with the plot of degree distribution indicating that over 20% of nodes with degree greater than 10.

Identifying airports which have are important to the overall connectivity of the network is crucial in gaining a better understanding of the network's topology. We consider several metrics for importance including PageRank, Betweenness, Coreness as well as the In and Out Degree of nodes, and present a table of the 20 most important airports according to by Page Rank (Table 1). Airports with high Page Rank are also major international destinations and extremely well connected with

Table 1: Summary Statistics for Top 20 Airports

Airport	PageRank	Betweeness	Coreness	deg(+)	deg(-)	Country
ATL	0.0047	0.0294	60	216	**217**	United States
IST	0.0044	**0.0412**	62	230	**227**	Turkey
ORD	0.0043	**0.0474**	60	203	**206**	United States
DEN	0.0043	0.0262	60	168	169	United States
DFW	0.0042	0.0251	60	185	187	United States
DME	0.0041	0.0294	62	189	189	Russia
CDG	0.0039	**0.0617**	62	233	**237**	France
FRA	0.0038	**0.051**	62	238	**239**	Germany
PEK	0.0038	**0.0492**	60	206	**206**	China
DXB	0.0036	**0.0594**	62	182	188	United Arab Emirates
AMS	0.0036	**0.0427**	62	231	**232**	Netherlands
IAH	0.0035	0.023	60	168	169	United States
LAX	0.0033	**0.0662**	60	148	149	United States
SYD	0.0032	0.0326	45	83	85	Australia
YYZ	0.003	**0.0425**	60	146	147	Canada
JFK	0.003	0.0258	62	160	162	United States
BOG	0.0029	0.0248	40	74	74	Colombia
PVG	0.0029	0.0221	56	153	152	China

a coreness of over 60, meaning that they are members of a set of airports in which every airport is connected to at least 60 others in the set! It is interesting to note that airports with high betweenness (Charles De Gaulle, Dubai, Beijing, Amsterdam, Los Angeles, Toronto, Frankfurt) are also major connecting flight hubs, with the having degrees of over 200. Whilst there are some discrepancies between in and out degree, this is likely due to missing data, as most airlines today operate return flights, with only a few exceptions.

To identify community structure within the network we employ a Stochastic Block Model, utilising the efficient inference method described by Daudin et al, 2008 [1]. It is clear that the communities found by the model represent geographic clusters (Figure 2a). This is quite surprising as this information is not encoded explicitly in the data provided to the algorithm. This would strongly suggest a high degree of connectivity of airports not only globally but also within regions. We also note from the dot plot in Figure 2b that the majority of connections are within relatively large communities representing the geographic clustering observed in Figure 2a, but also towards the lower right corner there is significant disassortative behaviour, likely these nodes are large international hubs such as the small community of London, Frankfurt, Amsterdam, Charles De Gaulle which share connections to many cities across the world.

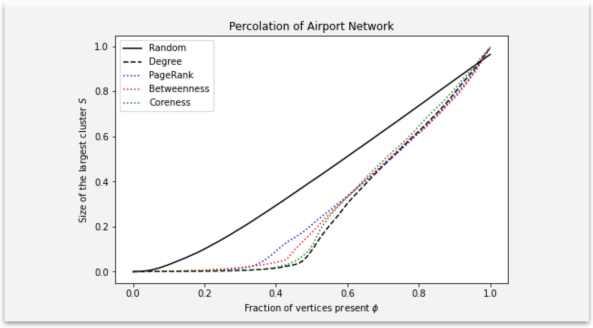
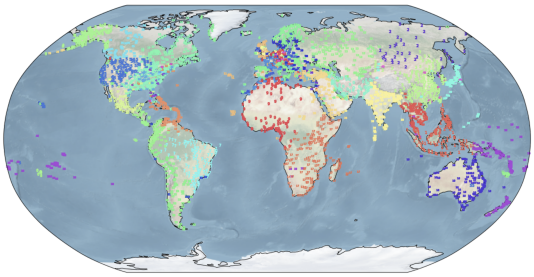
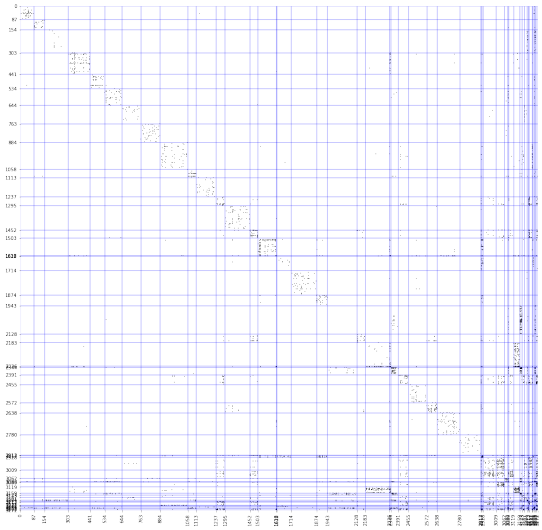


Figure 3: Percolation of Airports Networks via a variety of ranking criteria (Results average over 100 trials)



(a) Visualisation of Block Stochastic Communities



(b) Adjacency Matrix Dot Plot with 48 Communities

We percolate the Airport network both randomly and by degree thresholding to simulate the removal of airports from the network (Fig 3). The network is highly resilient to random attacks required the removal of almost all nodes to break network connectivity in the Giant Component. However the network is moderately more vulnerable to targeted (degree based) attacks, yet would still require more than half of all airports to be removed for connectivity to be broken. Similarly percolation is performed by other ranking factors (PageRank, Betweenness, Coreness), note that these procedures are also averaged over many trials to account for the removal of vertices of equal rankings in different orders, however the results are very quite similar to percolation by degree.

In conclusion the Open Flights network summary statistics show that airports which are large regional destinations or hubs for connecting flights tend to have high importance to network connectivity. Furthermore it is observed that nodes in the network are extremely well connected, both at regional and global level with significant geographical community structure. The network is also highly resilient to the most forms of percolation. Given these initial findings, it is rather unsurprising that a virus such as 2019-nCov could spread globally in a matter of months. The poor handling of this recent crisis by governments globally and potential for future similar outbreaks or resurgences illustrates the need for a model-based approach. This will aid in the evaluation of potential interventions, so that policy makers can make, effective and justifiable decisions to best protect their citizens

and economies from catastrophic damage.

3 Model Specification

3.1 Theoretical Underpinnings

Before we develop the main model of this report we must first introduce two existing models which can be found in the domains of applied mathematics and epidemiology. Firstly we specify the graph diffusion model which describes the flow of a fluid across a network, driven by pressure differences between adjacent nodes. This can be expressed as a vector of differential equations denoting changes of fluid volumes at each node and time step (Equation 1). We use the notation $\boldsymbol{\psi}$ to represent the vector of fluid volumes at every node, $\boldsymbol{A}$ to denote the adjacency matrix of the network and $\boldsymbol{D}$ to denote a diagonal matrix of containing the degrees of every node. A full derivation of this can be found in [3].

$$\frac{d\boldsymbol{\psi}}{dt} = \boldsymbol{c(A - D)\psi}$$

Additionally we introduce the SEIRS Compartmental epidemiology model. Each letter of the model name denotes a compartment of the system (Susceptible (S), Exposed (E), Infectious (I) and Recovered (R)), in which some number of individuals from the total population (M) reside. Figure 4. illustrates the direction of progression from state to state, whilst equation 2 indicates the exact rates at which the population in each compartment changes. The greek letters are constants of the system which can be fitted to match the characteristics of some observed epidemic. Whilst the system cannot be solved analytically, we can find a numerical approximation to the solution, which is sufficient for our simulation purposes.

$$\frac{\boldsymbol{dS}}{\boldsymbol{dt}} = \boldsymbol{\delta R} - \frac{\boldsymbol{S\beta I}}{\boldsymbol{M}}$$

$$\frac{\boldsymbol{dE}}{\boldsymbol{dt}} = \frac{\boldsymbol{S\beta I}}{\boldsymbol{M}} - \boldsymbol{\epsilon E}$$

$$\frac{\boldsymbol{dI}}{\boldsymbol{dt}} = \boldsymbol{\epsilon E} - \boldsymbol{\gamma I}$$

$$\frac{\boldsymbol{dR}}{\boldsymbol{dt}} = \boldsymbol{\gamma I} - \boldsymbol{\delta R}$$

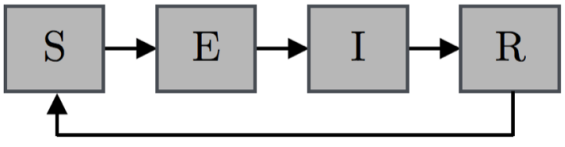


Figure 4: SEIRS Compartmental Flow Diagram

3.2 Model Definition

In order to model the transmission of disease through international flight networks we opt to use the SEIRS model combined with a Graph Diffusion model as described in the previous section. We will refer to the airports adjacency matrix as A and denote airport nodes as $\boldsymbol{v_j}$; $\boldsymbol{j = 1, \ldots, N}$. The total number of node in the network is $\boldsymbol{N}$ and the associated population at each node is $\boldsymbol{M_j}$. Let us define the local epidemic state vector as $\boldsymbol{\theta_j(t) = (S_j\ E_j\ I_j\ R_j)^T}$, which represent the compartments of the SEIRS model for airport population at any given time. A condition of the SEIRS model constrains the total population of all compartments to equal the total population:

$$\boldsymbol{S_j(t) + E_j(t) + I_j(t) + R_j(t) = M_j}$$

We assume that the local population is fully mixed (ie. everyone has equal chance of being infected), as this is a standard assumption of compartmental epidemic models, however we assume this only to be true at individual airport level and not for the entire global system. Additionally let $\boldsymbol{\alpha_j}$ be the proportion of the population which can afford to travel and $\boldsymbol{c}$ be probability that an individual departs from an airport on any given day. Thus we define an additional variable $\boldsymbol{\psi_j(t) = \alpha_j \theta_j(t)}$ and refer to this as the mobile epidemic state. We can define at high level our simulation procedure:

1. Update the local epidemic state $\boldsymbol{\theta}$ by performing one step of the SEIRS model for every airport. This describes the spread of the virus within the community surrounding the airport.

2. Take this new local epidemic state $\boldsymbol{\theta^*}$ and split it into the base population $\boldsymbol{\theta_B}$ who are permanent residents to the local area and $\boldsymbol{\theta_T}$, the transient population who are temporary visitors (eg. on business / holidays)

3. Compute the proportions of $\boldsymbol{\theta_T}$ and $\boldsymbol{\theta_T}$ who can afford to fly

4. Using our diffusion model compute the changes to $\boldsymbol{\theta_B}$ and $\boldsymbol{\theta_T}$ at each airport. The exact value of these changes is based on several factors including, the differences between outbound and returning passengers at every connected airport, the relative importance of the airports in the network and the node's degree)

5. Recombine updated values of $\boldsymbol{\theta_B}$ and $\boldsymbol{\theta_T}$ into $\boldsymbol{\theta}$ and loop for as many iterations of simulation as required.

