POLITECNICO DI TORINO

**Corso di Laurea Magistrale**
**in** INGEGNERIA TELEMATICA E DELLE COMUNICAZIONI

Tesi di Laurea Magistrale

# The Critical Node Detection Problem applied to COVID-19 diffusion

**Relatori**
prof. Emilio Leonardi
prof. Edoardo Fadda

**Candidato**
Rodrigo Guinovart Gutiérrez

Anno Accademico 2021-2022

## Abstract

The ongoing coronavirus disease, also known as COVID-19, has created a global health crisis with also deep social and economic impact. Individual testing has been shown as an effective non-pharmaceutical intervention strategy to mitigate the impact of the pandemic, either on the absence of an effective vaccine or co-existing with one, as vaccines standalone are insufficient to prevent widespread transmission, disease, and morbidity. In a context where COVID tests are scarce, pricey and highly demanded it is key to design strategies to optimize the allocation of them while promoting equality among people.

In this thesis we will propose different strategies to either statically or dynamically allocate COVID tests among people to contain the spread of the virus while minimizing the number of tests. We conduct a literature review to learn from experience and design a mathematical model that mimics the diffusion behaviour of the COVID-19, the so-called SEPIA model. We propose and implement different strategies to find the most relevant people to test, also know as the Critical Node Detection Problem. Finally, a Reinforcement Learning algorithm, a branch of Machine Learning, is trained by interacting with the environment to be able to decide the number of tests to perform at each time step as well as how to allocate them depending on the number of reported COVID-19 cases seeking to minimize both the number of tests and the number of deaths related to COVID-19.

Being able to model and simulate the diffusion of the COVID-19 pandemic plays a key role when designing non-pharmaceutical intervention strategies, such as testing, to contain the spread of the virus. Contact tracing resulted as the most effective mitigation strategy, but a fair allocation of tests can contain the spread of the virus quickly, hence we distribute the allocation of tests by proposing a novel buffer management techniques. Deciding how to act in response to the reported COVID cases optimizes the test allocation which is key in a test shortage context.

***Keywords—*** COVID-19, COVID test, Graph network, Critical Node Detection, Centrality, Contact tracing, SEPIA, Reinforcement Learning, Exploration, Exploitation, Diffusion model, Python, OpenAI gym

# Contents

# List of Figures

# List of Tables

# Part I

# Introduction and Objectives

# 1 Introduction

In December 2019, an outbreak of a disease caused by a novel coronavirus is registered in the Chinese municipality of Wuhan. The World Health Organisation officially recognises the novel coronavirus as SARS-CoV-2 and the disease is named COVID-19 Council (2022). Current evidence suggests that the virus spreads mainly between people who are in close contact with each other, for example at a conversational distance (Organization (2022)). Most people who fall sick with COVID-19 will experience mild to moderate symptoms and recover without special treatment. However, some will become seriously ill and require medical attention. As of 6th of July 2022 more than 5.5 million COVID-19 cases and more than 6.3 million deaths related had been reported Ritchie et al. (2020).

It has been a period characterized by uncertainty. Since the outbreak of the pandemic there has been a race to find the most effective way to contain the spread of the virus. In the absence of a vaccine, the first non-pharmaceutical intervention was social distancing. More than 90 countries declared a total lockdown (Sandford (2021)). However, the total lockdown had a deep economic impact and, more importantly, an inestimable social impact (Yeyati and Filippini (2021)). Massive testing followed by social distancing was key to mitigating the impact of the pandemic, either in the absence of an effective vaccine or coexisting with one, as vaccines alone are insufficient to prevent widespread transmission, disease, and morbidity Haines (2022).

In this thesis we will propose different strategies to either statically or dynamically allocate COVID tests among people to contain the spread of the virus while minimizing the number of COVID tests, optimizing it by applying a Reinforcement Learning method.

The pandemic spreading process can be studied as a diffusion network. The network can be denoted as G = (V, E) where each node V represents a person and each edge E represents the relationship between two persons, as it is shown in Figure 1. The diffusion model can be mathematically represented and is characterized by the possible states of each of the nodes and the equations that govern the transmission dynamics, defining transmission, recovery and fatality rates among others. In this thesis we start by implementing and testing a simple SIR model (Susceptible, Infected and Removed states) to finally extend it and propose the so-called SEPIA model (Susceptible, Exposed, Pre-symptomatic, Infected and Asymptomatic, as well as Quarentized and Death states).

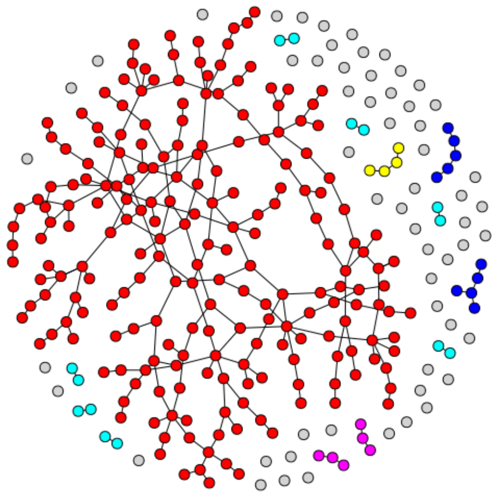


Figure 1: Erdos Renyi graph network

In Part III, we propose ten different strategies of Critical Node Detection to identify the $N$ most important nodes on which to perform a COVID test to check whether this node is infected or not. The objective of performing tests is to mitigate the diffusion of the virus while minimizing the cost function, defined as a weighted sum of the number of ill people and the unitary cost of a COVID test. We propose ten different strategies for Critical Node Detection, listed below:

- Random allocation of COVID tests
- Degree centrality
- Eccentricity centrality
- Betweenness centrality
- PageRank centrality
- Degree discount centrality
- VoteRank centrality
- Community based method
- Contact tracing based on most infected neighbors
- Contact tracing plus centrality method

Once that we have proven the effectiveness of each of the above listed strategies, we propose a novel Reinforcement Learning method to be able to decide the optimum number of tests to allocate at each simulation step. The goal of the RL agent is to minimize a goal function which basically is the number of death nodes. The agent is trained by interacting with the environment and learning which action to take depending on the observed state. The environment is defined as the above described diffusion model and the state is the number of ill tested nodes at each step.

The thesis is structured as follows. Part 2 presents an overview of related literature and works, including the main scientific areas studied in this document: Graph theory and Critical node detection and how it can be applied to the COVID-19 diffusion, and an introduction to Reinforcement Learning technique and how it can be applied to the problem presented in this thesis. Part 3 presents a simple model based on a graph network that represent the diffusion of COVID-19 and proposes ten different strategies for critical node detection and test allocation. Part 4 proposes an extended version of the diffusion model presented in Section 3, taking advantage of the experience gained since the outbreak of the pandemic. Finally, Part 5, proposes and studies different Reinforcement Learning algorithms to optimize the test allocation within the graph network, by learning from interactions with the environment proposed in Section 4.

## 2 Objectives and Methodology

The main objectives of this thesis can be stated as:

- Design a mathematical model that mimics the diffusion behaviour of COVID-19 pandemic in a reduced population
- Develop ten strategies for Critical Node Detection in a graph network to test them in order to mitigate the spread of the COVID-19 pandemic within the network
- Develop and train a Reinforcement Learning framework to mitigate the spread of the COVID-19 pandemic while minimizing the number of COVID tests performed

We followed an incremental approach both from a conceptual perspective and from an implementation perspective, starting with an easiest conceptual and practical model and extending it and adding more software resources.

- Step 1 was to conduct a review of similar literature on the main fields of this thesis: Graph theory, Critical Node detection and Reinforcement Learning techniques applied to the diffusion of a virus
- Step 2 was to design and implement a basic diffusion model that mimics the diffusion behaviour of COVID-19, considering three statuses Susceptible-Infected-Exposed

- Step 3 was to design, implement, simulate and analyze ten different strategies of Critical Node Detection to find the K-top relevant nodes to test in order to mitigate the diffusion of the pandemic

- Step 4 was to design and implement a complex diffusion model that accurately represents the diffusion behaviour of the COVID-19. Considering the following states: Susceptible, Exposed, Presymptomatic, Infected, Asymptomatic, Quarantined and Dead. Diffusion parameters were set based on experience.

- Step 5 was to design, implement, simulate and analyze a Reinforcement Learning framework to mitigate the spread of the COVID-19 pandemic while minimizing the number of COVID tests performed

# Part II

# State of the art

# 1 Graph Theory and Critical Node Detection applied to COVID-19 diffusion

Graph theory, which is a branch of mathematics, is concerned with networks of points connected by lines. Graph theory originated in recreational math problems, but it has grown into a significant area of mathematical research, with applications in several sciences, such as Computer Science, Linguistics, Physics and Chemistry, Biology, Social Sciences, Mathematics and many other topics.

We focus in the Biology field, where this thesis lies, and more precisely in "disease spreading". This term is used to imply the diffusion of contagious diseases caused by biological pathogens, like influenza, measles, chickenpox as well as sexually transmitted diseases. The spread of a disease among people has tight similarities to other examples as the spread of computer viruses (Szor (2004)) or the diffusion of knowledge, innovations, products in an online social network (Burt (1987)). Indeed, in this thesis on Section 3 in Part III, *Strategies for test allocation*, we consider an algorithm that was designed by `Google` for ranking web pages.

## 1.1 Graph Theory applied to COVID-19

On this section we conduct a review of the literature on eight papers that propose a graph model to describe the dynamic behaviour of the COVID-19 pandemic. The objective of this literature review are to highlight singularities proposed on these papers and to adapt as per convenience to our model.

The pandemic spread can be described as a diffusion network, this problem is broadly defined as the transmission of an influence from one individual to another. Models of contagion typically fall into one category that we delineate in terms of the relationship between successive exposures of a "susceptible" to an "infectious" individual: (i) independent interaction models, in which successive contacts result in contagion with independent probability p; and (ii) threshold models, in which the probability of infection changes rapidly from low to high as a critical number of simultaneous exposures is exceeded. The SI model of disease spread, the canonical model of biological contagion, is an example of an independent interaction model, it was introduced by Kermack and McKendrick (1927). There are some variants such as the former susceptible-infected (SI), susceptible-infected-recovered (SIR), susceptible-infected-susceptible (SIS), susceptible-infected-recovered-susceptible (SIRS), susceptible-exposure-infective-recovered (SEIR), and susceptible-infective-quarantine-recovered-susceptible (SIQRS), which are proposed to analyse and study the general characteristics of epidemic.

Among the main concerns in the scientific community are: predicting the evolution of the COVID-19 pandemic worldwide or in specific countries (Buonomo and Marca (2020), Giordano et al. (2020), Domenico et al. (2020), Flaxman et al.

(2020) ); predicting epidemic peaks and ICU accesses Supino et al. (2020); assessing the effects of containment measures (Buonomo and Marca (2020), Giordano et al. (2020), Domenico et al. (2020), Flaxman et al. (2020) and Gatto et al. (2020)] and, more generally, assessing the impact on populations in terms of economics, societal needs, employment, healthcare, death toll etc.

**Paper number one**

Buonomo and Marca (2020) proposed on *"Effects of information-induced behavioural changes during the COVID-19 lock-downs: the case of Italy"* a transmission model which is an information-dependent SEIR-like model. It is based on the key assumption that the choice to respect the lockdown restrictions is partially determined on a fully voluntary basis. The possible states are susceptible S, exposed E, post-latent $I_p$, asymptomatic/mildly symptomatic $I_m$, severely symptomatic (hospitalized) Is, quarantined Q and recovered R, as shown in Figure 2

It is interesting how the dynamics of the model are also ruled by the available

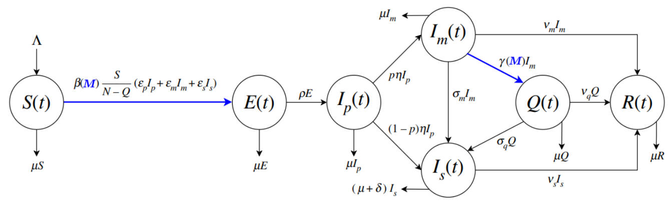


Figure 2: Transmission model proposed by Buonomo and Marca (2020)

information and rumours about the disease status in the community. Buonomo and Marca (2020) used the Reproduction numbers for measuring the potential spread of an infectious disease and assess how the available information affects the Reproduction number.

**Paper number two**

Giordano et al. (2020) proposes on *"Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy"* some non-pharmaceutical intervention strategies, such as social distancing, testing and contact tracing, to end the global SARS-CoV-2 pandemic.

They proposed a novel transmission model, the so-called SIDARTHE, which is an extension of the SIR model, consider eights stages of infection: susceptible (S), infected (I), diagnosed (D), ailing (A), recognized (R), threatened (T), healed (H) and extinct (E) as shown in Figure 3.

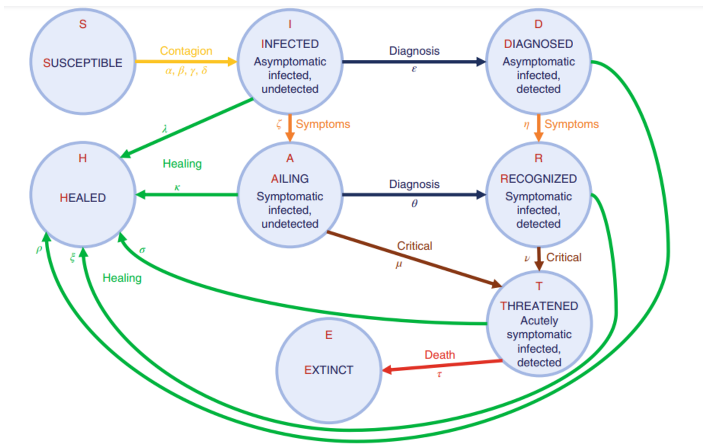


Figure 3: Transmission model proposed by Giordano et al. (2020)

In this paper some possible scenarios of implementation of measures are depicted demonstrating that social-distancing policies must be combined with massive testing (as suggested in Peto (2020)) and contact tracing. The reinfection possibility is neglected. A detailed evolution of the reproduction number, $R_t$, over time in Italy is presented. Authors suggests that the case fatality rate is unaffected by the extent of social restriction and testing.