For completeness we include both a diagrammatic form of the algorithm (Figure 5) as well as a the full algorithm (Table 2) which reflects the high level overview above. For a full derivation visit the appendix. For those who are happy with understanding our algorithm at a high level, we have mapped the steps above to the algorithm below (in the right hand margin). The algorithm described above has been vectorised so that $\boldsymbol{\theta}$ is now 4xN a matrix containing the states for all airports $\boldsymbol{\theta(t) = (\theta_1(t) \ldots \theta_N(t))}$, and similarly for $\boldsymbol{\psi^*_+}$, $\boldsymbol{\psi^*_-}$ which represent the states of outbound and returning travellers respectively. The matrix $\boldsymbol{1/D}$ is a diagonal matrix with entries that are $\boldsymbol{1/deg(v_j)}$, this normalises outflows from airports preventing more passengers from leaving a node than exist at the given node (This fix is necessary due to simultaneous outflows in a vectorised algorithm). The matrix $\boldsymbol{B}$ is an operator encoding the differential equations of the SEIRS model for vectorised application to many airports simultaneously, whilst is a modified version of $\boldsymbol{\theta}$ to enable $\boldsymbol{B}$ be applied as a linear operator. Finally $\boldsymbol{C}$ is a weighted version of adjacency matrix $\boldsymbol{A}$, so that the outflows from vertices reflect the relative importance of adjacent airports. The above should provide an intuition, however we reiterate that there is a far more comprehensive derivation of the mathematics required in the appendix, accompanied a Python implementation.

Table 2: Simulation Psuedocode

	Algorithm: Flow & Degree Corrected Epidemic Diffusion Model	
	for t = 1 , ..., T:	
(5)	$\vartheta(t\text{-}1) = \vartheta_B(t\text{-}1) + \vartheta_T(t\text{-}1)$	
(1)	$\vartheta^*(t\text{-}1) = \vartheta(t\text{-}1) + \boldsymbol{B\tilde{\theta}}$	
(2)	$\vartheta^*_B(t\text{-}1) = (\lvert\vartheta_B\rvert/\lvert\vartheta\rvert) * \vartheta^*(t\text{-}1)$ $\vartheta^*_T(t\text{-}1) = (\lvert\vartheta_T\rvert/\lvert\vartheta\rvert) * \vartheta^*(t\text{-}1)$	Community Spread (SEIRS)
(3)	$\Psi^*_+(t\text{-}1) = \vartheta^*_B(t\text{-}1)\ \alpha_+$ $\Psi^*_-(t\text{-}1) = \vartheta^*_T(t\text{-}1)\ \alpha_-$	
(4)	$\Delta\Psi_+(t) = -c_+\Psi^*_+(t\text{-}1)\ (1/D)$ **(I - C^T)** $\Delta\Psi_-(t) = -c_-\Psi^*_-(t\text{-}1)\ (1/D)$ **(I - C^T)** $\vartheta_B(t) = \vartheta^*_B(t\text{-}1) + \text{Min}(\Delta\Psi_+(t),0) + \text{Max}(\Delta\Psi_-(t),0)$ $\vartheta_T(t) = \vartheta^*_T(t\text{-}1) + \text{Max}(\Delta\Psi_+(t),0) + \text{Min}(\Delta\Psi_-(t),0)$	International Spread (Diffusion)

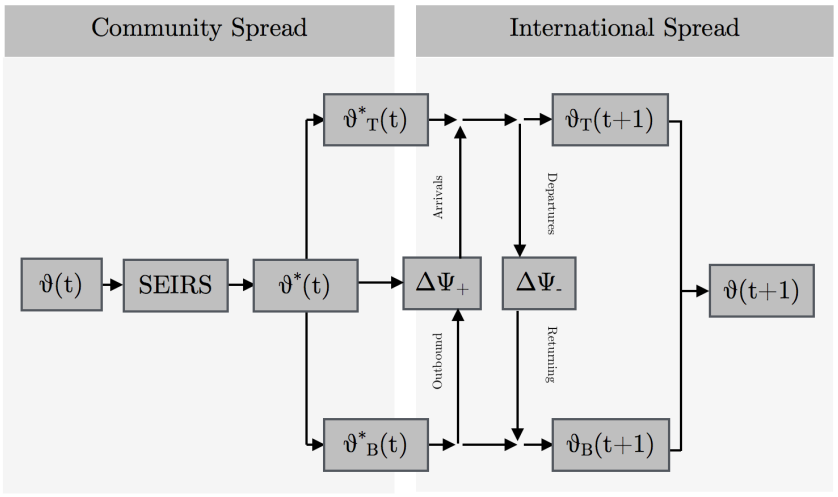


Figure 5: Diagram of Epidemic Diffusion Model Steps

3.3 Key Parameters

Now that we have discussed our approach to epidemic modelling on networks we now proceed to summarise the model's parameters in Table 3 for convenience.

Table 3: Model Parameters [*] necessary conditions for epidemic behaviour

	Parameter	Description	Dimension	Range
Data Based	$\boldsymbol{\vartheta(0)}$	Initial Populations (S,I,E,R) for each airport.	$\mathbf{R^{4xN}}$	(N/A) Gridded Population of the World (GPW), SEDAC
	$\mathbf{C}$	Relative Centrality	$\mathbf{R^{NxN}}$	(N/A) Open Flights Data
	$\boldsymbol{\alpha_+}$	% Base Population who can fly	$\mathbf{R^{NxN\ (Diagonal)}}$	(N/A) World Bank Data
Model Assumption	$\boldsymbol{\alpha_-}$	% Transient Population who can fly	$\mathbf{R^{NxN\ (Diagonal)}}$	$\alpha_- = I$, assuming no permanent migration.
	$\mathbf{c_-}$	% of people who return at the end of the week	$\mathbf{R}$	$c_- = 1$, assuming no permanent migration
Heuristic Based	$\mathbf{c_+}$	% of people who depart on any given day	$\mathbf{R}$	$\mathbf{c_+ \in (0,1)}$
	β	Rate of infection / contact (Daily)	$\mathbf{R}$	$\beta \in (0,\infty)$; $\beta > \boldsymbol{\varepsilon} > \boldsymbol{\gamma}$ [*]
	$\boldsymbol{\varepsilon}$	Rate of transfer from exposed to infectious state (Daily)	$\mathbf{R}$	$\boldsymbol{\varepsilon} \in (0,\infty)$
	$\boldsymbol{\gamma}$	Rate of transfer from infectious to recovered state (Daily)	$\mathbf{R}$	$\boldsymbol{\gamma} \in (0,\infty)$
	δ	Rate of loss of immunity (Daily)	$\mathbf{R}$	$\delta \in (0,\infty)$

3.4 Parameter Estimation

In order to create realistic simulations of epidemics, we obtain as many parameter estimates as possible from transformations of high quality data sources, which will proceed to discuss in Table 4.

Table 4: Estimation Methods

Parameter	Procedure
Relative Centrality	Let **P** be the vector of Page Ranks obtained from earlier analysis and **A** be the adjacency matrix (sourced from Open FlightsData). Then let **C** be the relative centrality matrix with elements defined by equation 4. The interpretation of this matrix is essentially a weighted graph which is identical to the original graph described by **A**, except now the edges have been assigned weights based on the relative importance of adjacent nodes. $$C_{ij} = \frac{A_{ij}P_j}{\sum_k A_{ik}P_k} \qquad (4)$$
Initial Population Estimates	We utilise the Gridded Population of the World dataset (SEDAC) to assign a value for total population at each airport (Fig 8). In order to perform this assignment there is an immediate problem as many airports are often in close proximity to each other. We assume that the maximal distance anyone will travel to reach an airport is 240km (60 kph * 4 hours = 240km). Using this we proceed by iterating over all airports within a 240km radius of each grid cell. We then assign a population contribution to each airport proportional to its Page Rank. This metric is chosen as it provides a proxy for the likelihood that the airport will be preferred by travellers in the local region. Note: This approach will exclude population grid cells which are not within 240km of any airport. These populations are excluded from the simulation as they are unlikely to be flying regularly, and are likely sufficiently isolated as to be unlikely to be infected by the disease.
% Of Base Population Who Can Fly	It is obvious that an impoverished country will not have the same number of fliers as a wealthy one, nor will everyone in a wealthy country travel. Thus it is clear that the population which can travel will be quite different from the population surrounding any given airport. In order to resolve this we acquire passenger estimates by country supplied by the World Bank and divide these through by the total airport populations for the given country. We then assume the proportion by airport is the same as at country level. Sometimes this proportion is greater than 1 particularly for major hubs, in these cases we replace the proportion by the average global proportion. Clearly this is imperfect but is likely as accurate as we will be able to obtain given this information is not readily available.

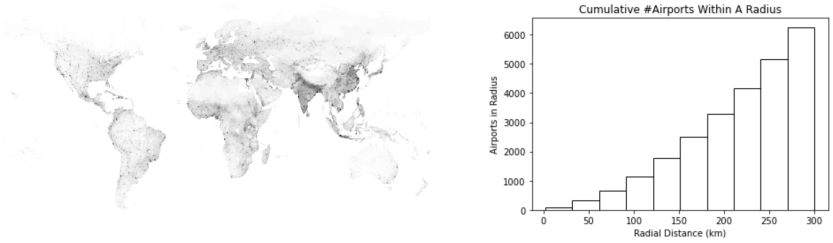


Figure 6: Population Grid ($\boldsymbol{X}^{\frac{1}{4}}$ Scale) on Left. Cumulative airports in radial distance on Right

3.5 Modelling Assumptions

Below we provide a comprehensive list of model assumptions.

1. **Fully Mixed Local Populations:** Within any given node every member of the population has equal chance of contact and thus passing on disease.
2. **Fully Mixed Wealth:** The proportion of population which may fly is distributed the same at node / airport level as at country level.
3. **Maximal Travel Distance:** We assume the maximum distance someone will travel is 240km to get to an airport, and thus anyone who exists outside of all airport radius is assume to be socially isolated and may be excluded for modelling purposes.
4. **Air Transit Only:** We assume the only way for the disease to spread between nodes is via air routes and that spread via other means eg. boat & road links are negligible.
5. **No Permanent Immigration:** Assume all individuals will return to their home country by the end of the business week. (Implicit assumption of A = AT , although small deviations from this won't have a massive impact on model behaviour)
6. **No Seasonality:** We assume that all parameters of the model are constant throughout the duration of simulation.
7. **Universal Rates:** We assume that the parameters of the SEIRS model are universal and do not vary significantly between countries.

4 Results

4.1 Visualisation of Unmitigated Spread

Now that we have outlined the theory and processes to develop our model we proceed to visualise and instance of an unmitigated epidemic (ie. no measures to decrease $\boldsymbol{\beta}$). Let $\boldsymbol{\epsilon = 0.14}$, $\boldsymbol{\delta = \frac{1}{730}}$, $\boldsymbol{\gamma = 0.048}$ and $\boldsymbol{\beta = 0.4}$, where these parameters where obtained from a study by Hou et al [2], with the additional parameter $\boldsymbol{\delta}$ which is set conservatively as we do not know the duration of which people will remain immune to 2019-nCov. We also setup the model such that the first cases occur in Wuhan for results with similarity to 2019-nCov outbreak [4]. We proceed to aggregate the time series data for S,E,I,R compartments to community levels (Utilising geographic region labels as discovered by the Block Stochastic Model) and consider the progression of the epidemic with regards to the most relevant states; Exposed **(E)** and Infected **(I)**.

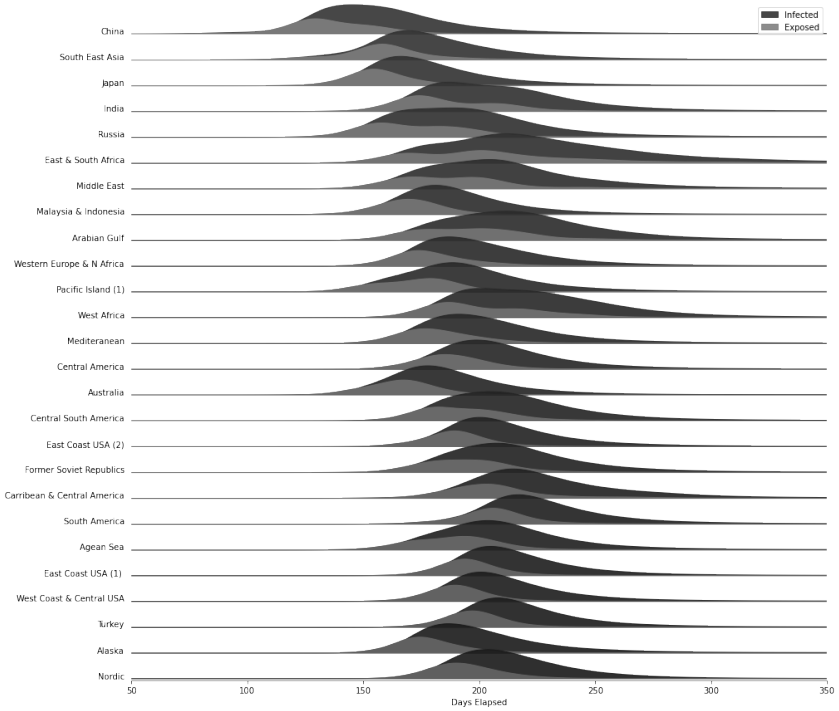


Figure 7: Exposed (Asymptomatic) and Infected (Symptomatic) Local Epidemic States for Key Communities derived from Block Stochastic Model. (Y-Axis scales not comparable, for illustration only)

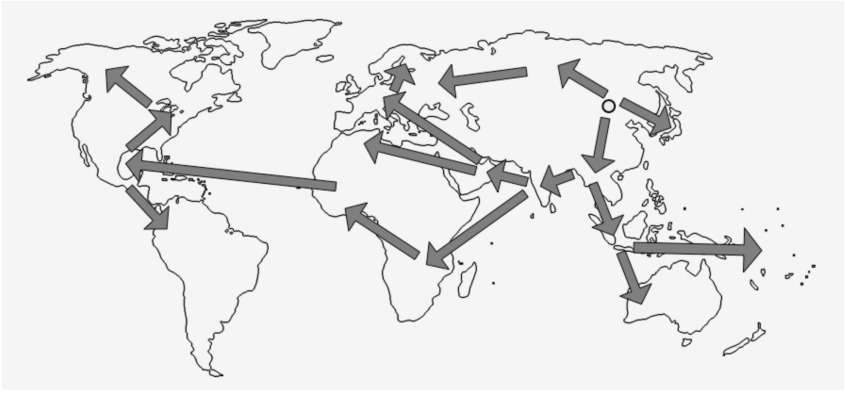


Figure 8: Illustrative Sketch of spread based off sequencing of epidemic spread provided by Figure 7)

The labelling choices are somewhat arbitrary but the main purpose of these plots in Figure 7 is to illustrate the network effect results in a somewhat staggered epidemic across different geographic communities. There is evidence of a gradual dispersion of the virus across the world starting in China, moving onward into South East Asia, Japan, Russia, India, South Africa, Middle East etc. Some of the last places to be infected being the Americas, Nordic Countries, Alaska and Turkey. Figure 8 is an illustrative sketch of one possible spread through the network which could be derived from the spread sequence in Figure 7. In reality the spread of the virus is far more complex due to the highly robust and connected nature of the network as we reported previously.

4.2 Sensitivity Analysis of Key Parameters

Where possible we have utilised data to obtain estimates for parameters and where not available we have elected to make some simplifying assumptions regarding travel behaviour. We consider data regarding rates of disease spread to be quite biased due to the nature of data collection, as has been well publicised in the recent 2019-nCov outbreak. Thus in order to obtain a better understanding of the impact of these spread parameters on our model we proceed to conduct a sensitivity analysis. Whilst we can be reasonably confident that reported rates of recovery are reliable, this is perhaps less true for infection and exposure rates. Thus we let $\boldsymbol{\beta} = \boldsymbol{k\gamma}$ (infection rate) and $\boldsymbol{\epsilon} = \boldsymbol{s\gamma}$ (exposed to infectious rate) where we assume $\boldsymbol{\gamma}$ (recovery rate) is known. It is clear that $\boldsymbol{k} > \mathbf{1}$ otherwise $\boldsymbol{\beta} < \boldsymbol{\gamma}$ and the epidemic quickly vanishes, similarly let $\boldsymbol{k} > \boldsymbol{s} > \mathbf{1}$ to ensure valid parametrisation of the model. We will not consider varying $\boldsymbol{\delta}$ (loss of immunity rate) as we will confine our analysis exclusively to the first wave of the epidemic. Let $\boldsymbol{k} \in \{\mathbf{1.1, 1.5, 2, 2.5, 3, 4, 5}\}$, and $\boldsymbol{s} \in \{\mathbf{1.05, 1.25, 1.75, 2, 2.5, 3}\}$ with $\boldsymbol{\gamma} = \frac{\mathbf{1}}{\mathbf{16}}$. We choose a smaller range for s as medical research suggests that people remain in the exposed state for around 1 week for 2019-nCov virus.

Dubai International Airport

Time to I(t) > 10K	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	-	-	443	335	276	213	179
	1.25	-	-	410	310	255	197	165
	1.75	-	-	359	270	222	171	143
	2.0	-	-	-	257	211	163	136
	2.5	-	-	-	-	195	150	125
	3.0	-	-	-	-	-	140	117

Time to Max I(t)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	+	+	458	350	291
	1.25	-	+	+	+	425	323	270
	1.75	-	-	+	450	368	280	232
	2.0	-	-	-	428	350	265	220
	2.5	-	-	-	-	322	245	201
	3.0	-	-	-	-	-	228	188

Max I(t) (Millions)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	-	0.0	0.06	5.42	13.75	14.48	14.36
	1.25	-	0.0	0.18	11.78	13.98	14.63	14.49
	1.75	-	-	1.58	13.24	14.25	14.8	14.57
	2.0	-	-	-	13.31	14.31	14.8	14.57
	2.5	-	-	-	-	14.3	14.71	14.45
	3.0	-	-	-	-	-	14.6	14.3

Time to I(t) < 1000	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	+	+	458	350	291
	1.25	-	+	+	+	425	323	270
	1.75	-	-	+	450	368	280	232
	2.0	-	-	-	428	350	265	220
	2.5	-	-	-	-	322	245	201
	3.0	-	-	-	-	-	228	188

Atlanta International Airport

Time to I(t) > 10K	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	-	-	-	385	316	243	203
	1.25	-	-	474	356	293	225	188
	1.75	-	-	414	310	255	195	163
	2.0	-	-	-	295	242	185	154
	2.5	-	-	-	-	224	170	142
	3.0	-	-	-	-	-	160	133

Time to Max I(t)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	+	+	485	371	310
	1.25	-	+	+	+	450	343	286
	1.75	-	-	+	477	391	298	248
	2.0	-	-	-	455	372	283	235
	2.5	-	-	-	-	343	260	215
	3.0	-	-	-	-	-	245	202

Max I(t) (Millions)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	-	0.0	0.01	0.87	6.58	7.54	8.01
	1.25	-	0.0	0.02	3.01	6.85	7.85	8.35
	1.75	-	-	0.21	6.37	7.31	8.39	8.92
	2.0	-	-	-	6.51	7.47	8.57	9.13
	2.5	-	-	-	-	7.69	8.84	9.42
	3.0	-	-	-	-	-	9.01	9.61

Time to I(t) < 1000	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	+	+	485	371	310
	1.25	-	+	+	+	450	343	286
	1.75	-	-	+	477	391	298	248
	2.0	-	-	-	455	372	283	235
	2.5	-	-	-	-	343	260	215
	3.0	-	-	-	-	-	245	202

London Heathrow Airport

Time to I(t) > 10K	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	-	-	455	344	283	219	183
	1.25	-	-	422	318	262	202	169
	1.75	-	-	369	277	228	175	147
	2.0	-	-	-	264	217	166	139
	2.5	-	-	-	-	200	153	128
	3.0	-	-	-	-	-	144	120

Time to Max I(t)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	+	+	455	345	287
	1.25	-	+	+	+	420	320	265
	1.75	-	-	+	445	365	276	230
	2.0	-	-	-	425	345	262	217
	2.5	-	-	-	-	320	240	200
	3.0	-	-	-	-	-	225	186

Max I(t) (Millions)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	-	0.0	0.04	3.52	7.56	7.86	7.78
	1.25	-	0.0	0.12	6.85	7.65	7.95	7.88
	1.75	-	-	1.07	7.27	7.75	8.03	7.99
	2.0	-	-	-	7.28	7.77	8.04	7.99
	2.5	-	-	-	-	7.75	8.04	8.02
	3.0	-	-	-	-	-	8.01	8.0

Time to I(t) < 1000	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	+	+	455	345	287
	1.25	-	+	+	+	420	320	265
	1.75	-	-	+	445	365	276	230
	2.0	-	-	-	425	345	262	217
	2.5	-	-	-	-	320	240	200
	3.0	-	-	-	-	-	225	186

JKF International

Time to I(t) > 10K	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	-	-	475	357	295	227	190
	1.25	-	-	440	331	273	210	176
	1.75	-	-	384	289	237	182	152
	2.0	-	-	-	275	225	173	145
	2.5	-	-	-	-	208	159	133
	3.0	-	-	-	-	-	150	124

Time to Max I(t)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	+	+	455	347	290
	1.25	-	+	+	+	420	321	267
	1.75	-	-	+	446	365	278	231
	2.0	-	-	-	425	347	265	220
	2.5	-	-	-	-	320	243	201
	3.0	-	-	-	-	-	228	188

Max I(t) (Millions)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	-	0.0	0.02	1.95	4.38	4.76	4.92
	1.25	-	0.0	0.07	3.88	4.48	4.89	5.06
	1.75	-	-	0.59	4.22	4.65	5.11	5.32
	2.0	-	-	-	4.26	4.71	5.19	5.41
	2.5	-	-	-	-	4.79	5.29	5.54
	3.0	-	-	-	-	-	5.36	5.63

Time to I(t) < 1000	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	+	+	455	347	290
	1.25	-	+	+	+	420	321	267
	1.75	-	-	+	446	365	278	231
	2.0	-	-	-	425	347	265	220
	2.5	-	-	-	-	320	243	201
	3.0	-	-	-	-	-	228	188

Hong Kong International

Time to I(t) > 10K	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	-	-	335	249	204	155	129
	1.25	-	-	308	229	187	143	119
	1.75	-	-	266	197	161	123	102
	2.0	-	-	-	187	152	116	96
	2.5	-	-	-	-	140	106	88
	3.0	-	-	-	-	-	99	82

Time to Max I(t)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	+	477	389	294	244
	1.25	-	+	+	439	358	271	225
	1.75	-	-	+	379	309	234	193
	2.0	-	-	-	361	293	221	182
	2.5	-	-	-	-	269	202	167
	3.0	-	-	-	-	-	189	155

Max I(t) (Millions)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	0.0	0.0	0.87	10.55	12.08	13.73	14.52
	1.25	-	0.01	2.44	10.98	12.54	14.27	15.1
	1.75	-	-	8.67	11.66	13.31	15.17	16.11
	2.0	-	-	-	11.89	13.56	15.48	16.47
	2.5	-	-	-	-	13.93	15.93	17.0
	3.0	-	-	-	-	-	16.25	17.38