**Paper number three**

Domenico et al. (2020) on *"Impact of lockdown on COVID-19 epidemic in Île-de-France and possible exit strategies"* proposes an age-structured method to assess, evaluate the impact of the lockdown and possible exit strategies. The Transmission model considers the following possible states: susceptible, exposed, infectious, hospitalized, in ICU, recovered, and deceased, where the infectious phase can be divided into incubation and symptomatic/asymptomatic.

Non-pharmaceutical interventions are analyzed in different scenarios. Results show that lifting lockdown with no exit strategy in place would lead to large rebound effects). Contact tracing is one essential component allowing the partial release of social distancing constraints. This paper considers that symptomatic people have a higher transmission rate, representing those asymptomatic individuals a 20% of the infected population. While the majority of the literature estimates the Reproduction number from reported cases, here it is estimated backwards from the number of hospital admissions. Authors deduce the reduction in mobility from mobile phone data to track lockdown periods. They did not consider the seasonal behaviour in coronavirus transmission due to lack of evidence.

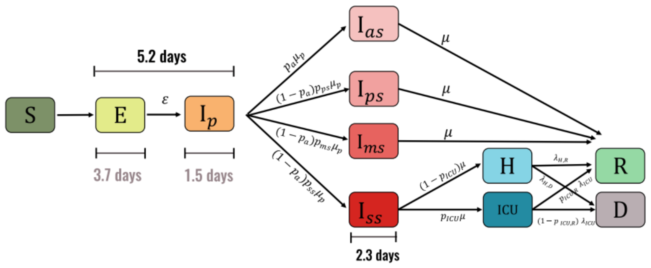


Figure 4: Transmission model proposed by Domenico et al. (2020)

**Paper number four**

Flaxman et al. (2020) conducted on *"Estimating the effects of non-pharmaceutical interventions on COVID-19 in Europe"* a study of non pharmaceutical interventions in 11 European countries in order to reduce $R_t$(reproduction number), a fundamental epidemiological quantity that represents the average number of infections generated at time t by each infected case over the course of their infection. $R_t$ is only dependant on changes in the intervention policies.
$R_0$(basic reproduction number) is defined as the expected number of secondary infectious cases generated by an average infectious case in an entirely susceptible population. In absence of control measures $R_t = R_0 * x_t$, where $x_t$ represents the fraction of the population infected. Some interesting points on this document are:

- Authors infer the the population infected and $R_t$ backwards from observed deaths as they seem to be far more reliable than case data(infected people not detected by the health system, testing policies...).
- Performed sensitivity analysis for under-reporting data.
- Individual measures, such as isolation, lead to an 82% reduction on $R_t$ compared to the pre-intervention values. Therefore, a reduction in the number of deaths predicted with respect to the deaths predicted under the model with no intervention.

**Paper number five**

*"The effects of containment measures in the Italian outbreak of COVID-19"* (Supino et al. (2020)) shows that it is possible to predict when the intensive care units will saturate, within a few days from the beginning of the exponential growth of COVID-19 intensive care patients. Also lockdown intervention is assessed.The authors predict the ICU saturation date by performing a linear regression of the logarithm of the number of ICU patients.

- The later the measures are taken the stronger these measures need to be to contain the diffusion

- The number of ICU patients represents a more robust information compared to the number of infected people (under-testing)

**Paper number six**

Tang et al. (2020) proposes on *"Estimation of the Transmission Risk of the 2019-nCoV and Its Implication for Public Health Interventions"* a deterministic variation of the classic SEIR model, including the following states: susceptible (S), exposed (E), infectious but not yet symptomatic (pre-symptomatic) (A), infectious with symptoms (I), hospitalized (H) and recovered (R) compartments, and further stratified the population to include quarantined susceptible (Sq), isolated exposed (Eq) and isolated infected (Iq). As shown in Figure 5

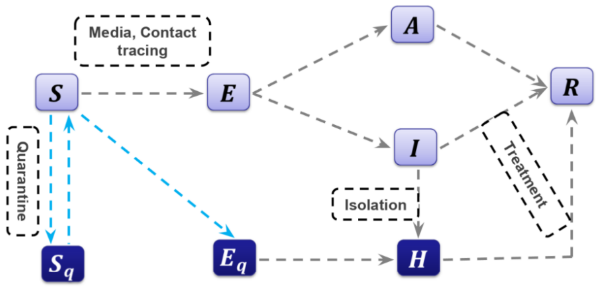


Figure 5: Transmission model proposed by Tang et al. (2020)

The authors use in the document the exponential growth law to deduce the number of reported cases per day in mainland China. Results on this paper show that non-pharmaceutical interventions, such as intensive contact tracing followed by quarantine and isolation, can effectively reduce the control reproduction number and transmission risk.

**Paper number seven**

Gatto et al. (2020), *"Spread and dynamics of the COVID-19 epidemic in Italy: Effects of emergency containment measures"* proposes an extended version of the meta-community SEIR model, considering the following cases: susceptible (S), exposed (E), presymptom (P), symptomatic infectious (I), and asymptomatic infectious (A). Presymptom transmission is demonstrated since the *serial interval* (time interval between the onset of symptoms in the primary (infector) and secondary case (infectee) tends to be shorter than the incubation period, thus it plays an important role in speeding up the spread of the disease within a community

The model proposed in the doc includes the implementation of progressive restrictions after the first case confirmed in Italy in different scenarios assessing their impact. Important consideration from this document are:

- Results suggest that the sequence of restrictions posed to mobility and human-to-human interactions have reduced transmission by 45% and $0.22 \times 10^6$ averted hospitalization cases (results obtained thanks to a well-defined spatially explicit model)
- In order to reduce the uncertainty the data retrieved from reported data on hospitalizations, fatality rates, and recovered individuals, without considering the statistics on reported infections.
- This doc uses detailed information about human mobility among the nodes (i.e. fluxes and connections), and updates on containment measures and their effects by relying also on mobile phone tracking (anonymized calls)
- The estimated high presymptomatic transmission parameter with respect to the transmission rates of symptomatic and asymptomatic infectious reproduces field epidemiological evidence -> this fact suggests the need for a massive swab testing to identify and isolate presymptomatic infectious cases.
- Underestimation of contagions: 600,000 (estimated) vs 74,386 according to official accounts

**Paper number eight**

*"Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (SARS-CoV-2)"* (Li et al. (2020)), proposes a mathematical model that simulates the spatiotemporal dynamics of infections among 375 Chinese cities. Infections are divided in two classes: documented infections (reported) and undocumented infected individuals. **Model state variables**: Susceptible, Exposed, documented Infected and undocumented Infected. **Model parameters**: the average latency period (Z), the average duration of infection (D), the transmission reduction factor for undocumented infections ($\mu$), the transmission rate for documented infections($\beta$), the fraction of documented infections($\alpha$), and the travel multiplicative factor($\theta$). Important recalls are:

- Reported infection within China + mobility data + Bayesian inference to estimate the number of undocumented data and its contagiousness.
- Authors estimates that 86% of all infections were undocumented. Transmission of undocumented infections were 55% the rate of the documented infections.

## 1.2 Critical Node Detection applied to COVID-19

Identifying a set of influential nodes is an important topic in virus containment. Regarding this, researchers have proposed different methods to classify nodes. Some studies believe that the degree, i.e. the number of connected nodes, can measure the importance of a node. Pastor-Satorras and Vespignani (2001) demonstrate that the probability of a node to be infected depends on the number of connections of this node. The LocalRank algorithm Zhong et al. (2014) extends the previous one and considers the information of the 4th steps nodes. Chen et al. (2013) combines the previous one with cluster coefficient of a node.

Contrary, Kitsak et al. (2010) stand that peripherical nodes have neglectable spreading impact regardless the degree of the node. Centrality plays an important role in node identification. Seidman (1983) underlined on 1983 the concept of network cohesion and k-core, arguing that the position of the node within the network is more important that the degree. Lü et al. (2016) propose h-index considering 2nd order neighbours as the node importance. The h-index of a node is defined as the max value such that there are at least h neighbours with value equal or greater than h. Eccentricity centrality (Hage and Harary (1995)), closeness centrality (Sabidussi (1966)) and betweenness centrality (Freeman (1977)) are shortest path-based methods. The latter one is used to identify the bridges between two communities.

Other researchers consider not only the number of neighbours but the mutual enhancement effect Wittenbaum et al. (1999). Eigenvector centrality Bonacich (1972) calculates the influence of a node proportionally to the sum of the centralities of the nodes which is connected to. Some variants of the eigenvector centrality are PageRank (Chen et al. (2007)) or LeaderRank (Li et al. (2014)). Some methods relays on the entropy centrality (Qiao et al. (2017)) to measure the influence of a node, which implies that the removal of certain nodes is most likely to cause structural variation.

Some novel methods based on node dynamics has been recently proposed. In a susceptible–infected–recovered (SIR) process, any information about the infection state could be of great value to health checking the neighbours of the infected nodes. Hu et al. (2018) show that any node's influence can be quantified purely from its local network environment, based on the nature of the spreading dynamics.

All the above-mentioned approaches introduced techniques to find individual vital nodes. If we use these methods to find a set of vital nodes, we can rank them according to one centrality and pick the top-N nodes. However, some nodes from this set are overlapped. To find a set of vital nodes subject to some functional objectives is usually called influence maximization problem (IMP). IMP problems can be structural or functional. Kempe et al. (2005) proposed a general framework for IMP under operational dynamical processes and studied two simple yet widely applied spreading models: the linear threshold model and the independent cascade model (Kempe et al. (2003)). The SIR model is very close to the independent cascade model with some assumptions.

As the IMP problem are NP-hard, an approximate solution is searched instead of the exact solution. Heuristic algorithms are the most common among all approximate algorithms, e.g. rank all nodes according to specific centrality measure and directly pick up the N top-ranking nodes. A slight improvement is the adaptive recalculation, that is, to choose the node of the largest centrality at first, and then recalculate the centralities of nodes after every step of node removal (Chen et al. (2009)). Another kind of algorithms to solve IMP problems is the greedy algorithms, they add nodes to the target set, ensuring that each addition brings the largest increase of influence to the previous set.

# 2 Reinforcement learning applied to COVID-19 diffusion

On this section we conduct a review of available studies on the field of COVID-19 diffusion that use Machine Learning techniques but more in detail Reinforcement Learning (RL) techniques.

Machine Learning has been broadly used for COVID-19 pandemic for forecasting and predicting the evolution of the pandemic, to facilitate COVID-19 diagnosis for healthcare professionals or even for drug discovery and vaccine development. It is very interesting how ML can assist in detecting COVID cases , for example through X-Ray techniques (Chen et al. (2021)), screening patients using a Chat Bot (Bharti et al. (2020)) and assisting healthcare professionals. We now focus on ML techniques on predicting and tracking the COVID diffusion.

Many computational models are developed for modeling COVID-19's transmission dynamics. Kumar et al. (2021) investigates the modified LSTM approach to forecasting the likely COVID-19 cases and deaths. It also describes deep reinforcement learning for optimizing the prediction results based on symptoms using real data. Ohi et al. (2020) investigate optimal strategies to reduce the spread of disease using reinforcement learning. A SEIR-like model is used in this paper. Author proposes various lockdown strategies such as age-based lockdown or n-work-m-lockdown ((n days without lockdown followed by m days of lockdown). Beigi et al. (2021) on a Susceptible-Exposed-Infectious-Recovered type model implement reinforcement learning optimal control and proposes four different vaccination strategies.

Part III

# Strategies for test allocation

# 1 Introduction, objectives and methodology

## 1.1 Introduction and Objectives

As seen in Section 1.1 in Part II, Graph Theory has been broadly used to model and analyse the spread of a disease. We define $G = (V, E)$ as a a network where each node $V$ represents a person and each edge $E$ represents the relation between two people. The model is characterized by the stages of infection and the dynamic equations, in this Part III we first define a simple SIR model to represent COVID-19 considering the following stages of infections: Susceptible (S) if a node has not been infected, Infected (I) and Removed (R) when a node that is infected tests positive and is isolated so cannot infect other nodes, which is a slight modification of the traditional SIR model, widely used in the literature, where the R stands for Recovered. However, we will consider a sufficiently small number of simulation steps (analogue to days) to assume that a node, once infected, will not have time to recover and become infected again.

Throughout this document, we define a simulation as a bunch of subsequent steps or iterations that analogues to days. A simulation imitates the behaviour of the COVID-19 pandemic within a network. We can set the inputs of the simulation, such as the network topology, the initial percentage of the network infected, the number of tests the strategy for selecting which nodes to test. As an output we get the evolution of the dynamics of the simulation, that is, the number of nodes for each status at each time step. The dynamics can be visualized as Diffusion trends such as in Figure 6. The simulation ends after a predefined number of steps or iterations, typically 100 steps, or when there is no more nodes infected.

The dynamic equations of the SIR model are mainly characterized by the trans-

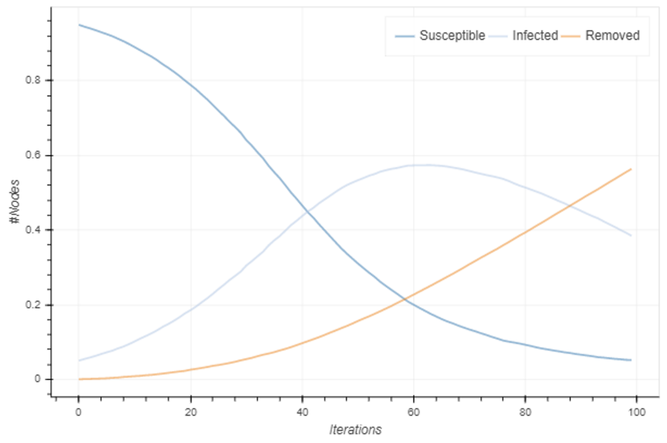


Figure 6: Diffusion trends of a simulation

mission rate ($\beta$), that is, the probability that an Infected node infects a Susceptible neighbour node, defining *neighbours* as two nodes connected by an edge which is a concept that will be widely used in this Part, and the removal rate ($\lambda$), that we will later introduce in this document. This Part aim is to propose strategies to

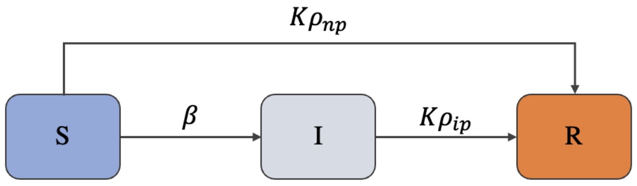


Figure 7: SIR transmission model

effectively contain the spread of the COVID-19 disease within the network. Performing individual testing to detect infected nodes and isolating them to the rest of the network, has been proved as an effective measure in virus containment in the absence of an effective vaccine (Section 1 in Part II). The strategies proposed are then based on finding which are the most important nodes within a network to test. If a node is tested positive, then is quarantined and will lose his ability of contagion, what turns to less infected nodes in the long run.