Time to I(t) < 1000	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	+	477	389	294	244
	1.25	-	+	+	439	358	271	225
	1.75	-	-	+	379	309	234	193
	2.0	-	-	-	361	293	221	182
	2.5	-	-	-	-	269	202	167
	3.0	-	-	-	-	-	189	155

Benchmark No Network

Time to I(t) > 10K	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	-	484	267	193	155	115	95
	1.25	-	440	242	175	140	105	86
	1.75	-	-	205	148	118	88	72
	2.0	-	-	-	139	111	82	67
	2.5	-	-	-	-	100	74	61
	3.0	-	-	-	-	-	69	56

Time to Max I(t)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	435	325	266	203	168
	1.25	-	+	400	299	244	186	154
	1.75	-	-	345	257	210	159	132
	2.0	-	-	-	244	199	151	125
	2.5	-	-	-	-	182	137	113
	3.0	-	-	-	-	-	128	105

Max I(t) (Millions)	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	0.0	0.01	0.19	0.29	0.36	0.48	0.56
	1.25	-	0.02	0.21	0.31	0.4	0.52	0.61
	1.75	-	-	0.24	0.36	0.45	0.6	0.7
	2.0	-	-	-	0.37	0.47	0.63	0.73
	2.5	-	-	-	-	0.51	0.67	0.79
	3.0	-	-	-	-	-	0.71	0.83

Time to I(t) < 1000	s/k	1.1	1.5	2.0	2.5	3.0	4.0	5.0
	1.05	+	+	435	325	439	357	318
	1.25	-	+	400	479	404	328	292
	1.75	-	-	345	417	352	286	254
	2.0	-	-	-	397	336	273	243
	2.5	-	-	-	-	313	255	227
	3.0	-	-	-	-	-	242	216

Figure 9: Sensitivity Analysis conducted on 3 Major Airports $(-)$ indicates $\boldsymbol{\beta} < \boldsymbol{\gamma}$ $(+)$ indicates time exceeded 500 days.)

We see from Figure 9. that the model behaves as expected. As $\boldsymbol{s}$ increases the maximum number of infections decreases and time until peak infections increases. This makes sense in the context

of Figure 4, as when $\boldsymbol{s}$ approaches $\boldsymbol{k}$, the rate of change in the exposure compartment approaches zero. Similarly as $\boldsymbol{k}$ increases the maximum number of infections increase and the time to reach peak infections decreases, which make sense in the context of the converse of the previous argument. It is also interesting to note that time for the epidemic to return below 1000 infections is often greater than 500 days except for when $\boldsymbol{k}$ is very large. Similarly a larger value of $\boldsymbol{s}$ will also prolong the epidemic due to the slower rate of change in the exposed compartment. It is clear from Figure 9. that the values of $\boldsymbol{s}$ and $\boldsymbol{k}$ can vary the peak number of infections quite drastically often on the order of millions of cases, even with only a 0.25 change in parameter values. Thus we suggest that in the absence of reliable estimates of $\boldsymbol{\beta, \gamma, \epsilon}$ the reader should treat any numerical conclusions presented as stylised versions of reality, which convey general trends but not precise predictions. Finally we have included a Benchmark case in which the world average city population is located in a single airport - essentially a global SEIRS model. The benchmark is provided for comparison but also to highlight how this global approximation is grossly inadequate for modelling a networked system. The benchmark typically underestimates the time until peak infections and the peak number of infections.

4.3 Mitigation Strategies

The following section will examine potential mitigation strategies in the context of epidemics on international flight networks. In order to evaluate these strategies we must first select some performance metrics. We decide to select metrics which are easily interpretable by policy makers and the general public, whilst also being useful in the context of managing hospital ICU capacity and overall impact of the epidemic. Specifically we will measure the Peak Number of Infections and the Total Number of Cases of the disease, as these are transparent and can easily be measured from our simulations.

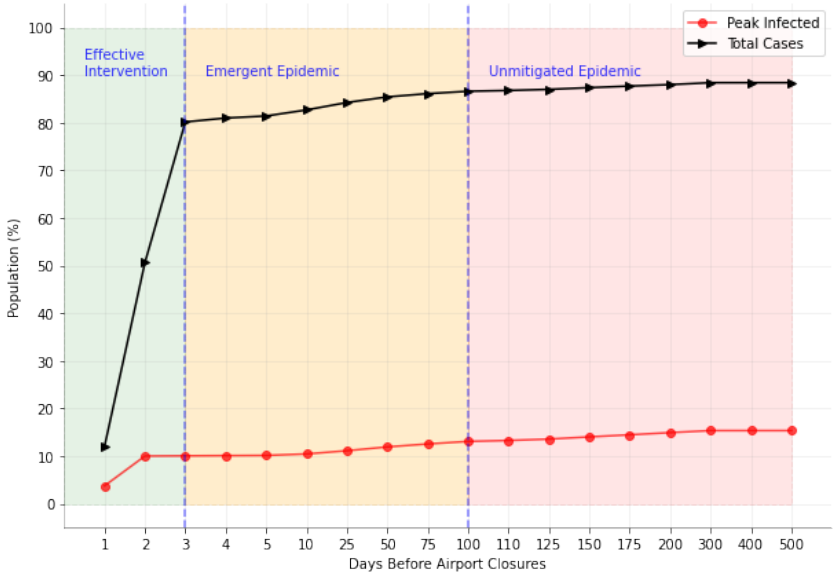


Figure 10: Impact of Worldwide Permanent Airport Closures from Nth Day since first Infection

Nth Day Rule As the first mitigation strategy we test we consider the effect of permanent closure of air routes from the Nth day after the initial outbreak (with $\boldsymbol{s = 2}$ and $\boldsymbol{k = 5}$). Our simulations in Figure 10 demonstrate that closing routes earlier reduces the peak number of infections, but does not significantly reduce the total number of cases, unless all airports are closed by the end

of Day 2. This demonstrates the high level of connectivity within the network with cases of infections proliferated across every continent within the first 3 Days of the outbreak. It is quite remarkable that the impact of these infections only becomes noticeable after around 50 days, which would explain how the recent 2019-nCov virus spread across the world unnoticed for months before a pandemic was declared. We note that closing airports immediately after Day 1 would have a significant impact, reducing the total number of cases by almost 80%. In theory this would be impossible to implement, since we assume that the infected individuals remain asymptomatic for 7 days on average! However in practice it is unlikely that the first infected individuals are also boarding a flight on Day 0, something that our model implicitly assumes due to the 'fully mixed population' assumption. Thus their may be more leeway on airport closures than our simulations predict. It is also known that the use of Natural Language Processing techniques (NLP) applied on news and social media sites were able to predict the outbreak of 2019-nCov and thus the existence of such techniques also improves the ability of governments to act early on closing their air links to affected countries.

Extending on the previous result we determine whether reducing the rate of infection whilst bringing forward airport closures have a joint effect on the peak / total number of cases. We already know that reducing the rate of infection reduces the speed at which the disease spreads (Figure 9). Thus a combination of early closures which prevent international spread and measures such as social distancing & quarantines which reduce community spread could potentially have a joint global impact on our metrics. After controlling for the reduction in infection rate, we find that an additional reduction of 4 - 5% in peak cases is unexplained (Figure 11), relative to a placebo experiment in which airport closures are implemented after 500 days for $\boldsymbol{k = 5}$ and $\boldsymbol{k = 3}$ respectively. Similarly we find an additional unexplained reduction of 4 - 5% in total cases when both early closure and social distancing / other community spread measures are employed. However the benefit of employing both measures seems to become less significant when closures happen in Day 1 or Day 2, as we approach a somewhat saturated state of reduction in peak and total cases. Whilst an interaction may still be present at later delays than Day 5 we do not consider this as the reduction in total number of cases becomes trivial after 5 days.

		% Reduction on 500 Day Closure Scenario *					**% Population**
	k	**1**	**2**	**3**	**4**	**5**	**(500)**
Peak Infections	**5**	75.69	34.86	34.55	34.26	33.99	15.33
	3	75.91	39.01	38.76	38.53	38.31	9.94
	Interaction Effect **	0.22	**4.15**	**4.21**	**4.27**	**4.32**	-
Total Cases / Recoveries	**5**	86.44	42.76	9.3	8.4	7.92	88.40
	3	86.12	42.89	14.0	13.09	12.56	79.87
	Interaction Effect **	-0.32	**0.13**	**4.70**	**4.69**	**4.64**	-

Figure 11: Comparison of $\boldsymbol{k = 5}$ and $\boldsymbol{k = 3}$ on relative reduction. * Cell value is computed as: $\mathbf{1} - \frac{\max(\text{Infections—Day N Closure})}{\max(\text{Infections—Day 500 Closure})}$ and similarly for total cases / recoveries ** Interaction Effect $= \boldsymbol{k_3}\text{Row} - \boldsymbol{k_5}\text{Row}$

Threshold Infected Rule A further modification to the previous results involves dynamically closing airports whenever the the total number of cases exceeds 1 in every 10X people within the local population. This differs from the previous method in which we implemented blanket global closures.The results of this experiment shown in Figure 12 indicate that this 'wait and see' strategy is totally ineffective for mitigation of an epidemic, despite this it has been the dominant strategy followed in some shape or form by most governments in the recent 2019-nCov epidemic! Even

considering a highly unrealistic version of this strategy in where we suppose it is possible to detect cases up to a fineness of 1 in every 10 Million people, it is already too late to close airports, providing little more than a 8% reduction in peak infections and almost no change in the total number of cases (albeit when the rate of infection is reduced there is somewhat of an improvement for higher detection sensitivities). It is clear that any of our previously proposed strategies vastly exceed the performance of this strategy. It also illustrates that strategies which involve waiting until members of the local community become infected before taking action are bound to fail.

		% Reduction on No Closure Scenario *					% Population
	k	10^7	10^6	10^5	10^3	10^2	(1)
Peak Infections	**5**	8.07	6.07	6.19	4.2	2.87	15.33
	3	7.92	5.65	6.05	4.2	2.9	9.94
	Interaction Effect	-0.15	-0.42	-0.14	0.00	0.03	-
Total Cases / Recoveries	**5**	0.4	0.25	0.26	0.12	0.03	88.40
	3	31.5	26.77	8.35	0.81	0.03	79.87
	Interaction Effect	31.10	26.52	8.09	0.69	0.00	-

Figure 12: Figure 14 Comparison of $\boldsymbol{k = 5}$ versus $\boldsymbol{k = 3}$ on relative reduction. * Cell value is computed as: $\mathbf{1} - \frac{\text{max(Infections—1 in 10N threshold)}}{\text{max(Infections—1 in 100 threshold)}}$

Limited Nth Day Rule In the previous section we realise that it is far more effective to close airports preemptively than it is to wait on some threshold level on infections to be achieved within the local population. However one could argue that it is impractical to close all airports globally, both from a economic and political point of view. Thus we proceed to examine what performance we can achieve by only closing a subset of key airports, which we rank by several metrics; Population, PageRank and betweenness. Its quite interesting to see that even with only the top 1% of airports closed we still obtained significantly greater reductions than the Threshold Rule (Figure 13a). This further emphasises the point that early intervention is far more important in the network than attempting to detect when certain infection thresholds have been breached. Its also quite interesting to see that the falloff in the strategies performance is quite non linear both down the columns and across the rows. In Figure 13b, we present the same experiment but with the ranking of airports provided by Page Rank, we notice that the reductions in peak infections are drastically superior to that of Population ranking, suggesting that closing airports by their centrality within the network structure is more important for epidemic mitigation than the size of the population. Finally we apply the same methodology to ranking airports by Betweenness. In Figure 13c, we see the results are somewhat mixed, with the strategy performing better than with previous metrics at the 5% level, and marginally worse at other levels.

% Airports Closed (Ranked by Population)		% Reduction on 500 Day Closure Scenario *					% Population
		1	2	3	4	5	(500)
Peak Infections	1%	31.47	30.44	29.3	27.5	25.75	15.33
	5%	64.77	28.48	25.49	23.85	22.75	
	10%	75.4	24.82	24.72	24.6	24.46	
	25%	75.69	33.64	33.36	33.1	32.85	
	50%	75.69	34.66	34.36	34.07	33.81	
	100%	75.69	34.86	34.55	34.26	33.99	
Total Cases / Recoveries	1%	24.74	9.97	9.37	8.4	7.79	88.40
	5%	51.89	21.13	8.28	7.52	7.06	
	10%	66.01	28.23	8.32	7.69	7.29	
	25%	86.44	40.55	8.71	8.0	7.58	
	50%	86.44	42.55	9.21	8.33	7.86	
	100%	86.44	42.76	9.33	8.4	7.92	

(a) (Airports sorted by **Population**

% Airports Closed (Ranked by PageRank)		% Reduction on 500 Day Closure Scenario *					% Population
		1	2	3	4	5	(500)
Peak Infections	1%	25.63	23.35	21.64	20.65	20.02	18.00
	5%	68.68	30.77	30.36	29.93	29.52	
	10%	74.9	35.29	34.85	34.49	34.15	
	25%	75.68	34.66	34.35	34.07	33.81	
	100%	75.69	34.86	34.55	34.26	33.99	
Total Cases / Recoveries	1%	5.7	2.28	2.2	2.13	2.06	79.10
	5%	42.16	12.21	5.24	4.64	4.28	
	10%	82.23	30.92	6.96	6.31	5.93	
	25%	86.38	41.93	8.22	7.34	6.91	
	100%	86.44	42.76	9.33	8.4	7.92	

(b) Airports sorted by **PageRank**

% Airports Closed (Ranked by Betweenness)		% Reduction on 500 Day Closure Scenario *					% Population
		1	2	3	4	5	(500)
Peak Infections	1%	24.69	22.85	20.93	19.67	18.84	18.00
	5%	68.26	34.1	34.01	33.87	33.7	
	10%	70.57	39.43	38.91	38.46	38.04	
	25%	75.44	36.53	36.1	35.72	35.38	
	100%	75.69	34.86	34.55	34.26	33.99	
Total Cases / Recoveries	1%	6.26	2.35	2.22	2.1	1.99	79.10
	5%	46.49	19.12	8.08	7.32	6.85	
	10%	77.89	29.89	8.47	7.71	7.27	
	25%	84.58	40.16	9.37	8.46	7.97	
	100%	86.44	42.76	9.33	8.4	7.92	

(c) Airports sorted by **Betweeness**

Figure 13: Relative Reductions on 500 Day Closure Scenario for percentages of airport closures.

5 Strategy Optimisation

6 Genetic Algorithm

In section 4 we explored several mitigation strategies, ultimately finding greatest flexibility and reductions in the Limited Nth Day Rule. Whilst the rule was reasonably successful in according to our key metrics, it is highly unlikely that the metrics we selected as ranking factors are in anyway close to optimal, given the complexity of the network and simulation processes. Instead we employ a genetic algorithm (GA) to search for the optimal combination of airport closures in order to maximise the utility of the Nth Day Rule. Genetic algorithms in their simplest form operate on binary strings called chromosomes which are an encoding of the parameters of interest, commonly referred to as genes. Any particular instance of a chromosome has a genotype which refers to a specific string of bits each with values 1 or 0 representing a particular gene's allele. Once the problem can be formulated within this framework the procedure followed by genetic algorithms is the following:

1. Define a fitness function F(X) which evaluates the optimality of a given genotype
2. Initialise a population of chromosomes with randomly assigned genotypes
3. Evaluate the fitness of all members of the population. Individuals will then be selected for breeding at a frequency proportional to their fitness. (Survival of the fittest)
4. In a process known as *crossover* pairs of *selected* genotypes are split uniformly at some locus along the chromosome and then *recombined*, to form new chromosomes.
5. *Mutations* are then applied at random to alleles of the the recombined chromosomes (simply by bit flipping) in order to prevent irrevocable loss of any characteristic
6. The process (3 - 5) is then repeated until convergence of the fitness of the fittest member of the population.

For a more detailed explanation of the entire process see Genetic Algorithms, Search, Optimisation and Machine Learning (David E Goldberg). We opt to use a genetic algorithm for this problem as our problem can be easily represented as a binary string and is ill suited to classical optimisation methods such as Gradient Descent, as it is not possible to analytically compute gradients and our search space is too large for approximation methods. Additionally a fitness function can be easily defined from the metrics we have described previously. In the Table 14 we identify the key information required to formulate our problem within the genetic framework.

In order to evaluate the fitness function, we must compute the values of T and P (Computing A is trivial). This clearly involves inputting the parameters encoded in the chromosomes genotype into our simulation developed in previous sections. We perform this by first using a lookup table to convert between the 195 bit country closures string specified in the genotype to a 3425 bit string airport closures vector required by our model. Next we zero out the rows and columns of the closed airports within the adjacency matrix at the appropriate time steps (to disconnect an airport from the network). Finally we proceed to run the algorithm for sufficient iterations as for the first wave of the epidemic to be completed. As our GA will run an entire simulation to evaluate the fitness of a single genotype the optimisation process could take weeks to execute. Hence we perform extensive optimisation to our simulation code in order to achieve a 20x speedup in computations, however this is still insufficient for reasonable runtime and thus we run our simulation with $\boldsymbol{\beta = 30}$ which reduces the number of required simulation iterations by a factor of 8 (by speeding up the spread of the disease). Whilst this high level of infection is unrealistic we find that the rules learned by the genetic algorithm generalise very well to typical values of $\boldsymbol{\beta}$.

Parameter	Definition
Chromosone	Instead of optimising for *which airports to close*, we choose to optimise for *which countries should close their airports*. This reduces the GA search space by 15 times which is highly desirable for convergence to global optimum. This reframing is also sensible from a policy perspective, as if some airports where to remain open, then travellers to the closed airports would simply be rebook to the remaining open airports (completing the remainder of their journey by road) defeating the point of the closure (which was to reduce international spread of the disease). We now define our chromosome as a 195 bit string where 1 in the i^{th} indicates that the i^{th} country's airports are closed and 0 indicates the i^{th} country's airports are open.
Fitness Function	In designing the fitness function we seek to balance the economic impacts of airport closures along with the reduction in Peak & Total Cases of infections, relative to an unmitigated epidemic. Since we have no prior biases towards Peak, Total Case or economic impacts we will weight their contributions equally in the fitness function: $$\textit{Let } T = \%\textit{ Reduction in Total Infections}$$ $$\textit{Let } P = \%\textit{ Reduction in Peak Infections}$$ $$\textit{Let } A = \%\textit{ Airports Remaining Open}$$ $$F(X) = T(X) * P(X) * Sin(0.5\pi * A(X))$$ The sin term in the fitness function encodes our prior knowledge that very few airport closures are associated with increasingly worse performance on our key metrics. This non linear shaping of the fitness function should help our algorithm to converge faster, avoiding exploration of poor solutions.

Figure 14: Defining key components of our problem in Genetic Framework

7 Optimisation Results

The results of the algorithm in Figure 15 are presented in the usual format for consistency, however each row now represents a different GA optimised for metrics on of Days 1 to 5 respectively, with the corresponding quantities optimised bolded in the table.

% Airports Closed (Ranked by Genetic Algo)		% Reduction on 500 Day Closure Scenario *					% Population
		1	2	3	4	5	(500)
Peak Infections	18%	**78.21**	-1.88	-1.55	-1.31	-1.12	18.00
	40%	60.71	**48.12**	45.47	43.57	42.04	
	33%	51.12	50.21	**45.57**	42.42	40.05	
	25%	42.4	42.36	42.29	**42.19**	42.06	
	32%	52.49	50.5	46.52	43.81	**41.75**	
Total Cases / Recoveries	18%	**86.55**	5.2	1.38	1.14	1.01	79.10
	40%	60.92	**34.34**	7.16	5.69	5.19	
	33%	50.2	29.28	**8.16**	6.68	6.12	
	25%	43.55	26.51	8.47	**7.27**	6.71	
	32%	51.3	30.07	7.75	6.27	**5.73**	

Figure 15: Relative Reductions on 500 Day Closure Scenario for percentages of airport closures. (Airports sorted by **Genetic Algorithm**), **bolded** number indicate is the day for which the row was optimised by GA

The results are quite remarkable, but better visualised in Figure 16 for the Day 3 Closure scenario. Not only does the GA outperform our previous ranking methods (with an equivalent % of airport closures), it also improves on the close all airports strategy by 15 - 20%! Whilst this is counter-intuitive, the GA leverages hidden structure within the network to flatten the curve over 50 days earlier, whilst also reducing peak and total infections when compared with Page Rank, All Closed and unmitigated strategies, as seen in Figure 17.