It is obvious to think that the more tests performed at each step, the faster the spread of the pandemic will be contained. However, since the outbreak of the pandemic the World has experienced periods of resource shortages, especially in the early stages or in the spike of COVID-19 cases, the so-called "waves" (Haines (2022)), so we consider tests as a limited resource. The problem instanced in this Part then becomes in finding the minimum number of tests that leads to the containment of the virus spread in the long run. To fine tune the trade-off between the number of tests performed and the number of nodes infected we introduce the concept of *the cost function*. The cost function, Eq. 1, is defined as the sum of the number of infected nodes, weighted by the penalty of a node becoming infected, the number of test performed, weighted by the cost of an individual test and a novel factor proposed in this thesis that penalises when there are a high percentage of the population infected at the same time.

$$\min\left(\sum_{t=0}^{\inf}\left(ck+r\left(\sum_{i=1}^{N}V_i(t)\right)+w\left(\sum_{i=1}^{N}\frac{X_i(t))}{N}\right)^3 N\right)\right), \tag{1}$$

where $N$ is the number of nodes in the network, $c$, $r$ and $w$ are the cost of a test and the penalties respectively. $V_i(t)$ is the distribution of nodes infected at time, that is $V_i(t=t_n)=1$ if node $i$ became infected at time $t_n$ and $X_i(t)$ the infection state vector at time t. And $k$ is the number of COVID tests done at each time step.

We propose in this report ten different strategies to decide, at each simulation step, which are the most important nodes to test in order to minimize the cost function, what converts the problem into an optimization problem. The strategies proposed include random test allocation, seven distinct *centralities* and contact tracing of already detected infected nodes. A centrality quantifies the importance of a node in a network and can be measured by various metrics, such as degree centrality, eccentricity centrality, and betweenness centrality.

## 1.2 Methodology

The following steps were followed to address the objective of this Part, which is to effectively define and compare strategies for test allocation:

- Step 1. Defining the nodes of the graph and the probability of edge creation, which is the probability of two nodes being connected. Defining the possible states of each node and the dynamic equations, defining the variables of the network: transmission rates, initial fraction infected.
- Step 2. Implementing Step 1 in Python. Converting the graph network into a Diffusion model by means of a simulation. Each simulation consist of a fixed number of subsequent simulation steps. At the end of the simulation we obtain the trends for each of the states.
- Step 3. Defining the cost function, which is the aggregated value of the cost function calculated at each time step. Defining the rate between the cost of a unitary test and the penalty of a node becoming infected and adjusting them after several experiments.
- Step4. Defining ten different strategies and implementing them to the Diffusion model stated above.
- Step 5. Testing each of the ten strategies to find which is the one that retrieves a minimum result of the cost function.

# 2 Discussion of the instance generation

## 2.1 Problem statement

We consider our problem as a functional Influence Maximization Problem (functional IMP). As we have previous mentioned in Section 1 in Part II, typical IMP problems are NP-hard. Thus, we will try to find an approximate solution by a heuristic algorithm. This algorithm consists on select at each time step a set of top-k most spread influencer nodes under the SIR model. Let's now introduce some definitions of the mathematical problem.

Let $A$ be the $N \times N$ adjacency matrix of an undirected network of N nodes ($A_{ij} = 1$ if there is an edge between nodes i and j, and 0 otherwise), and $X(t)$ the infection state vector at time t ($X_i(t) = 1$ if node i is infected at time t, and 0 otherwise). The dynamics of the system are defined as:

$$X_i(t) = \begin{cases} 0 \to 1 & \text{at rate } \beta \sum_j A_{ij} X_j(t) \\ 1 \to 0 & \text{at rate } \delta \end{cases} \tag{2}$$

where $\beta$ and $\delta$ are the infection rate and recovery rate, respectively. In this specific problem we assume that the recovery rate is large compared to the overall time of the simulation, thus $\delta$ is equal to zero. Whenever a node goes from state 0 to 1, a penalty $r$ is paid. Let $M(t)$, be the vector of length N representing the distribution of tests performed in the network ($M_i(t) = 1$ if node i is being checked at time t, and 0 otherwise). We apply the following constrain:

$$\sum_i M_i(t) \leq b(t) = b, \tag{3}$$

where $b(t)$ is the maximum number of tests that can be done at each time step. We assume it to be fixed at each time interval b. At each time t a test is performed in every node in $M(t)$, at a cost time $c$ and characterized by the following probabilities:

$$\begin{aligned} \rho_{ip} &= P(\text{ill} \mid \text{positive}) \\ \rho_{np} &= P(\text{not ill} \mid \text{positive}) \\ \rho_{in} &= P(\text{ill} \mid \text{negative}) \\ \rho_{nn} &= P(\text{not ill} \mid \text{negative}) \end{aligned} \tag{4}$$

Once that a node is tested and found positive, it is removed from the network otherwise no measure is taken. Thus, the dynamic removal rate of the network will be:

$$X_i(t) = 0, 1 \to \text{removed,at rate } \rho_{ip} \sum_i X_i(t) M_i(t) - \rho_{np} \sum_i (X_i(t)) M_i(t) \tag{5}$$

We also define $k(t) = \sum_i M_i(t)$ , the total number of nodes checked at each time step *t*. As the objective of this section is to compare the effectiveness of different strategies for Critical Node Detection, we consider the number of COVID tests fixed at each step within a simulation, *K*.

The set of *K* nodes to test at each step, that is, the distribution of the $M(t)$ vector, are the top K-ranked nodes according to the strategies proposed. If we think of the network as nodes connected to each other, the most intuitive strategy will be to test the nodes that have more neighbors, that is, that are connected to more nodes. These strategies are called *"Strategies for Critical Node Detection"*, and will be detailed in section 3.

The value of *K*, i.e. the number of nodes checked at t, will be optimized to minimize the total cost function, defined as:

$$\min\left(\sum_{t=0}^{\inf}\left(\sum_{i=1}^{N}(cM_i(t)+rV_i(t))+w\left(\sum_{i=1}^{N}\frac{X_i(t))}{N}\right)^3 N\right)\right) \tag{6}$$

$$\sum_{i}^{N} M_i(t) = k, \tag{7}$$

$$\min\left(\sum_{t=0}^{\inf}\left(ck+r\left(\sum_{i=1}^{N}V_i(t)\right)+w\left(\sum_{i=1}^{N}\frac{X_i(t))}{N}\right)^3 N\right)\right) \tag{8}$$

Where $V_i(t)$ represents the distribution of the node that had been infected at time t and N the total number of nodes in the network. The parameters *r*, *w* and *c* are the penalty of a node becoming infected, the penalty of simultaneous nodes becoming infected and the cost of a test, respectively.

We also add a third term to the cost function: $\left(\sum_{i=1}^{N}\frac{X_i(t)}{N}\right)$, that represent the instantaneous number of infected nodes divided by the total number of nodes, that is, the fraction of people that is infected at time t. Thus, the cost function increases if there is a high percentage of the network infected. This term is very important because deals with a very sensitive topic in the COVID-19 pandemic which is that if the number of infected people who need health care at the same time is greater than the hospital capacity, the hospitals collapse and people are left unattended. Therefore, in this document we consider that for the same number of infected people is better that these people are more spread in time than concentrated in a shorter time interval. This term has a polynomial growth with almost negligible influence when the fraction of infected people is up to 20 $\%$ (hospitals can only deal with low percentage of people infected). This function is multiplied by a penalty for nodes being infected at the same time, $w$.

We fix the infection penalty $r = 1$ and we choose *c* and $w$ accordingly.

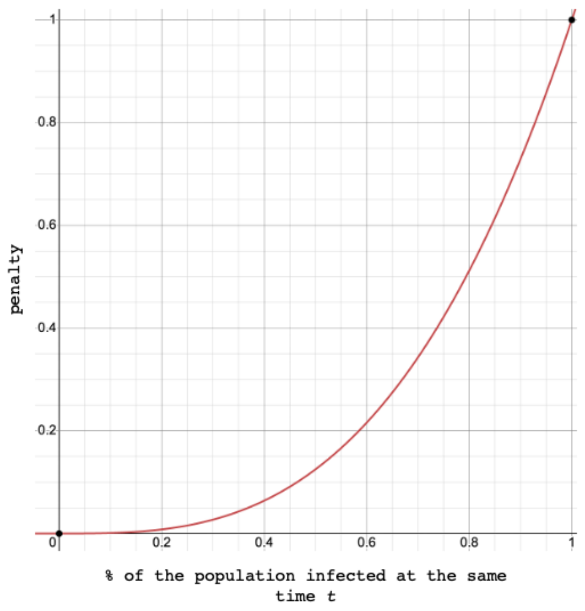


Figure 8: $f(x) = x^3$, function that models the penalty paid for fraction of people infected at same time

In order to find the number of tests *K that minimizes the total cost function presented in Eq.* 8, several simulations will be run in parallel over a given network topology while varying the number of tests, $k_j$, done at each step. Let us define then:

$$f(x_j) = f(x|k = k_j) \tag{9}$$

as the value of the cost function when doing $k_j$ at each steps in a simulation. Then, for each simulation *j* we will obtain a tuple $(f(x_j), k_j)$. That can be represented as Figure 9. Then the goal is to find the value of $k_j)$ that minimizes $(f(x_j)$.

Also note that the number of simulation steps is not always the same as the simulation can be finished when there is no more infected nodes, then the cost function is the sum of the values of the cost function at each simulation step divided by the number of time step.

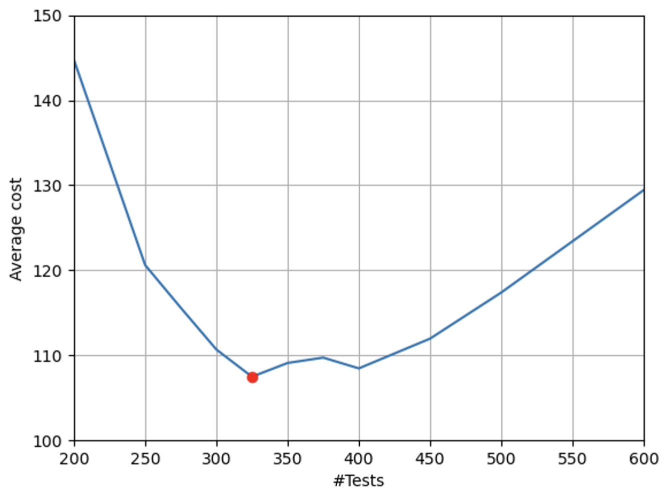


Figure 9: Cost function value vs number of tests

## 2.2 Environment set up

For the simulation of the experiment, we use the Erdos Renyi model for randomly generate the graph $G = (V, E)$ from the `NetworkX` library of `Python`, with the number of nodes in the network equal to 10000, and the probability of edge creation equal to 0.0005 which results an average degree per node equal to 5. Refer to Part IV to see the rationale on why we designed the network with this parameters. As we have said to model the diffusion spread of the disease, we use a Susceptible-Infected-Removed model which is a slight variation of the standard SIR model where the R stands for Recovered. To develop the model we use as a reference the `NDLIB` Python software package that allows to describe, simulate, and study diffusion processes on complex networks(Rossetti (2022)). To see the code explained, refer to Appendix A

We represent each of the three possible status with 0-1-2 respectively. The dynamic of the diffusion process works as follows: At each iteration (each simulation day) we check all not-removed nodes (status 1 or 0). If the node is Susceptible (status 0) a pseudo random number between 0 and 1 is calculated and multiplied by the number of infected neighbours, this means that the higher the number of infected neighbours the higher the probability to become infected. If this number is smaller or equal to the infection rate $\beta$ this node becomes infected. The same idea to detect infected nodes in the network: if a node (either infected or susceptible) belongs to the target set of nodes to be checked at a time t, its health status is checked and a pseudo random number between 0 and 1 is calculated, if it is infected and the number is lower or equal to the true positive rate $\rho_{ip}$ the node is removed from the network. On the other hand, if it is susceptible and the number is lower or equal to the false positive rate $\rho_{np}$ the node is removed from the network although it is not infected. The pseudo-code of the behaviour of the diffusion process is described in Algorithm 1.

**Algorithm 1** Diffusion model behaviour

```
$\beta \leftarrow \beta$                                    ▷ Infection rate
$P_T \leftarrow p_{ip}$                                     ▷ True positive rate
$P_F \leftarrow p_{np}$                                     ▷ False positive rate
$K \leftarrow [K]$          ▷ Set of nodes to test selected according some strategy
while $t \leq 100$ do
    for every node u in the Network if u is not "Removed" do
        $\epsilon$ is a random number
        if status(u) is "Susceptible" then
            $V$ are the infected neighbors of u
            if $\epsilon < P_F$ then
                status(u) $\leftarrow$ "Removed"
            else
                exposure_rate = 1 $-$ (1 $-$ $\beta\, len(V)$)
                if $\epsilon <$ exposure_rate then
                    status(u) $\leftarrow$ "Infected"
                end if
            end if
        else if status(u) is "Infected" then
            if u in K and $\epsilon < P_T$ then
                status(u) $\leftarrow$ "Removed"
            end if
        end if
    end for
    $t \leftarrow t + 1$
end while
```

We fix the probability of a node become infected, $\beta$, equal to 0.03 and as we said before the rate of becoming infected depends on the number of directed infected nodes. At the initial step we consider that a $5\%$ of the network is infected. The false positive rate is equal to 0.02 and the true positive rate equal to 0.95. In the following table we can see a summary of all the model parameters. The parameters of the cost function $r, c, r_i$ had been defined according to several simulations.

| **Parameter** | **Description** | **Value** |
|---|---|---|
| $N$ | Number of nodes | 10000 |
| $D$ | Probability of edge creation | 0.0005 |
| $\langle d \rangle$ | Average node degree | 5 |
| $T$ | Number of iterations (days) | 100 |
| $\beta$ | Infection rate | 0.03 |
| $\alpha$ | Initial infected fraction | 0.05 |
| $p_{ip}$ | True positive rate | 0.95 |
| $p_{np}$ | False positive rate | 0.02 |
| $r$ | Penalty | 1 |
| $c$ | Unitary test cost | 0.2 |
| $r_i$ | Penalty paid for simultaneous infected nodes | 0.3 |

Table 1: Summary of SIR model parameters.