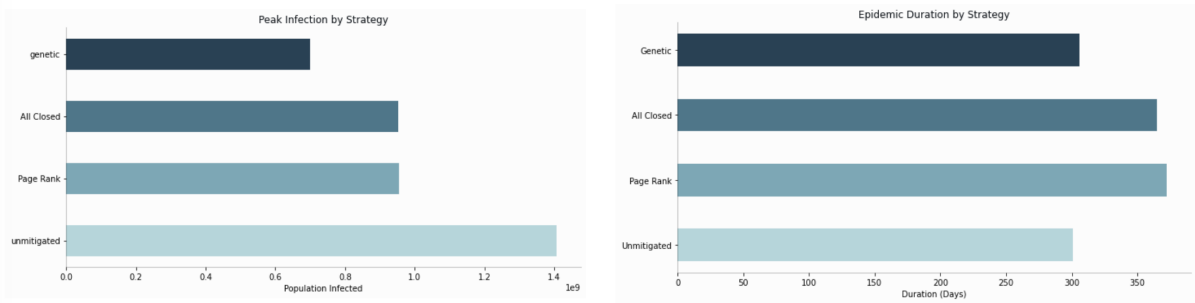


Figure 16: Day 3 Comparison of Strategy Performance

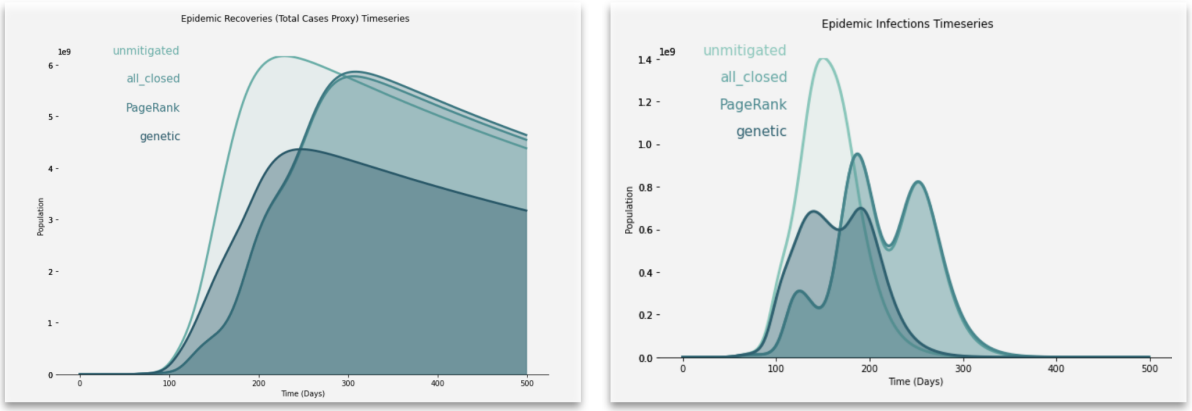


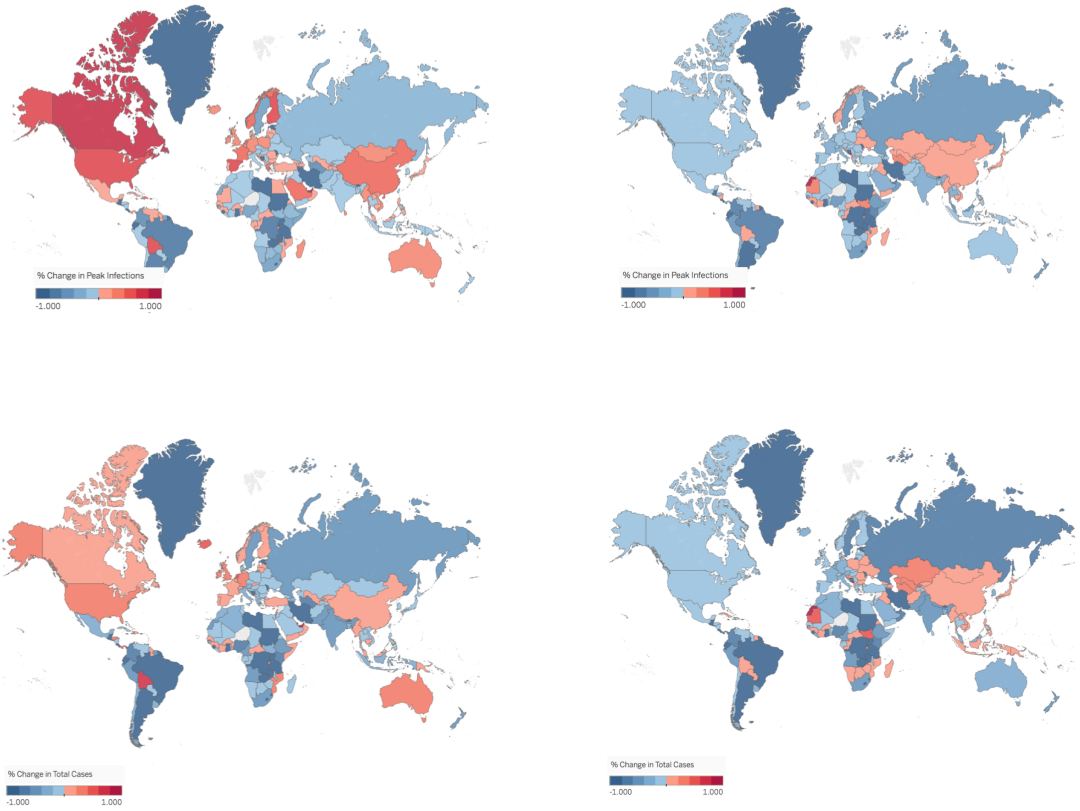
Figure 17: Day 3 Comparison of Strategy Performance

This suggests that the GA learns to leverage the network structure via closures in such a way as to accelerate the initial infection rate but achieving a lower point of equilibrium. To gain further intuition into the GA behaviour it is best to look at Figure 18 which exhibits the evolution of the GA strategy as we alter the Day at which closures occur. What we observe is that it is initially optimal to close China and certain other countries such as France which are very well connected to China via air routes. However as governments delay closures up to Day 5 we find that the GA shifts focus away from China and starts to establish 'firebreaks' in other surrounding countries such as India, Kazakstan and Russia, whilst also selective targeting certain several African and South American countries.

Figure 18: Genetic Algorithm: The Optimal Countries To Close Starting From Nth Day (Dark Blue indicates closed)

Returning our focus back to the Day 3 strategy learned by the GA in Figure 19, we examine the percentage change in peak infections and total cases under the GA strategy compared with the All

Closed and Unmitigated strategies. Despite only 33% of airports being closed under the GA strategy 73% of countries see a reduction in Peak Infections relative to an unmitigated case, whilst 60% see a reduction in peak infections relative to the all closed case. Similarly we see that 67% of countries see a reduction in total cases relative to the unmitigated case and 69% see a reduction in total cases relative to the All Closed scenario. This provides overwhelming evidence of the effectiveness of the genetic algorithm strategy over naive propositions of worldwide closures of airports and highlights how a machine learning approach to solving such a problem can provide significant added value over simple intuitive strategies such as closing by population or network centrality.



(a) Genetic Relative to All Closed

(b) Genetic Relative to Unmitigated

Figure 19: A comparison of Genetic Optimsation Strategy Versus Naive Strategies and Total Inaction

8 Conclusion

Flight networks are highly complex and connected. When applying dynamical simulation of epidemics through network we see the frightening speed at which they may spread undetected. We performed a sensitivity analysis to show that the parameters of our model are robust to perturbation before proceeding to examine some naive mitigation strategies based on various properties of the network such as population and centrality. Our findings suggest that the first week of dispersion of the disease through the network is a critical time period for effective intervention, however interventions in the network such as airport closures still provide some reductions to peak infections and total cases up to 3 months into the simulation. Furthermore we show that policies which reduce community spread can be combined with our proposed airport closure strategies to provide greater benefits than if

either policy had been used separately. Finally we explore the application of machine learning based optimisation to identify optimal airport closures within the critical first week of disease spread, in order to reduce the global impact of the epidemic whilst keeping as many airports open as possible (to preserve international commerce). Whilst the optimisation function which we select is quite arbitrary and the potential of genetic algorithms in this application have not been explored fully we find that the algorithm learns strategies which are far superior to the naive ranking based strategies explored earlier in the paper. Due to the black box nature of genetic algorithms it is not clear what strategy has been learned, however visualisation of the effects of the strategy suggest that the algorithm has leveraged the complex structure of the network to place strategic 'fire breaks' which drastically reduce peak infections and total cases. Despite 'sacrificing' some countries for the greater good our analysis that the majority of countries are far better off under this strategy, compared to other strategies proposed and also compared with an unmitigated scenario. This study highlights the potential of machine learning methods in the mitigation of global epidemics through complex networks.

References

[1] J-J Daudin, Franck Picard, and Stéphane Robin. "A mixture model for random graphs". In: *Statistics and computing* 18.2 (2008), pp. 173–183.

[2] Can Hou et al. "The effectiveness of quarantine of Wuhan city against the Corona Virus Disease 2019 (COVID-19): A well-mixed SEIR model analysis". In: *Journal of medical virology* (2020).

[3] Mark Newman. *Networks*. OUP Oxford, Mar. 2010. DOI: 9780191500701.

[4] Muhammad Adnan Shereen et al. "COVID-19 infection: Origin, transmission, and characteristics of human coronaviruses". In: *Journal of Advanced Research* (2020).

9 Appendix

9.1 Algorithm Derivation

In our review of theoretical underpinnings (Section 3.1) we discussed the graph diffusion model. Where there is an amount of fluid $\boldsymbol{\psi_j}$ at nodes $\boldsymbol{j = 1 \dots N}$. Thus fluid flows from node $\boldsymbol{i}$ to node $\boldsymbol{j}$ at a rate proportional to the difference in the amount of fluid at each node $\boldsymbol{c(\psi_i - \psi_j)}$ where $\boldsymbol{c}$ is the constant of proportionality or more commonly referred to as the *diffusion constant*. We note however that fluid can only flow between nodes if they are adjacent in graph $\boldsymbol{G}$ with adjacency matrix $\boldsymbol{A}$. Thus the total instantaneous change in fluid volume at node $\boldsymbol{j}$ is given by the following equation.

$$\frac{\boldsymbol{d\psi_j}}{\boldsymbol{dt}} = \boldsymbol{c} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\boldsymbol{\psi_i} - \boldsymbol{\psi_j}) \boldsymbol{A_{ij}}$$

Following from thus we can easily proceed to vectorise this equation for all vertices as follows, where $\boldsymbol{\psi} = (\psi_1 \dots \psi_N)^{\boldsymbol{T}}$ and $\boldsymbol{D} = \text{diag}(\deg(\boldsymbol{v_1}), \dots, \deg(\boldsymbol{v_N}))$.

$$\frac{\boldsymbol{d\psi_j}}{\boldsymbol{dt}} = \boldsymbol{c} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{\psi_i A_{ij}} - \boldsymbol{c\psi_j} \deg(\boldsymbol{v_j})$$

$$\frac{\boldsymbol{d\psi}}{\boldsymbol{dt}} = \boldsymbol{c(A - D)\psi}$$

$$\frac{\boldsymbol{d\psi}}{\boldsymbol{dt}} = \boldsymbol{-c(D - A)\psi}$$

In the basic implementation described above we consider there to be only a single fluid, however we will require 4 different fluids for the 4 states of our SEIRS model. Thus we let $\boldsymbol{\theta_j} = (\boldsymbol{S_j}\ \boldsymbol{E_j}\ \boldsymbol{I_j}\ \boldsymbol{R_j})^{\boldsymbol{T}}$ represent the number of people in the airport / node population which are in the Susceptible, Exposed, Infectious and Recovered states and $\boldsymbol{\psi_j} = \boldsymbol{\alpha_j \theta_j}$ be the proportion of people available for travel. Hence $\boldsymbol{\psi} \in \mathbb{R}^{\boldsymbol{4xN}}$ is now a matrix and thus we must modify the ordering of our equation to correct for this.

$$\frac{\boldsymbol{d\psi}}{\boldsymbol{dt}} = \boldsymbol{-c\psi(D - A^T)} \in \mathbb{R}^{\boldsymbol{4xN}}$$

Breaking this down, the components of the original definition are reconfigured such that $\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{\psi_i A_{ij}}$ becomes $\boldsymbol{\psi A^T} = (\sum_{j=1}^{N} \boldsymbol{A_{1j}\psi_j} \dots \sum_{j=1}^{N} \boldsymbol{A_{Nj}\psi_j}) \in \mathbb{R}^{\boldsymbol{4xN}}$ and $\boldsymbol{\psi_j} \deg(\boldsymbol{v_j})$ becomes $\boldsymbol{\psi D} \in \mathbb{R}^{\boldsymbol{4xN}}$. As a final simplification we assume that $\boldsymbol{A = A^T}$ which is appropriate in the case of flight routes which are almost always operated in both directions.

$$\frac{\boldsymbol{d\psi}}{\boldsymbol{dt}} = \boldsymbol{-c\psi(D - A)} \in \mathbb{R}^{\boldsymbol{4xN}}$$

This equation provides a simple model for the diffusion / travel of people through the international flight network. Thus we can now define a simple update algorithm which enables travel through the network and updating of the sizes of the local populations.

1. Repeat for $\boldsymbol{t = 1 \dots T}$
   (a) Compute $\boldsymbol{\psi = \theta \alpha}$ where $\boldsymbol{\alpha} = \text{diag}(\boldsymbol{\alpha_1, \dots, \alpha_N})$.
   (b) Update the local populations due international travel $\boldsymbol{\theta = \theta + \frac{d\psi}{dt}}$.

However there are some obvious issues with this simplistic model, namely that $\boldsymbol{\theta_i}$ typically converge to some equilibrium value $\boldsymbol{\theta^{(0)} \neq \theta^{(eq)}}$, which does not represent the reality that people travelling on holidays or business will typically return to their home country. Secondly there are some technical issues which arise from implementation, such as for vertices of large degree *simultaneous* outflows may exceed the total population at the node (this is an artefact of vectorization). Finally the flows to other vertices are not influenced by how central the airport is to the network, which is obviously important as travellers are more likely to visit major tourist / business destinations and this information is not necessarily reflected in the population (eg. Ethiopia has a very large population but it is not as important in the network as France or the UK).

The first of these 3 issues is quite easy to address, we can simply split the population $\boldsymbol{\theta}$ into a base population $\boldsymbol{\theta_B}$ who live permanently in the local area and $\boldsymbol{\theta_T}$ being the transient population in a given node on holidays / business. Refactoring our algorithm to incorporate this change is relatively simple.

1. Repeat for $\boldsymbol{t = 1 \dots T}$
   (a) Compute $\boldsymbol{\psi_+ = \theta_B \alpha_+}$ and $\boldsymbol{\psi_- = \theta_T \alpha_-}$ where $\boldsymbol{\psi_+}$ represents outbound travellers from the local base population and $\boldsymbol{\psi_-}$ represents returning transient travellers to their home countries.
   (b) We then compute $\boldsymbol{\frac{d\psi_+}{dt} = -c_+ \psi_+ (D - A)}$ and $\boldsymbol{\frac{d\psi_+}{dt} = -c_- \psi_+ (D - A)}$ respectively
   (c) Finally we now have two update equations for international travel
      i. $\boldsymbol{\theta_B = \theta_B + \min(\frac{d\psi_+}{dt}, 0) + \max(\frac{d\psi_-}{dt}, 0)}$ where we subtract the *outbound* base populations and add the *arriving* travellers.
      ii. $\boldsymbol{\theta_T = \theta_T + \max(\frac{d\psi_+}{dt}, 0) + \min(\frac{d\psi_-}{dt}, 0)}$, where we add the *returning* base populations and subtract the *departing* travellers. Note that we have chosen the wording *outbound*, *arriving*, *departing*, *returning* carefully to reflect the sequence of steps all travellers pass through in order. This is crucial to ensure logical behaviour of travellers and also to prevent leakage of fluid / people from the network.

To address the second issue we can simply divide $\boldsymbol{\psi_i}$ by the degree of the node which ensures that the outflows computed to every other node will be at most $\boldsymbol{\psi_i}$. This modification we will denote by $\boldsymbol{\frac{\psi_i}{D}}$ but it is understood that this represents $\boldsymbol{\frac{\psi_i}{D_{ii}}}$ element wise and will be implemented by Numpy broadcasting operation in Python.

To Address the final issue we replace the adjacency matrix $\boldsymbol{A}$ with a weighted matrix $\boldsymbol{C}$ to encourage flows to more central airports, where C is defined as follows, where $\boldsymbol{P_j}$ is the page rank centrality of node $\boldsymbol{v_j}$.

$$\boldsymbol{\frac{A_{ij} P_j}{\sum_{j=1}^{N} A_{ij} P_j}}$$

We then translate this back into our original formulation, with A replaced by $\boldsymbol{C}$ and $\boldsymbol{D}$ replaced by $\boldsymbol{I}$, the identity matrix (it is trivial to check by a similar derivation to previously that since C is row stochastic D may be replaced by the identity matrix). Similarly this generalises to the more complex models addressing the other issues with the original graph diffusion model.

$$\frac{d\boldsymbol{\psi}}{\boldsymbol{dt}} = -\boldsymbol{c\psi}(\boldsymbol{I} - \boldsymbol{C}) \in \mathbb{R}^{\mathbf{4xN}}$$

Next we addition another the SEIRS model which models the changes in $\boldsymbol{\theta}$ due to community spread of the disease (rather than due to international travel). As in Section 3.1 we define the following differential equations which define the SEIRS model.

$$\frac{\boldsymbol{dS}}{\boldsymbol{dt}} = \boldsymbol{\delta R} - \frac{\boldsymbol{S\beta I}}{\boldsymbol{M}}$$

$$\frac{\boldsymbol{dE}}{\boldsymbol{dt}} = \frac{\boldsymbol{S\beta I}}{\boldsymbol{M}} - \boldsymbol{\epsilon E}$$

$$\frac{\boldsymbol{dI}}{\boldsymbol{dt}} = \boldsymbol{\epsilon E} - \boldsymbol{\gamma I}$$

$$\frac{\boldsymbol{dR}}{\boldsymbol{dt}} = \boldsymbol{\gamma I} - \boldsymbol{\delta R}$$

These changes represent the spread of disease within a single community, but for efficiency we will define an operator $\boldsymbol{B}$ which acts on a modified $\tilde{\boldsymbol{\theta}}$ which enables application of the linear transformation to obtain the updated community state.

$$\boldsymbol{B} = \begin{pmatrix} -\boldsymbol{\beta} & \mathbf{0} & \mathbf{0} & \boldsymbol{\delta} \\ \boldsymbol{\beta} & -\boldsymbol{\epsilon} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\epsilon} & -\boldsymbol{\gamma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\gamma} & -\boldsymbol{\delta} \end{pmatrix}$$

$$\tilde{\boldsymbol{\theta}}_{\boldsymbol{j}} = (\tfrac{\boldsymbol{S_j I_j}}{\boldsymbol{M_j}}, \boldsymbol{E_j}, \boldsymbol{I_j}, \boldsymbol{R_j})$$

Note that the community spread occurs prior to the splitting of the population $\boldsymbol{theta}$ into $\boldsymbol{\theta_B}$ and $\boldsymbol{\theta_T}$ as it is assumed that travellers and the local community are fully mixed. Thus the final implementation of the algorithm may be described in table 5, where we include one final step of splitting $\boldsymbol{\theta}$ into $\boldsymbol{\theta_B}$ and $\boldsymbol{\theta_T}$ according to the prior populations $\|\boldsymbol{\theta_B}\|$ and $\|\boldsymbol{\theta_T}\|$ respectively.

Table 5: Simulation Psuedocode

	Algorithm: Flow & Degree Corrected Epidemic Diffusion Model	
	for t = 1 , ..., T:	
(5)	ϑ(t-1) = ϑ_B(t-1) + ϑ_T(t-1)	
(1)	ϑ^*(t-1) = ϑ(t-1) + $\boldsymbol{B\tilde{\theta}}$	
(2)	ϑ^*_B(t-1) = (|ϑ_B|/|ϑ|) * ϑ^*(t-1) ϑ^*_T(t-1) = (|ϑ_T|/|ϑ|) * ϑ^*(t-1)	Community Spread (SEIRS)
(3)	Ψ^*_+(t-1) = ϑ^*_B(t-1) α_+ Ψ^*_-(t-1) = ϑ^*_T(t-1) α_-	
(4)	$\Delta\Psi_+$(t) = -$c_+\Psi^*_+$(t-1) (1/D) **(I - $\mathbf{C^T}$)** $\Delta\Psi_-$(t) = -$c_-\Psi^*_-$(t-1) (1/D) **(I - $\mathbf{C^T}$)** ϑ_B(t) = ϑ^*_B(t-1) + Min($\Delta\Psi_+$(t),0) + Max($\Delta\Psi_-$(t),0) ϑ_T(t) = ϑ^*_T(t-1) + Max($\Delta\Psi_+$(t),0) + Min($\Delta\Psi_-$(t),0)	International Spread (Diffusion)