## 3 Implementation and results

We will consider ten different strategies to select those most influencer nodes based on the literature reviewed in Section 1 in Part II and we will propose some novel strategies.

The most straightforward method is to rank all the nodes according to some centrality and directly pick at each time step the top $k$-valued nodes as the most influential ones and thus check those nodes. As it has no sense to continuously check the health status of the same $k$ nodes, once a node has been tested negative, we place it at the end of the buffer and it won't be tested again since an epoch has been completed. At the next iteration we will check the top $k$-valued nodes excluding those belonging to the buffer. An epoch has been completed when every either susceptible or infected node in the network had been tested, then the buffer is emptied. The existence of a node buffer is a concept that has not been studied in this type of problems (or at least no literature on the subject has been found) but provides great advantages in pandemic containment and whose management will be a topic of interest in future works.

Once the strategy had been decided, designed and developed we run the same experiment in the same Erdos Renyi network eleven times with different number of tests performed each day (different number of nodes checked at each time step). Those number of tests range between 200 and 600 tests per day. For each of those eleven simulations we calculate the cost function explained in 2 in this Part and find the number of tests that minimizes the cost function. One simulation consists in the random generation of the graph at the initial time step, the diffusion of the pandemic among the nodes for 100 iterations (days) and the target set testing at

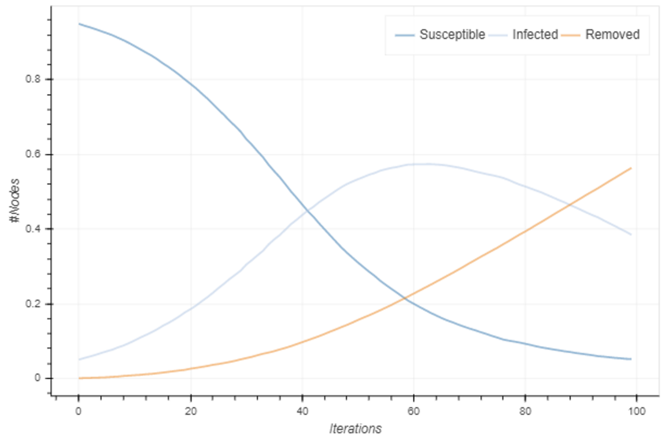


Figure 10: Diffusion trend of the coronavirus pandemic

each iteration. The average cost of each simulation is calculated as follows; the cost function calculated in each iteration is added up and divided by the total number of iterations.

In the following sections we will study some strategies including some centrality-based methods and some tracing-based methods. For each strategy we will follow the same procedure; a short introduction of the method used will be depicted along with an image of the cost function and the diffusion trend of the simulation with the optimum number of tests per day as the one depicted in Figure 10. The graphics are visualized with the `Bokeh` library for `Python`.

Figure 10 depicts an example of a diffusion trend of a simulation with a 1% of the population tested each day, that is, 100 nodes at each iteration. The diffusion trend represents the evolution of the pandemic over time, in this document we will only consider the first 100 days of the simulation. In the figure we can see three trends: The first one, the dark blue one, depicts the evolution of the Susceptible nodes that we can see that at the initial step is all the network except the initial infected fraction, it decreases as nodes become infected or removed. The light blue one depicts the evolution of the percentage of the node infected at each time that in this example the percentage of infected people reaches almost the 60%. And finally, the red one represents the evolution of the removed nodes.

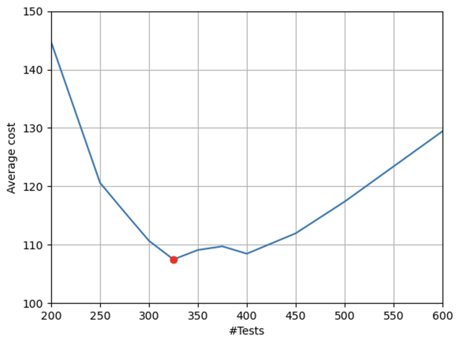
Furthermore, for the most relevant strategies we will change some of the parameters and probabilities such as the average node degree, the infection probability or the true positive rate and we will discuss the performance of those strategies. For other strategies we will not carry out the latter experiments as there are heavy time consumption strategies such as the eccentricity that has a complexity equal to $O(n^2)$ where n is equal to 10000 in this document. Thus, a single simulation (of the eleven carried out for each strategy) can last up to 3 hours.

In Section 4: *Discussions of the results* we will compare the different strategies and draw some conclusions.

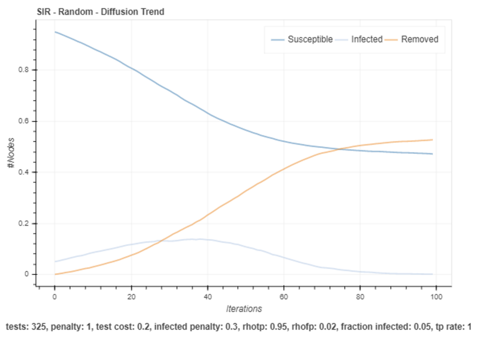
## 3.1 Random selection

The first strategy followed is the simplest one and is the basis to demonstrate that the strategies based on some centrality make sense. Each time step of the simulation represents a day, we select at each time step a set of random nodes without applying any centrality, excluding those nodes that had been removed from the network (status 2) and those belonging to the buffer, once a set of nodes had been tested negative we placed them at the end of the buffer as we explained in the previous section. Figure 11a represent the average cost function for the different simulations, we can observe that the number of tests that minimizes the cost function is equal to 325 tests per day which result a cost of 107.5. We can see that for lower number of tests the function cost increases as the number of infected nodes increases and for higher number of tests the cost function also increases as the number of tests increases and thus the overall cost of the tests. Thus, the optimum number must be a trade-off between these two parameters.

Figure 11b depicts the diffusion trend of the simulation with the optimum number of tests per day inferred from the previous image, 325 tests. In the diffusion trend we can see that the fraction of people infected does not exceed the 20 $\%$ of the total network. It starts decreasing after forty days and after eighty days the disease is mitigated. If we compare this figure with the one in the previous section, Figure 10, we can perfectly observe the enhancement in pandemic contention by performing more tests per day.



(a) Cost function vs number of tests for random nodes selection



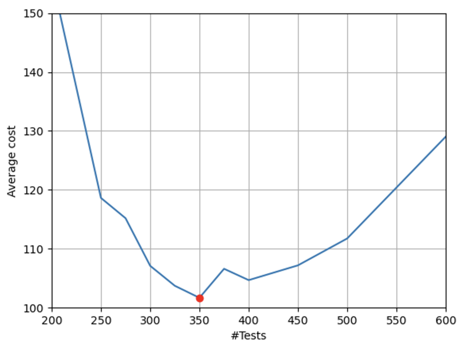
(b) Diffusion trend of random node selection

Figure 11: Random allocation of tests - The optimum number of tests to minimize the cost function is 325

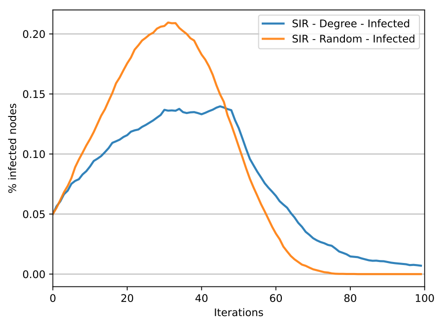
## 3.2 Degree centrality

Now we assume that the most influential nodes are those who has a higher degree, that is, a higher number of neighbours. We rank all the nodes according to their degree, at each iteration we pick the top-k valued nodes, where k is the number of tests. As the previous strategy, we run the same experiment for eleven simulations for different values of k. We have calculated the average cost function for each of them and they are depicted in Figure 12a. We can see that the minimum cost is achieved with 350 tests per day at a cost equal to 101.7.

In Figure 12a we can see the diffusion trend of this strategy. If we compare it to the diffusion trend of the random node selection strategy, we can see by studying the curvature of both trends that in this case the maximum is sharper, and the infected people start decreasing drastically. We also can observe that the first maximum occurs at iteration number 28 which precisely coincides after the end of the first epoch $\lfloor \frac{\#nodes}{\#tests} \rfloor = \lfloor \frac{10000}{350} \rfloor = 28$, this means that when the top k-valued nodes are checked again the number of infected people start decreasing drastically and when we test the lowest degree-valued nodes the trend flattens. Hence, we can prove the efficiency of testing the nodes with the highest degree.



(a) Cost function vs number of tests for degree centrality



(b) Diffusion trend of degree centrality (c=118) compared to Random test allocation (c=140)

Figure 12: Degree centrality - The optimum number of tests to minimize the cost function is 350. We can see the improvement when testing according to the degree centrality compared to random allocation

## 3.3 Eccentricity centrality

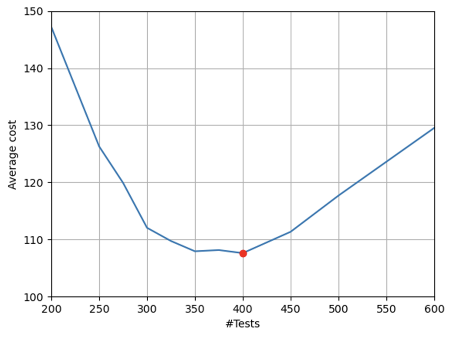
The degree centrality method is based only on the neighbourhood of a node, while from the point of view of disease spreading, the node who has the potential to spread the virus faster is more vital, which should be largely affected by the paths of propagation. Eccentricity centrality thinks that the shorter the distance a node from all other nodes, the faster the virus disseminated. To calculate the eccentricity of a node we have to compute the shortest path length of a node to all the nodes existing in the network but the eccentricity centrality only considers the maximum distance among all the shortest paths to the other nodes. The center of a network is the set of all nodes of minimum eccentricity, that is, the set of all those nodes *v* such that the greatest distance $dist_G(v,u)$ to other nodes *u* is minimal.

Before computing the eccentricity of each node, we must subdivide the graph in connected subgraphs, otherwise the eccentricity value (maximum shortest path length of each node) will be infinite. Once the eccentricity is computed, we order them according to their eccentricity in descending order, since as we said before the centrality of a node is inversely proportional to its eccentricity. We test the top-k valued nodes and we store them at the end of the buffer sequentially.

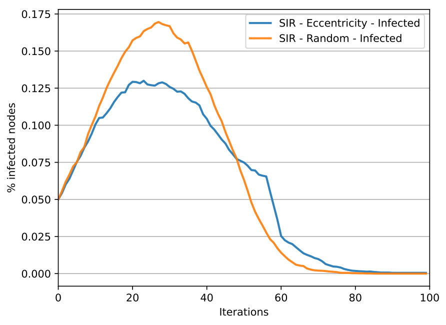
Figure 13a depicts the cost function depending on the number of tests done per day. We can observe that the minimum of the function or the optimum number of tests done per day is 400 at a cost of 107.6.

We can observe the diffusion trend for the optimum number of tests of this strategy in Figure 13b. If we focus in the trend of the infected people (light blue curve) we cannot appreciate the phenomenon previously described for the degree centrality, that is, the sharped maxima and the drastically decrease of the infected people after each maximum. This added to the fact that the cost function has similar values to the cost function of the random strategy makes us think that this centrality is not efficient. To demonstrate it, we focus in the values of the eccentricity of each node, the eccentricity only takes values among 5, 6 and 7, that is, each node can go to any other node in as much 7 steps and as minimum 5 steps. Moreover, every node has very similar eccentricity value. This also means that the erdos rainy network implemented with the defined parameters is similar to a small-world network, where most nodes are not neighbours of one another, but the neighbours of any given node are likely to be neighbours of each other and most nodes can be reached from every other node by a small number of hops or steps.

As every node has very similar values, rank them according the eccentricity centrality will lead to very similar results as the random selection strategy. Therefore, this strategy is not so efficient in this kind of networks with these parameters.



(a) Cost function vs number of tests for eccentricity centrality



(b) Diffusion trend of eccentricity centrality compared to random test allocation

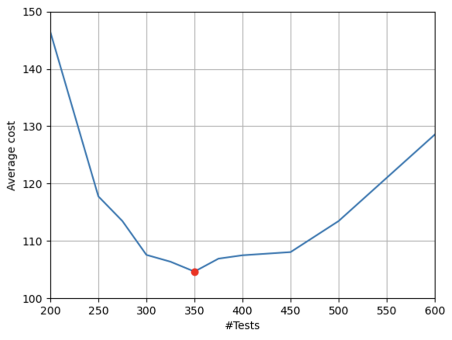
Figure 13: Eccentricity centrality - The optimum number of tests to minimize the cost function is 400. We can see the improvement when testing according to the eccentricity centrality compared to random allocation

## 3.4 Betweenness centrality

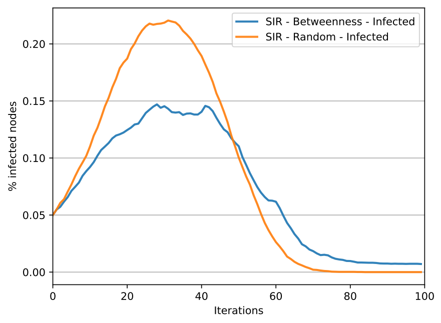
As the method above explained, betwenness centrality considers that the powerful of a node to spread a virus is largely related with the paths of virus propagation. Betweenness centrality measures the extent to which a node lies on paths between other nodes. Nodes with high betweenness may have considerable influence within a network by virtue of their ability over virus spreading between others. They are also the ones whose removal from the network will most disrupt communications between other nodes because they lie on the largest number of paths taken by virus spreading. Betweenness centrality is also useful to identify the bridges between two communities, which is a major issue because a single bridge node could infect a large number of nodes simply by passing the virus from an infected community to a “healthy” one.