9.2 Algorithm Python Code

The code provided involves an EpidemicEnvironment Object which must be set with the relevant parameters. To use this code input the parameters and then call the step method as many times as necessary (and call reset to revert the state of the environment to its starting state). The history of each step is recorded in the stateHistory Numpy array for each airport state in format (***n_iters*** x 4 x N). Note that this implementation resembles the algorithm in Table 5, but has been highly optimised for performance so several intensive computations have been migrated to separate functions which are recomputed infrequently.

```
"""MIT License

Copyright (c) [2020] [Hugo Dolan]

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."""

import numpy as np
import pandas as pd
import random
from collections import namedtuple

ActionSpace = namedtuple('ActionSpace', ['n','sample'])
ObservationSpace = namedtuple('ObservationSpace', ['n'])

class EpidemicEnvironment:
    def __init__(self, adj_matrix, population_vector, agent_idx, community, infected_idx =
        0,
                alpha_plus = 0.3, alphas_plus = None, c_plus = 0.1,
                c_minus = 1, beta = 57/160 + 1/7, beta_reduced = 25/160 + 1/7, gamma =
                    1/16, delta = 1/(2*365),
                epsilon = 1/7, centrality = None,p=1, lmbda = 1.2, mu = 0.2,
                    lockdown_threshold = -1):
        """
        SEIRS Network Epidemic Model Environment
        - Currently designed only for a single agent!
        :param adj_matrix: Adjacency Matrix (A)ij indicates edge from i to j (NxN)
        :param population_vector: Nx1 Vector of populations for each node
        :param agent_idx: Airport index associated with the agent
        :param community: A list of airport indexes of airports in the same community
        :param alpha_plus: Percentage of base population which can fly
```

```
:param alpha_plus: Percentage of base population which can fly (Specifed as an Nx1
    vector)
:param c_plus: Percentage of flying population who can embark on any day
:param c_minus: Percentage of flying population who can return on any day
:param beta: Rate of infection
:param beta_reduced: Rate of infection for when agent is locked down
:param gamma: Rate of recovery
:param delta: Rate of immunity loss
:param epsilon: Rate at which people move from the exposed to infected stage
    (syptomatic)
:param p: base economic penalty for lockdowns or infections
:param lmbda: rate at which penalty for infections in unmitigated state should be
    applied > 1
:param mu: rate at which penalty for infections in lockdown state should be
    applied < 1
:param lockdown_threshold: within (0,1) defining pct of population in any node
    which can get infected before a lockdown
"""

# Initialisation
self.A = adj_matrix # Adjacency Matrix
self.A_ones = np.ones_like(self.A) # Array of ones (for efficiency its computed
    only once)
self.population_vector = population_vector
self.N = self.A.shape[0] # Number of airports
self.ID = np.eye(self.N)
self.agent_idx = agent_idx
self.c_plus = c_plus
self.c_minus = c_minus

if type(alphas_plus) == type(None):
    self.alphas_plus = np.diag(np.repeat(alpha_plus,self.N)) # Airport Vector
        proportion of populations
else:
    self.alphas_plus = np.diag(alphas_plus.reshape(-1,))

self.D = self.A.sum(axis=1) # Degrees of airports
self.D_inv = np.array([1 / deg if deg > 0 else 0 for deg in self.D])
self.infected_idx = infected_idx

# Differential Equation Matrix
self.B = np.array([[-1*beta, 0, 0, delta],[beta, -1*epsilon, 0,0],[0,epsilon,
    -1*gamma, 0],[0, 0, gamma, -1*delta]])
self.B_reduced = np.array([[-1*beta_reduced, 0, 0, delta],[beta_reduced,
    -1*epsilon, 0,0],[0,epsilon, -1*gamma, 0],[0, 0, gamma, -1*delta]])

# Centrality Matrix (N x N)
# We will refer to it as the 'Diffusion Matrix'
# since we use a captial C (like the lower case c_plus/minus for diffusion coeff)
# Used to weight the distribution of flows on the network
# So that important airports get more traffic

self.use_centrality = type(centrality) != type(None)

if self.use_centrality:
    C = self.A * centrality.T
```

```
            C_norms = np.array([1 / norm if norm > 0 else 0 for norm in
                C.sum(axis=1)]).reshape((-1,1))
            self.C = C * C_norms # Normalised Centrality

        # Initialise temporal state
        self.reset()

        # State And Action Spaces
        # Vector observation space V[0] = Local State, V[1] = Community State, V[2] =
            Global State
        # With state categores (-1,0,1) = (Decreased Infections, Static, Increased
            Infections)
        self.observation_space = ObservationSpace(27)

        # Either 0 = Open , 1 = Lockdown
        self.action_space = ActionSpace(2, lambda: random.choice([0,1]))

        # Reward parameters
        self.lmbda = lmbda
        self.mu = mu
        self.p = p

        # State Parameters
        self.community = community

        # For simple automatic rules (rather than RL)
        self.lockdown_threshold = lockdown_threshold

    def reset(self):
        # For most populations in the network they will start disease free
        s_init, e_init, i_init, r_init = 1, 0, 0, 0

        # Selecting the inital infected population
        i_exposed = 1e-5

        # Population Proportions (s, e, i , r)
        theta_prop_init = np.array([s_init, e_init, i_init, r_init],
            dtype=np.float64).reshape((-1,1))
        theta_prop_infected = np.array([1 - i_exposed, i_exposed, 0, 0], dtype=np.float64)

        # Compute the population (S,E,I,R) values
        theta_props = np.repeat(theta_prop_init, self.N, axis=1)
        theta_props[:,self.infected_idx] = theta_prop_infected

        self.thetas_B = theta_props * self.population_vector.T # Dimension 3 x N
        self.thetas_T = np.zeros(self.thetas_B.shape) # Dimension 3 x N - No initial
            people currently abroad

        self.state_history = []
        self.t = 0
        self.total_population = self.population_vector.sum()
        self.population_vector_flat = self.population_vector.reshape((-1,))


        # Infection States
        self.last_S_t_global = 0
```

```
        self.last_S_t_community = 0
        self.last_S_t_local = 0

        self.set_disabled_airports()

        # Starting State
        return self.state_to_idx(np.array([0, 0, 0]))

    def corrected(self, A,action):
        """
        If a airport is locked down it disables it. WARNING: Currently designed for a
            single agent only.
        """
        if action == 1:
            mask = np.ones_like(A)
            mask[:,self.agent_idx] = 0
            mask[self.agent_idx,:] = 0

            return A * mask
        else:
            return A

    def corrected_multiple(self, A, actions):
        agent_idxs = actions == 1

        mask = self.A_ones
        mask[:,agent_idxs] = 0
        mask[agent_idxs,:] = 0
        masked_A = A * mask

        return masked_A

    def corrected_degree(self, D, A_corrected, action):
        if action == 1:
            return A_corrected.sum(axis=1)
        else:
            return D

    @property
    def stateHistory(self):
        return np.array(self.state_history)

    def transform_state(self, current, last):
        if current > last:
            return 1 # Increase
        if current == last:
            return 0 # Static
        else:
            return -1 # Decrease

    def state_to_idx(self, state):
        # State[i] has -1,0,1 options
        # We want to simplify by starting with 1-indexing

        idx1 = 2 + state[0]
        idx2 = 2 + state[1]
```

```
        idx3 = 2 + state[2]

        flattened_idx = idx1 + 3 * (idx2 - 1) + 9 * (idx3 - 1)

        return flattened_idx - 1 # Zero indexed

    def set_disabled_airports(self, disable_airports = None):
        # Note we have now depreceated individual actions
        self.disable_airports = disable_airports

        if type(disable_airports) != type(None):
            if self.use_centrality:
                self.C_c = self.corrected_multiple(self.C, disable_airports)
                self.ID_c = self.corrected_degree(self.ID, self.C_c, 1)
            else:
                self.A_c = self.corrected_multiple(self.A, disable_airports)
                self.D_c = self.corrected_degree(self.D, self.C_c, 1)
        else:
            if self.use_centrality:
                self.C_c = self.C.copy()
                self.ID_c = np.ones((self.ID.shape[0],))
            else:
                self.A_c = self.A.copy()
                self.D_c = self.D.copy()

        row, col = np.diag_indices(self.C_c.shape[0])
        if self.use_centrality:
            self.C_c[row, col] = self.C_c[row, col] - self.ID_c
            self.outer_faster = -1 * self.C_c # ID_c - C_c
        else:
            self.A_c[row, col] = self.A_c[row, col] - self.D_c
            self.outer_faster = -1 * self.A_c # ID_c - C_c


    # RK4
    def ODE_solve(self, f_prime, y_0, step_size = 1):
        y = y_0

        # Slightly more complicated again step method
        k1 = step_size * f_prime(y)
        k2 = step_size * f_prime(y + 0.5 * k1)
        k3 = step_size * f_prime(y + 0.5 * k2)
        k4 = step_size * f_prime(y + k3)
        y = y + (1/6) * (k1 + 2*k2 + 2*k3 + k4)

        return y

    def step(self, action, disable_travel=False):
        """
        WARNING: Action is no longer implemented
        :param action: 0 = Open; 1 = Lockdown -> leads to beta_reduced being utilised for
            the agents airport & No travel in or out permitted
        :param disable_travel: Disables all travel
        :param disable_airports: Disable all travel for selected airports (N Binary Vector)
        :return: (State, Reward)
        """
```

```
# Populations
thetas = self.thetas_B + self.thetas_T
thetas_B_populations = self.thetas_B.sum(axis=0)
thetas_T_populations = self.thetas_T.sum(axis=0)
thetas_populations = thetas_B_populations + thetas_T_populations

 # Epidemic model
def SIRS_coupled_ode(thetas):
    # Creates the vectors [(S_i * I_i/ M_i, I_i, R_i)^T, ....]
    diff_vector = np.ones((4,self.N))
    diff_vector[0,:] = (thetas[0,:]) * (thetas[2,:]) * (1/thetas.sum(axis=0)) # S
    diff_vector[1,:] = (thetas[1,:])                                          # E
    diff_vector[2,:] = (thetas[2,:])                                          # I
    diff_vector[3,:] = (thetas[3,:])                                          # R

    # Computes differential equation (4x4) @ (4,N) => (4,N)
    d_thetas = self.B @ diff_vector

    if self.lockdown_threshold >= 0:
        pct_infections = thetas[2,:] / self.population_vector_flat
        actions = pct_infections > self.lockdown_threshold
        d_thetas[:, actions] = self.B_reduced @ diff_vector[:,
            actions].reshape(d_thetas.shape[0],actions.sum())
    elif type(self.disable_airports) != type(None):
        d_thetas[:, self.disable_airports] = self.B_reduced @ diff_vector[:,
            self.disable_airports].reshape(d_thetas.shape[0],self.disable_airports.sum())

    return d_thetas

# Updating community sizes
# Community Spread (Immediately before diffusion)
#thetas_star = thetas + d_thetas # Airport Community States
thetas_star = self.ODE_solve(SIRS_coupled_ode, thetas)

thetas_B_ratio = thetas_B_populations / thetas_populations
thetas_T_ratio = thetas_T_populations / thetas_populations

thetas_B_star = thetas_B_ratio * thetas_star
thetas_T_star = thetas_T_ratio * thetas_star

# Travelling populations
omegas_plus_star = thetas_B_star @ self.alphas_plus # Departures
omegas_minus_star = thetas_T_star # Arrivals (we got rid of alpha minus as it is
    redundant)

# International Spread / Diffusion (1/D prevents simultaneous changes exceeding
    the supply)
# Note we did the derivation and it turns out we can replace D with identity and A
    with C
# This now successfully weights passenger destinations according to centrality

if self.use_centrality:
    if self.lockdown_threshold >= 0:
        pct_infections = thetas_star[2,:] / self.population_vector_flat
        actions = pct_infections > self.lockdown_threshold
        row, col = np.diag_indices(self.C_c.shape[0])
```

```
            self.C_c = self.corrected_multiple(self.C, actions)
            self.ID_c = self.corrected_degree(self.ID, self.C_c, 1)
            self.C_c[row, col] = self.C_c[row, col] - self.ID_c
            self.outer_faster = -1 * self.C_c

        d_omegas_plus = -1 * self.c_plus * ((omegas_plus_star * self.D_inv) @
            self.outer_faster)

        if self.t % 5 == 0:
            # Faster implementation:
            d_omegas_minus = -1 * self.c_minus * ((omegas_minus_star * self.D_inv) @
                (self.outer_faster))
            # d_omegas_minus = -1 * self.c_minus * ((omegas_minus_star * self.D_inv) @
                (ID_c - C_c))
        else:
            d_omegas_minus = np.zeros(omegas_minus_star.shape)

        if self.lockdown_threshold >= 0:
            pct_infections = thetas_star[2,:] / self.population_vector_flat
            actions = pct_infections > self.lockdown_threshold

    else:
        if self.lockdown_threshold >= 0:
            pct_infections = thetas_star[2,:] / self.population_vector_flat
            actions = pct_infections > self.lockdown_threshold
            row, col = np.diag_indices(self.C_c.shape[0])
            self.A_c = self.corrected_multiple(self.A, actions)
            self.D_c = self.corrected_degree(self.D, self.A_c, 1)
            self.A_c[row, col] = self.A_c[row, col] - self.D_c
            self.outer_faster = -1 * self.A_c # ID_c - C_c

        d_omegas_plus = -1 * self.c_plus * ((omegas_plus_star * self.D_inv) @
            (self.outer_faster))

        if self.t % 5 == 0:
            d_omegas_minus = -1 * self.c_minus * ((omegas_minus_star * self.D_inv) @
                (self.outer_faster))
        else:
            d_omegas_minus = np.zeros(omegas_minus_star.shape)

    # Net change in Community States
    if disable_travel:
        self.thetas_B = thetas_B_star
        self.thetas_T = thetas_T_star
    else:
        self.thetas_B = thetas_B_star + np.minimum(d_omegas_plus,0) +
            np.maximum(d_omegas_minus,0) # Base population recieves returning travellers
        self.thetas_T = thetas_T_star + np.maximum(d_omegas_plus,0) +
            np.minimum(d_omegas_minus,0) # Transient population recieves travellers who
            have left their home country

    # Record state
    thetas = self.thetas_B + self.thetas_T
    thetas_populations = thetas.sum(axis=0)
    self.state_history.append(thetas)
    self.t += 1
```

```
        # Compute The State and Reward from last action for the agent
        S_t = thetas[2,self.agent_idx] # Number infected
        M_j = thetas_populations[self.agent_idx] # Current Population we will assume
            transient individuals count

        reward_unmitigated = S_t * self.lmbda * self.p / M_j
        reward_lockdown = (S_t * self.mu * self.p / M_j) + self.p
        reward = - 1 * (reward_unmitigated * (1 - action) + reward_lockdown * action)

        # Agent State
        S_t_local = S_t
        discrete_state_local = self.transform_state(S_t_local, self.last_S_t_local)

        # Community State
        S_t_community = thetas[2,self.community].sum()
        discrete_state_community = self.transform_state(S_t_community,
            self.last_S_t_community)

        # Global State
        S_t_global = thetas[2,:].sum()
        discrete_state_global = self.transform_state(S_t_global, self.last_S_t_global)

        self.last_S_t_local = S_t_local
        self.last_S_t_community = S_t_community
        self.last_S_t_global = S_t_global
        current_state = self.state_to_idx(np.array([discrete_state_local,
            discrete_state_community, discrete_state_global]))

        return current_state, reward
```