We follow the same procedure that in previous methods, that is, ranking the nodes according to their betweenness centrality value, test the top-k valued nodes, removing the infected ones and storing the susceptible ones into a buffer since an epoch has been completed. We simulate this strategy eleven times for different values of number of tests per iteration and calculate the cost function trend, we can see it in Figure 14a. The optimum number of tests is 350 tests per iteration with a related cost of 104.6.

Figure 14b depicts the diffusion trend of the betweenness centrality strategy. If we look at the infected people trend we can observe the same phenomenon explained in the degree centrality; when an epoch had been completed, after 28 iterations all the not-removed nodes had been checked, occurs a sharp maximum, then the number of infected nodes starts decreasing drastically when the most influential nodes are tested and flattens when the least influential nodes are checked. This phenomenon proves that the strategy of testing those nodes that connects through the shortest path more pair of nodes is an efficient algorithm to content the spread of the coronavirus.



(a) Cost function vs number of tests for betweenness centrality



(b) Diffusion trend of betweenness centrality compared to random test allocation

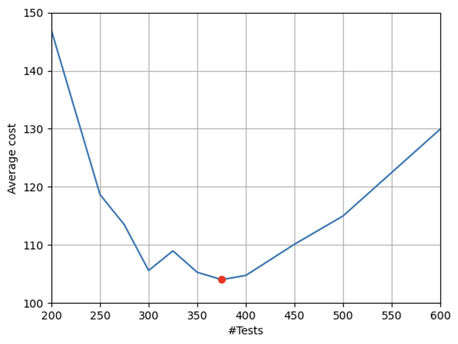
Figure 14: Betweenness centrality - The optimum number of tests to minimize the cost function is 350. We can see the improvement when testing according to betweenness centrality compared to random allocation

## 3.5 PageRank centrality

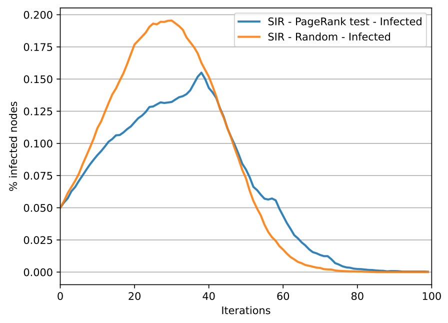
This centrality is a variant of the eigenvector centrality. The eigenvector centrality believes, as the betweenness and eccentricity centralities, that the influence of a node is not determined by the number of its neighbours, but also by the influence of each neighbour, known as the mutual enhancement effect. The centrality of a node is proportional to the summation of the centralities of the nodes to which it is connected. The PageRank centrality was initial used by Google to rank websites in Google search engine. This centrality introduces a node random jumping factor to solve the dangling node problem. There are three distinct factors that determine the PageRank of a node: (i) the number of links it receives, (ii) the link propensity of the linkers, and (iii) the centrality of the linkers.

In the graph of Figure 15a we can see the cost function over the number of tests performed, we can observe that the minimum occurs for 375 tests per day at a cost equal to 104.0.

The diffusion trend of this strategy is shown in Figure 15b. We can observe that the disease does not exceed almost the 10% of the population and it is mitigated before 80 days. We also can observe the same phenomenon of the decrease of the infected people whenever the nodes with the higher pagerank value are tested, which proves the efficiency of the pagerank centrality strategy.



(a) Cost function vs number of tests for PageRank centrality



(b) Diffusion trend of PageRank centrality compared to Random test allocation

Figure 15: PageRank centrality - The optimum number of tests to minimize the cost function is 375. We can see the improvement when testing according to PageRank centrality compared to random allocation

## 3.6 Degree discount centrality

The methods proposed above could be inefficient since the nodes of the highest centrality values may be highly clustered. These methods could be improved by removing from the network the top valued node according to some centrality and then recalculate the centrality for the rest of nodes until reach a set of k nodes. For example, if we consider that the node with the highest degree is the most influential in the network, once we have found the node with the highest degree we remove it from the network and we recalculate the centrality for the rest of the nodes until there is no more nodes remaining.

Based on the previous concept, Chen et al. (2009) proposed a so-called Degree Discount algorithm which is the strategy that we are going to propose in this section. This algorithm runs amazingly faster than greedy algorithms. Greedy algorithms are another important solution to IMP problems and have a similar procedure to the algorithm proposed in this section, they start with an empty set $S$ and at each iteration the node that maximizes the objective function is added to the output set $S$. The basic idea of the algorithm is that when considering node u as a candidate (being u a neighbour of a node v belonging to the output set $S$ of influential nodes), we should not count edge $\overline{uv}$ by taking network effect into consideration. We can see above a pseudo-code to compute the degree discount algorithm in a graph $G = (V, E)$, where $t_v$ is the number of neighbours that v has in the output set $S$ and $d_v$ is the degree of node *v*. We also consider the probability that *v* is not influenced by its immediate neighbours adding the infection rate $\beta$ to the formula. A wider demonstration of the formula can be found at Chen et al. (2009).

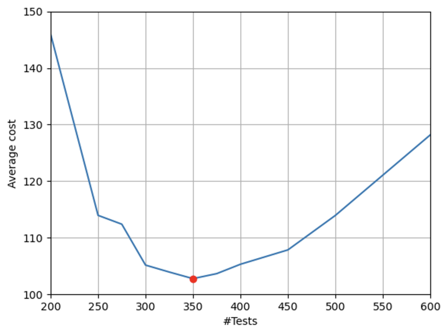
Once we have developed the code of the algorithm, we will follow the same

**Algorithm 2** Degree discount algorithm

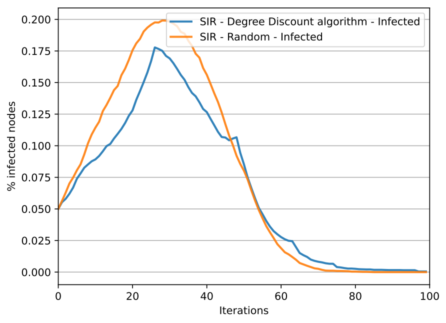
```
initialize S = $\emptyset$
for each vertex v in Network do
    Compute its degree and assign to $d_v$
    $dd_v \leftarrow d_v$
    $t_v \leftarrow 0$
end for
for i=1 to N do
    u = argmax$_v\{$dd$_v|$v $\in$ V$\setminus$S$\}$
    S = S $\cup\{u\}$
    for each neighbor v of u and v $\in$ V$\setminus$S do
        $t_v \leftarrow t_v + 1$
        $dd_v \leftarrow d_v - 2t_v - (d_v - t_v)t_v\beta$
    end for
end for
Output S
```

procedure to study the efficiency of the performance of the proposed degree discount strategy and to find the optimum number of tests. In Figure 16a we can see a graph of the evolution of the cost function though the different simulations with distinct number of tests and it is achieved the minimum at 350 tests per day with a value equal to 102.6.

In order to see how the number of infected people evolves over the time we can see in the light blue trend of Figure 16b the diffusion trend of the degree discount algorithm when 350 tests are performed in each iteration. we can observe the previous analyzed phenomena and a very similar behaviour to the degree centrality. It will be interesting to analyze the performance of the strategy for different values of the infection rate $\beta$ since is the only strategy dependent on the probability of infection.



(a) Cost function vs number of tests for degree discount centrality



(b) Diffusion trend of degree discount centrality compared to Random node selection

Figure 16: Degree discount centrality - The optimum number of tests to minimize the cost function is 350. We can see the improvement when testing according to the degree discount centrality compared to random allocation

## 3.7 Voterank centrality

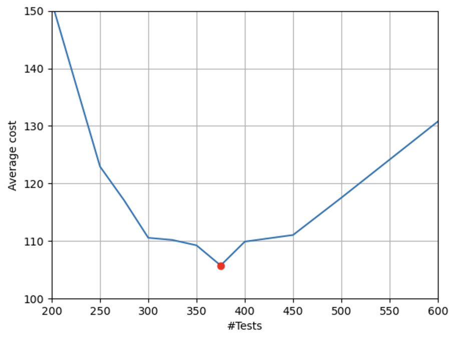
In this method each node v of the network is characterized by a tuple $(S_v, V_v)$ where $(V_v)$ is the voting ability of node *v* and $(S_v, V_v)$ its score defined as the sum of voting ability of his neighbours $S_v = \sum_1^K V_i, \qquad i = 1, 2, ..., K$ the neighbours of *v*. At the initial step, every node's voting ability is set to 1 and $S_i$ equals the degree of $v_i$. We are going to explain the algorithm with the following a pseudo-code, where $\langle d \rangle$ is the mean degree of the network.

**Algorithm 3** VoteRank algorithm

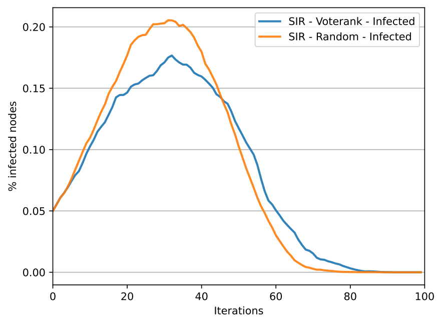
```
for each node v in Network do
    select u = argmax$_v$ $\{S_v | v \in V \backslash S$ and S = S $\cup \{u\}$
    $V_u \leftarrow 0$
    for each neighbor v of u and v$\in V \backslash$S do
        $V_v \leftarrow V_v - \frac{1}{d}$
        if $V_v < 0$ then
            $V_v \leftarrow 0$
        end if
    end for
end for
Output S
```

After implementing the algorithm in our python code, we simulate it for eleven different number of tests, compute the average cost function for each one and represent the function. We can see the function over the number of tests performed plotted in Figure 17a. The function reaches a minimum when 375 tests are performed each day with a value of the cost function equal to 105.7.

We can observe the diffusion trend for the optimum number of tests of this strategy in Figure 17b. We also observe that from the iterations when the most influential nodes are tested $\lfloor \frac{\#nodes}{\#tests} \rfloor = \lfloor \frac{10000}{375} \rfloor = 26$ onwards the number of infected nodes starts decreasing.



(a) Cost function vs number of tests for Voterank centrality



(b) Diffusion trend of Voterank centrality compared to Random node selection

Figure 17: VoteRank centrality - The optimum number of tests to minimize the cost function is 375. We can see the improvement when testing according to VoteRank centrality compared to random allocation

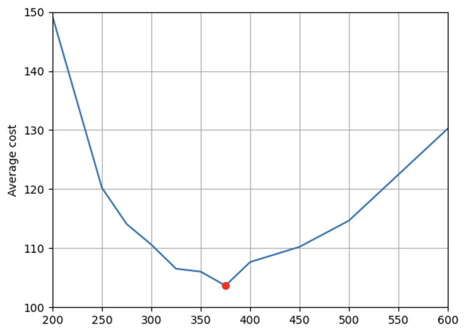
## 3.8 Community based method

Many real networks present structures based on communities. A community is a dense subnet while connections between communities are sparse. Therefore, a node is more likely to spread the virus among his community than to a different community. For this reason, is more reasonable to choose the most influential nodes from different communities rather than many nodes from one community where they could be overlapped.

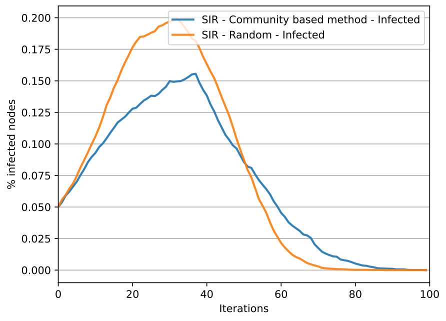
Thus, we propose the so-called community-based method which works as follows: First, the network is divided into many communities using a community detection algorithm, that we will further explain. Then all communities are ranked in decreasing order according to their sizes.

The first spreader is selected from the largest community according to a certain centrality index (in this case degree centrality). Similarly, the node with the largest centrality index in the second largest community is selected as the second spreader and sequentially since all the communities are visited. If the number of chosen spreaders is not enough, we restart the above process and choose the remaining spreaders following the same rules until k spreaders are found. Thus, the influential spreaders selected are more likely to be distributed in the network.

The algorithm used to divide the network in communities is the Clauset-Newman-Moore greedy modularity maximization. Greedy modularity maximization begins with each node in its own community and joins the pair of communities that most increases modularity until no such pair exists. We can see in Figure 18a the cost function over number of tests performed with a minimum at 375 test/day at a cost equal to 103.3. We can also see at Figure 18b the diffusion trend of this method with a reasonable good performance in the virus spreading mitigation.



(a) Cost function vs number of tests for community-based strategy



(b) Diffusion trend of community-based strategy compared to Random node selection

Figure 18: Community based method - The optimum number of tests to minimize the cost function is 375. We can see the improvement when testing according to the community based method compared to random allocation

All the strategies that we have proposed so far are static in the sense of simulation iterations since the algorithm is performed at the initial iteration, nodes are ranked according to certain centrality and stored in a buffer. Onwards the first iteration the only thing we do is manage the buffer to meet a trade-off between importance of people within the network and fairness of testing every people at least once per epoch. However, it will be very useful to have some information about the status of the nodes before selecting which nodes we test at each iteration. Thus, we are going to propose some dynamic strategic based on this idea.

One step further in identifying a target set of nodes to test to mitigate a pandemic diffusion is contact tracing, where the potential risk-nodes are the primary focus. We define the risk-nodes as those that have been in contact with a registered positive node. This technique is the standard tool for eliminating minor outbreaks in the latter stages of disease eradication, especially when the disease may be asymptomatic (which is one of the main problems in the disease COVID-19).

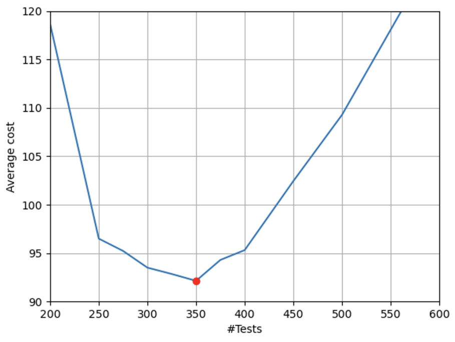
In order to know which nodes are the potential risk-nodes we should store which nodes had tested positive so far and consider testing the nodes who have been in direct contact to those positive-tested nodes. We are going to present two strategies and we will study their performance.

## 3.9 Contact tracing: Most infected neighbours

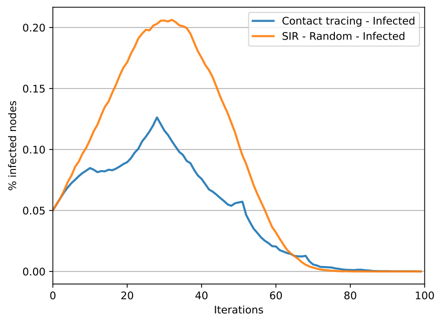
The first tracing strategy will consist in testing at each iteration the nodes that have the most infected neighbours. We must remark that when we refer to tracing infected nodes, we only refer to those nodes that had been tested positive, that is nodes with Removed status or status equal to 2 (does not matter if they are true positives or false positives) because the vast majority of infected nodes will have not been detected yet and will remain at Infected status, that is status equal to 1. The algorithm will work as follows: At the initial iteration, since we have no information of which nodes are infected, we rank it according some static centrality and select the top-k valued ones. Onwards, at each iteration, it will be dynamically recalculated the number of infected neighbours that each node has, the nodes will be ranked according it and the top-k valued nodes will be selected.

We run this algorithm eleven times for distinct number of tests performed at each iteration and a cost function that has been previously explained will be compute. We can see below in Figure 19a the cost function over time, with an optimal value at 350 tests/day at a cost equal to 92.1

Figure 19b depicts the diffusion trend of this contact tracing strategy over the time. If we focus in the evolution of infected people (light blue trend) we see that in the first iterations the growth of infected people is the highest one of the whole simulation since we have not yet enough information of the health status of the nodes to decide which ones are the critical nodes. However the number of infected people does not increase much further from the initial infected fraction. We also observe that after each epoch, that is, when the most influential nodes are tested again the number of infected people starts decreasing considerably.



(a) Cost function vs number of tests for contact tracing (MIN)



(b) Diffusion trend of contact tracing (MIN) compared to Random node selection

Figure 19: Contact tracing method - The optimum number of tests to minimize the cost function is 350. We can see the improvement when testing the neighbors of infected people compared to random allocation

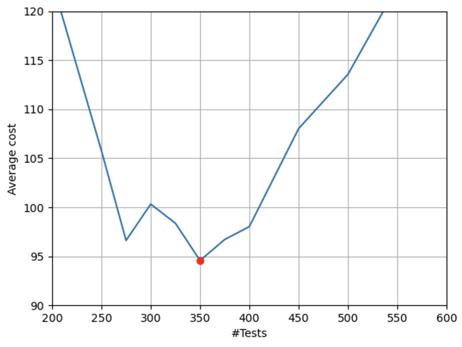
## 3.10 Contact tracing combined with a centrality

In the context of contact tracing methodology, we are going to propose a new strategy. We have previously defined the risk-nodes as those nodes that are directly connected to an already detected infected node. We remind that when we talk about infected nodes in contact tracing, we only consider the already detected infected nodes, so whenever a new positive has been detected in the network, we add its neighbours to the “risk nodes” database, avoiding duplicated nodes in the database.

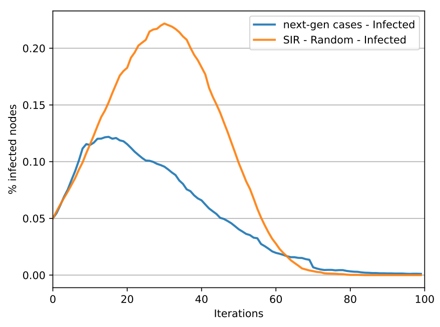
Before each iteration we will dynamically rank all the nodes belonging to those “risk nodes” according to one centrality of those presented in this document, in this section we will consider the degree centrality due to the low complexity and reasonable good performance. When the risk nodes are ranked, we will check the health status of the top-k valued risk nodes. If a node tests positive, we will remove the node from the network and add its neighbours to the risk-nodes database. Contrary, if tested negative we will add it to the end of the buffer in order to meet the mentioned fairness trade off.

We perform this algorithm for 100 iterations in each simulation each of them with different values of nodes tested. In Figure 20a we can observe the cost function, the optimum value is reached when 350 tests are performed at each iteration at a cost equal to 94.5.

Figure 20b depicts the diffusion trend of the ranking the risk nodes strategy. As we have mentioned in the previous strategy, in the first iterations the number of people infected increases as there is not enough information about which nodes are infected. After the initial days, the number of infected people starts decreasing smoothly but never increases again.



(a) Cost function vs number of tests for contact tracing (Risk nodes)



(b) Diffusion trend of contact tracing (Risk nodes) compared to Random node selection

Figure 20: Contact tracing plus centrality method - The optimum number of tests to minimize the cost function is 350. We can see the improvement when ranking according a centrality the neighbors of detected COVID+ people compared to random allocation

## 4 Result discussion

So far, we have presented ten different strategies some of them where static based on ranking nodes according different centralities and we have finally proposed two dynamic strategies based on contact tracing. We are going to present a summary of the optimum number of tests for each centrality and the related cost in Table 2 and some graphics comparing the cost functions between centralities Figure 24 and between contact tracing strategies Figure 25.

| **Strategy** | **Optimum # tests** | **Related cost** |
|---|---|---|
| Random | 325 | 107.5 |
| Degree | 350 | 101.7 |
| Eccentricity | 400 | 107.6 |
| Betweenness | 350 | 104.6 |
| PageRank | 375 | 104.0 |
| Degree discount | 350 | 102.6 |
| VoteRank | 375 | 105.7 |
| Community-based | 375 | 103.3 |
| Contact tracing 1 | 350 | 92.1 |
| Contact tracing 2 | 350 | 94.5 |

Table 2: Summary of optimum number of tests and related cost

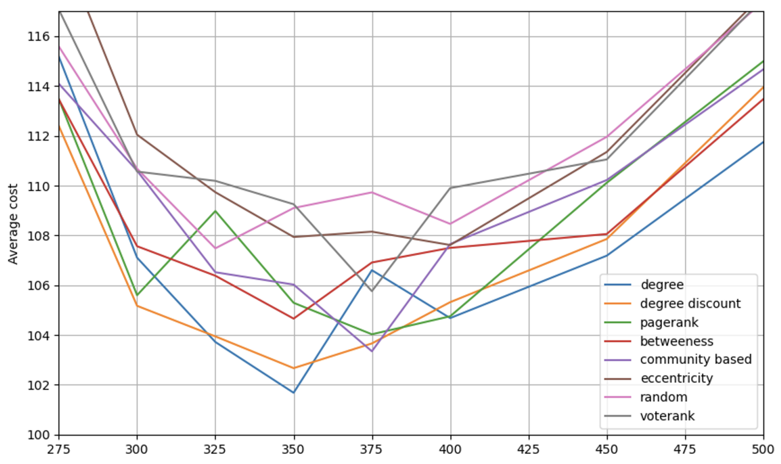


Figure 21: Static (centrality) strategies comparison

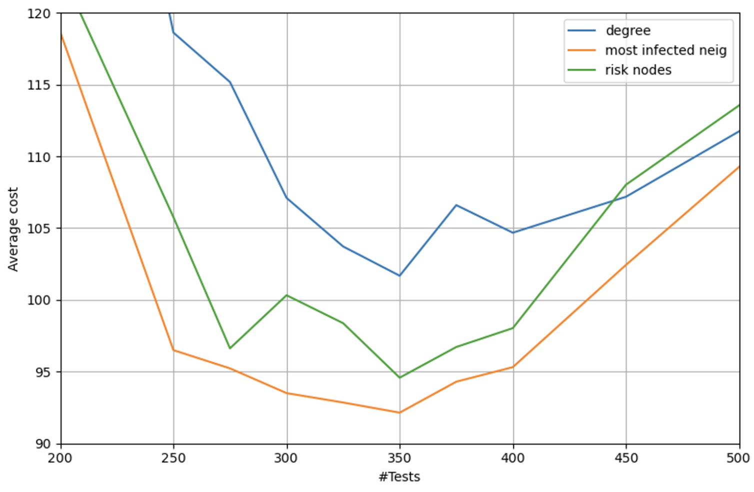


Figure 22: Contact tracing comparison vs centrality strategy

# Part IV

# Extended diffusion model - SEPIA

## 1 Introduction

We have studied in Part III: *Strategies for test allocation*, a simple COVID-19 diffusion model considering three possible states of the nodes: Susceptible, Infected and Removed. In this simple model we treated all the infected nodes equally regardless the severity of the symptoms, we did not considered the possibility of being recovered and re-infected and not even the deaths due to the disease. Nevertheless, it was sufficient for the purpose of the previous Part of defining strategies to detect the most important nodes to test at each simulation step.This Part aims to extend the previously described model resembling it to the reality.

The beginning of the COVID-19 pandemic was full of uncertainty as an inadequate understanding and a sense of incomplete, ambiguous or unreliable information. However, since the outbreak of the pandemic we have gained experienced and been able to draw some conclusions on the spreading behaviour of the COVID-19 virus. People are more contagious 2-3 days before having symptoms (Preidt (2021)); the first hours from the time a person is exposed, early stages of incubation, people are unlikely to spread the virus; People showing no symptoms are unlikely to die; Duration of viral shedding in asymptomatic people is lower than in people showing symptoms (8 compared to 19 according to Yang et al. (2020)); asymptomatic infections are known to have the same transmission rate as symptomatic infections (Chen et al. (2020)).

We first conducted an extensive review of the literature (Part II) to study already proposed models of COVID-19 diffusion so we can better adapt the model as close as possible to reality. We will propose in this Part our model of COVID-19 diffusion. Many mathematical models have been developed to try to describe the diffusion of the COVID-19 epidemic in individual countries or at global scale. Actually, no clear consensus has been reached on the different compartments that should be included in a proper model. Our model choice was motivated by a review of the existing approaches (see Part II).

Most models assume a standard SEIR structure but make different hypotheses on the nature of the different compartments and their respective residence time.

## 2 Proposed model

In this section we propose our model, defined by the possible statuses of each of the nodes and the dynamic equations. We will also provide with the rationale on why we have designed this model.

The main reasons to expand the number of statuses from the SIR diffusion model described in Part III, described by Susceptible, Infected and Removed statuses, are:

- When an individual first comes into contact with a virus or bacteria is COVID-19 infected but are not yet infectious, i.e. capable of transmitting the disease to others, for a very short time, they are called Exposed (or latent) individual. Some literature states that being exposed to the virus not always leads to an infected individual if, for example, the amount of virus that enters the body is not in a large enough quantity, or if the body's immune system is able to quickly fight it off. However, in this document we assume that every time an individual becomes exposed it will lead into a COVID-19 infection. (Buonomo and Marca (2020))

- An individual infected of COVID-19 can transmit the virus before showing symptoms. To demonstrate this assumption we first introduce two concepts: the *Serial interval* and the *Incubation period*. The serial interval, in epidemiology, are measured from when one infected person starts to show symptoms to when the next person infected becomes symptomatic (Ali et al. (2020)). Whereas the incubation period is the time duration between exposure to the pathogen and the appearance of the disease symptoms (dictionary (2022)). Alene et al. (2021) combined on an exhaustive research a total of 23 and 14 studies to conclude that the serial interval (weighted pooled mean: 5.2 days) was lower than the incubation period (weighted pooled mean: 6.5 days). We can conclude that an individual can infect before showing symptoms, we call this status Pre-symptomatic.

- There are individuals that do not show any signs that the COVID-19 virus is present in their body but are actually infected. They are called asymptomatic individuals.

- We finally assume that individuals showing mid or no symptoms cannot die

- The number of simulation steps (analogue to days) we will consider throughout this document will be small enough to assume that a recovered node will not be reinfected. Xiao et al. (2021) showed that the antibodies can last for more than 12 months, remaining stable for the first 6 months. Thus, we consider in this document negligible the reinfection value.

- Tests performed to identify infected nodes are not 100% accurate. Indeed, a test is characterized by its confusion values, i.e., to test positive when the node is actually not infected and to test negative when the node is actually infected. We define:

$$\begin{aligned} \rho_{ip} &= P(\text{ill} \mid \text{positive}) \\ \rho_{np} &= P(\text{not ill} \mid \text{positive}) \\ \rho_{in} &= P(\text{ill} \mid \text{negative}) \\ \rho_{nn} &= P(\text{not ill} \mid \text{negative}) \end{aligned} \tag{10}$$

Taking into account the previous assumptions, we now define the statuses of our model:

- **Susceptible**: an individual that can be infected if exposed to the infectious virus
- **Exposed**: an individual infected with the virus but is not yet infectious
- **Presymptomatic**: an individual infected of COVID-19 that can transmit the virus before showing any symptoms
- **Infected** o Symptomatic: an individual showing severe symptoms directly related to the virus
- **Asymptomatic**: an individual infected showing no or mid symptoms
- **Recovered**: an individual previously infected with the virus and now being immune to infection and consequently not affecting transmission when they come into contact with other individuals
- **Quarantined**: an individual tested positive for the virus
- **Dead**: an individual dead from causes directly linked to the virus

We call this model, the SEPIA diffusion model, which stands for the Susceptible status plus all the statuses representing the infected ones (Exposed, Pre-symptomatic, Infected, Asymptomatic). It is worthwhile noting that we also call the Symptomatic status as Infected, but infected statuses also include Exposed, Pre-symptomatic and Asymptomatic. It could lead to misunderstanding but we will try to avoid confusion throughout the document.

The status variables and the processes included in the model are illustrated in the flow chart in Figure 23. This flow chart describes the possible statuses that each status can transit to and the rate of transition. In Table 3, we provide a description for each parameter as well as the value for each parameter. The rationale on using these values will be given in Section 2.1 in this Part. Those parameters without value assigned in Table 3 mean that the value is not fixed and depends on the simulation.

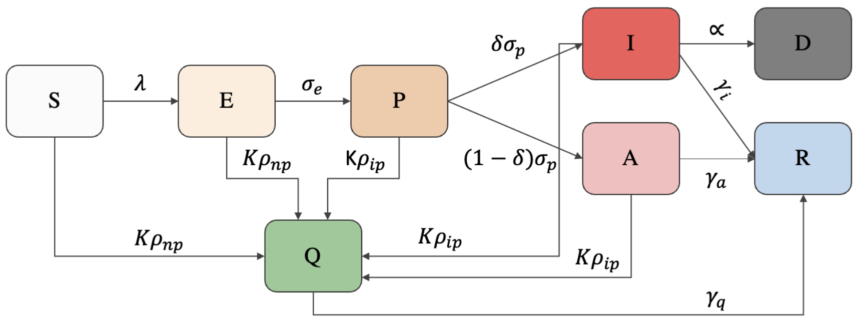


Figure 23: Flow chart for the COVID-19 model

The model is given by the following system of nonlinear ordinary differential equations, where each balance equation rules the rate of change of a state variable:

$$\begin{aligned}
&S = -\lambda.S - K\rho_{np} \\
&E = \lambda.S - \sigma_E.E - K\rho_{np} \\
&P = \sigma_E.E - \sigma_P.P - K\rho_{ip} \\
&I = \sigma_P.\delta.P - \gamma_I.I - K\rho_{ip} \\
&A = \sigma_P.(1-\delta).P - \gamma_A.A - K\rho_{ip} \\
&R = \gamma_I.I + \gamma_A.A \\
&D = \alpha_I.I \\
&\lambda = \frac{\beta_P.P + \beta_I.I + \beta_A.A}{N - D - R - Q}
\end{aligned} \tag{11}$$

| **Parameter** | **Description** | **Value** |
|---|---|---|
| $\lambda$ | Exposure rate | – |
| $\beta_P$ | Transmission rate of pre-symptomatic individuals | 0.12 |
| $\beta_I$ | Transmission rate of symptomatic individuals | 0.035 |
| $\beta_A$ | Transmission rate of asymptomatic individuals | 0.035 |
| $K$ | Number of tests | – |
| $\sigma_E$ | Latency rate ($d^{-1}$) | 0.3 |
| $\sigma_P$ | Post-latency rate ($d^{-1}$) | 0.8 |
| $\delta$ | Symptomatic rate | 0.6 |
| $(1-\delta)$ | Asymptomatic rate | 0.4 |
| $\gamma_I$ | Recovery rate of people with symptoms ($d^{-1}$) | 0.1 |
| $\gamma_A$ | Recovery rate of people with no symptoms ($d^{-1}$) | 0.1 |
| $\gamma_Q$ | Quarantine rate ($d^{-1}$) | 0.07 |
| $\alpha$ | Fatality rate | 0.014 |
| $\langle k\rangle$ | Average degree | 5 |
| $R_0$ | Basic reproduction number | 3 |

Table 3: Summary of SEPIA model parameters

## 2.1 Design of the proposed model

### 2.1.1 Reproduction number

We now introduce the basic Reproduction number. In epidemiology, the basic reproduction number $R_0$ (R naught) of a pandemic is defined as the expected number of secondary cases a primary individual can cause in a susceptible population. If it is greater than one, then an epidemic is expected to spread after the outbreak of the infection. Whereas, if $R_0$ is lower than one, the outbreak can be contained and the disease is expected to decrease.

The effective reproduction number, $R_e$, also denoted as $R_t$, is the number of people a contagious individual can infect at a specific time, where some individuals can have immune or protected to be infected. Real-time estimates of $R_t$ are a key topic for policy decisions during a pandemic (Leung et al. (2020)). Effective reproduction number estimations can be used to study the effectiveness of non-pharmaceutical interventions (NPIs) such as testing, which is the main topic throughout this report.

The definition of $R_0$ in a classic SIR model can be approximated to:

$$R_0 = \beta N \sigma \tag{12}$$

Where $\beta$ is the transmission probability, $N$ is the average number of contacts and $\frac{1}{\sigma}$ is the length of the infectious period.

In our SEPIA model, based on equation 12 and in the absence of quarantine intervention, the basic reproduction number can be approximated as:

$$\begin{aligned} R_0 &= \frac{\beta_P N}{\sigma_P} + \frac{\delta \beta_I N(1-\alpha)}{\sigma_I} + \frac{(1-\delta)\beta_A N}{\sigma_A} \\ &= N\left(\frac{\beta_P}{\sigma_P} + \frac{\delta\beta_I(1-\alpha) + (1-\delta)\beta_A}{\sigma_I}\right) \end{aligned} \tag{13}$$

According to Mahase (2020) the case fatality rate ($\alpha$) is 1.38%.According to Lavezzo et al. (2020) the distribution between symptomatic individuals and asymptomatic individuals is 60% - 40%. The transmission rate of symptomatic and asymptomatic individuals is the same as the viral load is similar for symptomatic and asymptomatic individuals (Lavezzo et al. (2020)). The post-latency rate, i.e., the pre-symptomatic rate is 0.8 $days^{-1}$. In Walsh et al. (2020) review, for 5 of the 13 studies (with culture attempted in at least 76 patients), the last day on which SARS-CoV-2 could be cultured (ability to infect) occurred within the first 10 days since onset of symptoms. Thus, the symptomatic rate and asymptomatic rates are equal to 0.1 $days^{-1}$.

Assuming the above described values for the parameters of the model, there remains three variables in the equation $R_0$, post-latency transmission rate ($\beta_P$) and symptomatic-asymptomatic transmission rate ($\beta_{I,A}$). Estimating the transmission rates from observations has not yet been well characterised. We propose in this document to estimate it from already observed $R_0$ in the literature. He et al. (2020) estimates that the of 44% of secondary cases were infected during the presymptomatic stage. Considering that the duration of infectiousness of pre-symptomatic individuals is $\frac{0.8}{0.1}$ times lower, the estimation in He et al. (2020) translates in our model that pre-symptomatic transmission rate is 3.25 times grater than $\beta_{I,A}$. Our $R_0$ equation in 13 is:.

$$\begin{aligned} R_0 &= 6.25\beta_P + 49.3\beta_{A,I} \\ &= 6.25\beta_{A,I}3.25 + 49.3\beta_{A,I} \\ &= 69.61\beta_{A,I} \end{aligned} \tag{14}$$

Doing an exhaustive review, we conclude that $R_0$ ranges between 2 and 3.5. We assume in our model that $R_0$ is equal to 3 so we can infer the transmission rate values. A summary of all the values of the parameters used in our SEPIA model are listed in Table 3.

### 2.1.2 Topology of the network

Barabási (2016) lists several examples of real undirected networks and the average degree of each of them. The average degrees of each real network shown by Barabási (2016) exceed $\langle k \rangle > 1$ but are below the giant threshold $\langle k \rangle < \ln N$, being $N$ the number of nodes, this means that the network is expected to be broken into numerous isolated components. We will consider a network with a number of nodes ranging between N = 10000 and N = 100000. Thus, the average degree must range between $\langle k \rangle = 1$ and $\langle k \rangle = \ln 100000 \approx 11$ to be close to reality. We assume an average degree equal to 5, meaning that each individual is on stretch contact to 5 individuals in average.

## 2.2 Implementation of the proposed model

Once that we have our model well designed we implement it on Python. Similarly as in III: *Strategies for test allocation*, the network is an Erdos Renyi graph network of type networkx. It is described by the Number of edges, N = 10000 or N = 100000 depending on the experiment, and the probability of edge creation, we set it to lead to an average number of neighbors per node $\langle k \rangle$ equal to 5.

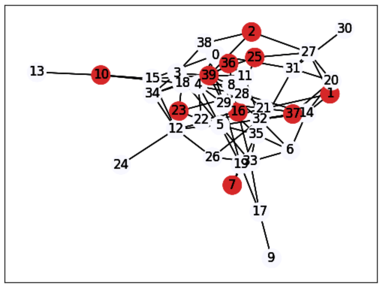
The Diffusion model is characterized as an object of class DiffusionModel from the python library ndlib. This class behaves as a bunch of simulation iterations. The graph starts with a fraction of infected nodes and at each simulation step or iteration, every node is visited to determine the individual node status transitions. As an example, if a node is Susceptible, first we calculate all the neighbors nodes of each of the contagious states (Pre-symptomatic, Infected and Asymptomatic) and the exposure rate is calculated. It is worth wile noting that the exposure rate depends also on the number of infectious nodes for each of the contagious states, i.e., if the node is in contact to two Pre-symptomatic the exposure probability will be higher than if it is in contact with only one Pre-symptomatic. Once the exposure rate is calculated, if it is greater than a random generated number, the node transitions to Exposed and subsequently with each of the eight possible states.

Figure 24 depicts an example of simulation process of a graph network generated with *N = 40* from a self developed code, the code can be found in the *feature-branch* branch on the GitHub repository. The visualizations are made with networkx method draw_networkx and the different statuses are re'resented by the different colors, Susceptible = white, Exposed = peachpuff, Pre-symptomatic = orange, Infected = red, Asymptomatic = salmon, Recovered = blue, Quarantined = green and Dead = grey.

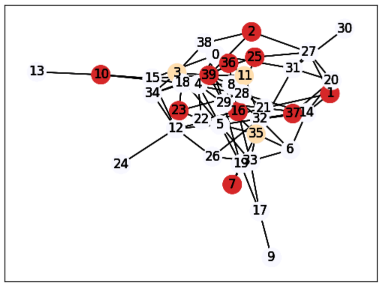
In the second simulation step (Image 24b in Figure 24) three nodes, which are in stretch contact to already infected nodes, became Exposed, leading to symptomatic or asymptomatic individuals in the subsequent steps. In the third simulation step (Image 24c in Figure 24) two nodes become Quarantined as a result of testing positive, we can also see in this Image how an infected node dies.

Showing the dynamic in such way becomes unpractical when the number of nodes becomes significantly high. Thus we will show the overall trends with the python library bokeh where each line represents the evolution of nodes belonging to each status. The DiffusionModel class contains a method to visualize the trends of each status as shown in Figure in Image 25a in Figure 25 and zoomed in Image 25b.

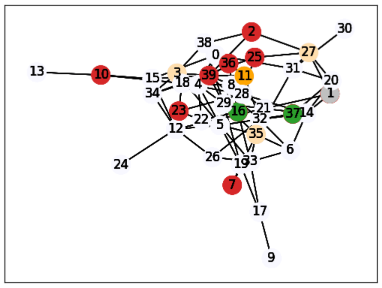
However, showing all the statuses overcomplicates the visuaization for analysis purpose. We then modify this class and proposes a new method to build the diffusion trends.



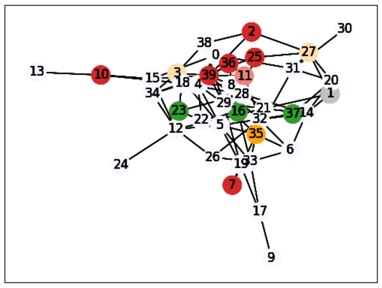
(a) Step 1: $25\%$ of the network is infected



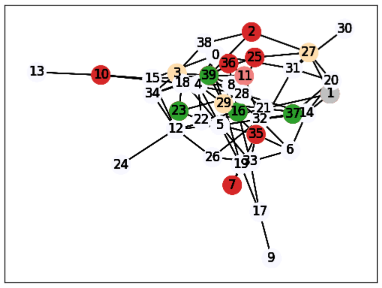
(b) Step 2: Some nodes become Exposed after being in contact with Infected nodes



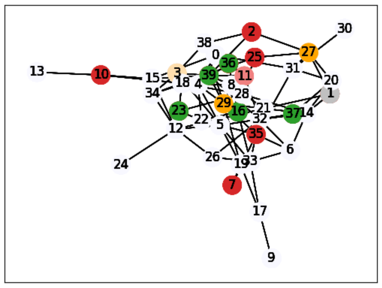
(c) Step 3: Green nodes goes from Infected to Quarantined, grey one has died and orange one has became Pre-symptomatic



(d) Step 4: Node 11 went from Pre-symptomatic to Asymptomatic, more nodes being Quarantined after tested positive



(e) Step 5: Node 33 went from Pre-symptomatic to Symptomatic



(f) Step 6: Node 36 is recovered and nodes 27 and 29 goes from exposed to pre-symptomatic

Figure 24: SEPIA dynamics.

In the visualization designed in this thesis, only the statuses that are relevant are shown, i.e., the infected statuses: Exposed plus contagious ones (Pre-symptomatic, Symptomatic and Asymptomatic), Quarantined and Daed statuses. Furthermore, the infected nodes should be shown as an aggregated to better understand the number of nodes infected of COVID-19. Thus, we develop the stacked visualization for infected statuses.

Figure 25 depicts a comparison of the visualization trends of the same simulation process. Top image (Image 25c) is the inherit method of the DiffusionModel ndlib class, the middle image as the top one but zoomed and the bottom image (Image 25c) is the visualization developed in this thesis. The latter is clearly more convenient for visual analysis and gives a better understanding of the dynamics of the simulation.

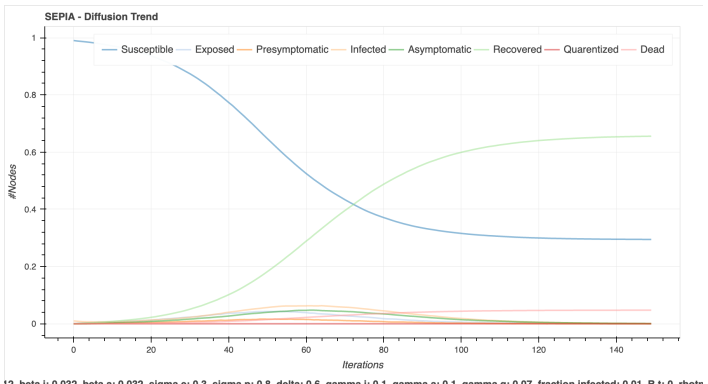
## 2.3 Reproduction number representation

Keeping the reproduction number below one is an indicator of pandemic diffusion containment. $R_0$ varies with time during the course of the pandemic and is dependant on epidemiological factors like susceptible population characteristics, disease transmission rates, and control measures adopted. Our goal now will be to maintain the basic reproduction number below 1 to control the outbreak of the pandemic.

We use an already existing estimate approach to represent the reproduction number over time, the python epyestim package in Hilfiker and Josi (2022). This method estimates the effective reproduction number from time series of reported case numbers assuming that there is a delay from actual infection to the event registration, i.e., the report of the infected case. We adapt this package to adapt our model as they assumed a mean of 10.3 from infection to reporting which is higher than our model.

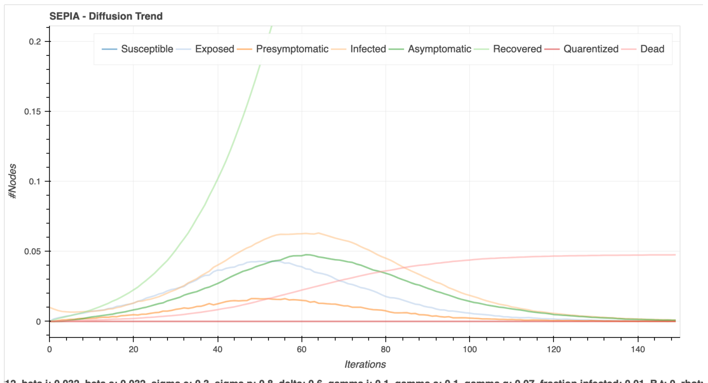
To better understand how can we measure how effective a strategy is by looking to the reproduction number over time we will plot the evolution of $R_t$ for different number of tests performed at each simulation step in Figure 26.

If we focus in the right side of the Figure, the reproduction number evolution is plotted for three different simulations. The first one, Figure 26b, shows the evolution of $R_t$ when no tests are performed, i.e. in the absence of any intervention. Indeed the value of $R_t$ in the early stage of the simulation reaches 2.2 which it is actually the value that we fixed to the basic reproduction number $R_0$ when designing the diffusion model. On images 26d and 26f the value of $R_t$ is lower as the number of tests meaning that the pandemic can be contained when the number of tests performed increases. However, the evolution of the reproduction number



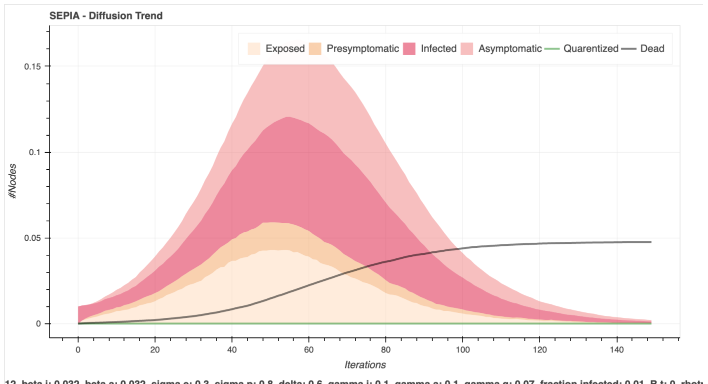
12, beta i: 0.032, beta a: 0.032, sigma e: 0.3, sigma p: 0.8, delta: 0.6, gamma i: 0.1, gamma a: 0.1, gamma q: 0.07, fraction infected: 0.01, B t: 0, rhotp

(a) Ndlib Diffusion trends visualization



12, beta i: 0.032, beta a: 0.032, sigma e: 0.3, sigma p: 0.8, delta: 0.6, gamma i: 0.1, gamma a: 0.1, gamma q: 0.07, fraction infected: 0.01, B t: 0, rhotp

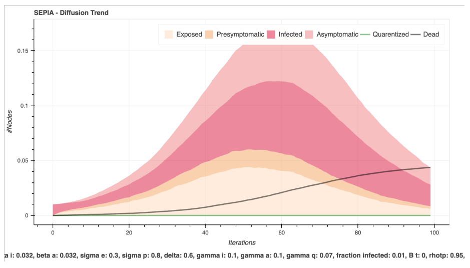
(b) Ndlib Diffusion trends visualization (zoomed)



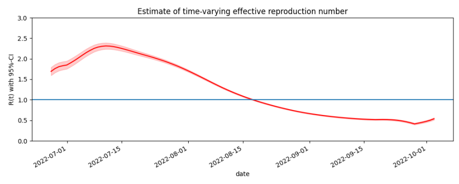
12, beta i: 0.032, beta a: 0.032, sigma e: 0.3, sigma p: 0.8, delta: 0.6, gamma i: 0.1, gamma a: 0.1, gamma q: 0.07, fraction infected: 0.01, B t: 0, rhotp

(c) Designed diffusion trends visualization

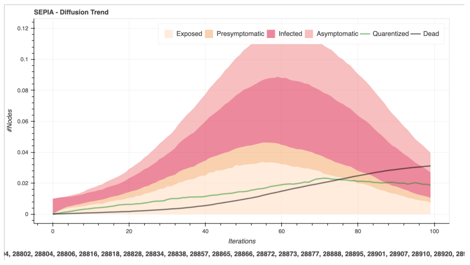
Figure 25: SEPIA diffusion trends



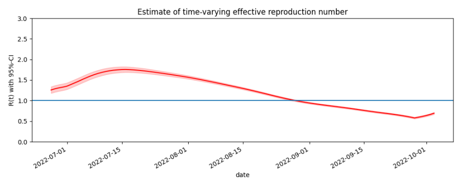
(a) Diffusion trends of the pandemic with no tests performed



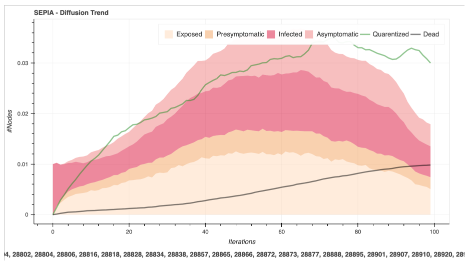
(b) Evolution of $R_t$ over time with no tests performed



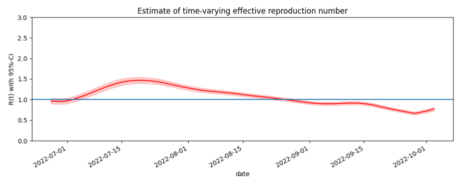
(c) Diffusion trends of the pandemic with 1000 tests performed at each step



(d) Evolution of $R_t$ over time with 1000 tests performed at each step



(e) Diffusion trends of the pandemic with 5000 tests performed at each step



(f) Evolution of $R_t$ over time with 5000 tests performed at each step

Figure 26: Reproduction number over time for different number of tests in a network with 1e5 individuals

needs to be shown alongside the diffusion trends to better interpret the results. For example, if we only focus on figure 26b, in the middle of the simulation process (by the 15th of August) $R_t$ is lower than 1, meaning that the pandemic has been contained and we can infer that we have reach our goal. However, when looking at the Figure in 26a the portion of people infected at this simulation step is over the 30% of the population and effectively the value of $R_t$ is less than 1 because, as its own definition says, there are no Susceptible nodes to infect since they are already infected, Recovered, meaning they are immune, or Dead, which is an undesired situation.

In the next section V, we seek the goal of contain the virus by designing non-pharmaceutical international strategies. We will leverage the model designed and implemented in this section, using the two tools designed in this section: the stacked visualization trends and the evolution of the Reproduction number.

# Part V

# Reinforcement Learning

## 1 Introduction and Objectives

We have designed and implemented in Part IV a mathematical model that represents the diffusion of the COVID-19, the SEPIA model. We have seen the importance of implementing non-pharmaceutical intervention strategies such as massive testing and quarantine to mitigate the spread of the pandemic and even contain the spread when the reproduction number is lower than 1. We have also introduced the $R_t$ a tool to estimate how controlled the pandemic is.

In Part III: *Strategies for test allocation*, we implemented and discussed the effective of different approaches to allocate tests among the network in order to contain the diffusion of the COVID-19 pandemic. We divided these strategies into centralities, based on finding the most important nodes within a network, and contact tracing, based on testing the neighbors of the already detected ill nodes. We discussed that each of the defined strategies performs better under some conditions and they are dependent on the characteristic of the network and the model.

In this Part, we will leverage the knowledge gathered in the previous Parts to design an effective non-pharmaceutical intervention strategy to contain the diffusion of the pandemic based on performing massive testings. However, we consider, as we did in Part III, that tests are scarce and costly. We have considered so far a constant number of COVID tests done at each step regardless the number of infected individuals. However, it would lead to wastage of COVID tests when there are few infected people, and an under-usage of them when the number of infected people is high, the so-called peak of COVID waves.

To optimize the usage of tests, we will need to know some insights about the state of the network, such as the actual number of infected people, the number of deaths or any available information on the environment in order to perform reactive strategies.

This problem is similar to a Black box optimization problem, where we don't know how the box behaves but we are able to take some actions and change the behaviour of it receiving a set of outputs from the box. A black box optimization involve the execution of a computer code or simulation.

Reinforcement learning is a similar technique as the black box optimization problem. It is a branch of machine learning different to the supervised and unsupervised learning, it is consider as the third branch. Reinforcement learning is based on the interactions between an agent and an unknown or unpredictable environment. The agent can decided among a set of predefined actions and perceives an observation from the environment lead by the action previously taken. The agent learns by interacting with the environment seeking to take the actions that leads to a maximum reward or goal(Sutton and Barto (2018)).

## 1.1 Optimization techniques

In this section we take a look at the basic concepts and definitions of black box optimization technique which will help us to connect and introduce to the Reinforcement Learning technique that will be used in this document to design interventional strategies to mitigate the diffusion of the COVID-19 pandemic.

Reinforcement learning and Black box optimization are the two main approaches to performing optimization of policy improvement methods. This methods goal is to optimize the parameters of a policy, a policy can be easily defined as a function that returns an action given a state or observation of the system.

### 1.1.1 Black box optimization

A black box is a system that can bee seen as a set of input and output without any knowledge about the performance within the system.

Bayesian optimization is a sequential design strategy for global optimization of black-box functions, this kind of optimization is advantageous when the objective function is unknown. The function is treated as a random function and place a prior (beliefs about the behaviour of the function) over it.

According to (Carson and Maria (1997)) Simulation optimization is defined as the process of finding the optimum set of input data among all the possible inputs without evaluating each possibility. A simulation experiment can be defined as a test or a series of tests in which meaningful changes are made to the input variables of a simulation model so that we may observe and identify the reasons for changes in the output variables. When the number of input variables is large and the simulation model is complex, the simulation experiment may become computationally prohibitive. The objective of simulation optimization is minimizing the resources spent while maximizing the information obtained in a simulation experiment.

## 1.2 Reinforcement learning

Reinforcement learning problems involve learning what to do so as to maximize a numerical reward signal. Three properties that characterize RL is that it is a closed-loop system, discovery and subsequent effect. Closed-loop as the learning system's actions influence its later inputs. The learner is not told which action to take, but instead must discover which actions yields to the most reward by trying them out. Actions not only affect the immediate reward but the subsequent ones.

The most important feature is the ability to evaluate the actions rather than instruct by giving correct actions. If you have the estimates of all the actions you can take for a specific state, the greedy one will b the one with the highest value. However, due to the uncertainty of the environment there could be another action that brings a higher value in the long run. Taking the greedy action is called exploitation while taking non-greedy actions is called exploration. Having a trade-off between exploitation and exploration is what characterizes a RL algorithm.

A **policy** defines how the agent behaves in a specific situation. Maps perceived states of the environment to actions to be taken.

On each time step the environment sends to the agent a single number, the **reward**. The agent's sole objective is to maximize the reward over the long run. In general, the reward signal can be seen as a stochastic function of the environment states the actions taken. Whereas rewards represents what is good in immediate sense, **value function** specifies what is good in the long run. The *value* of a state is the total reward that an agent can expect to accumulate starting from that state. We are more concerned about values when making a evaluating decision. *Values* must be estimated and re-estimated from the sequences of observations an agent makes over its entire lifetime.

**The model** mimics the behaviour of the environment i.e. allows to predict the next state and reward given an action and current state.

### 1.2.1 Tabular Solution Methods

These methods can be applied when the action-state space is small enough to be represented with an array or a table.

The multi-arm bandits are a special type of problems where there is only one possible state. Suppose that we face at each time step to n possible actions, each of them leads us to an expected or mean reward given that that action is selected; let us call this the **value of that action**. We define as *greedy action* to the selection of the action with the highest estimated value, if we select it we are *exploiting* our knowledge of the values. If instead we select one of the non greedy actions, then we say you are exploring. Exploitation is the right thing to do to maximize the expected reward on the one step, but exploration may produce the greater total reward in the long run ((Sutton and Barto (2018))).

### 1.2.2 Finite Markov Decision Processes

The interaction between the agent and the environment is depicted in Figure 27.

The agent's goal is to maximize the cumulative reward it receives in the long run. The return $G_t$ is defined as some specific function of the reward sequence (the sum of the rewards):

$$G_t = R_{t+1} + R_{t+2} + ... + R_T, \tag{15}$$

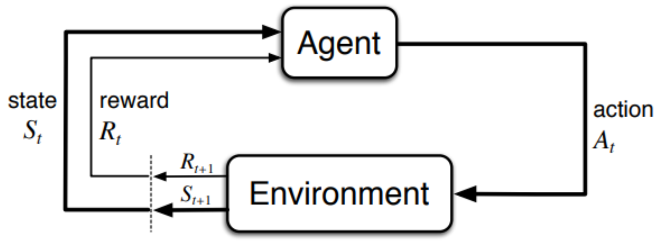


Figure 27: RL interface. Sutton and Barto (2018)

where *T* is a final step, this notation makes sense when the agent-env interaction breaks in subsequences, called *episodes*. Each episode ends in a special state called the *terminal state*, followed by a reset to a standard starting state or to a sample from a standard distribution of starting states. Task with episodes are called *episodic tasks*. We need to distinguish the set of all nonterminal states, denoted S, from the set of all states plus the terminal state, denoted $S^+$. The intuitive notation to refer to the state at the time step *t* of the episode *i*, would be $S_{t,i}$ but it is commonly referred as $S_t$ since we will almost never need to distinguish between different episodes. (Sutton and Barto (2018))

If we consider the task as a continuing task (the agent-env interaction do not break naturally into identifiable episodes) we should add an extra concept: the *discount rate*, which determines the present value of future rewards: a reward received *k* time steps in the future is worth only $\gamma^{k-1}$ times what it would be worth if it were received immediately.

### 1.2.3 The Markov Property

In the RL framework, the agent makes its decisions as a function of the state signal retrieved from the env. A property of environments and their state signals that is of particular interest is the Markov property. What we would like, ideally, is a state signal that summarizes past sensations compactly, yet in such a way that all relevant information is retained. A state signal that succeeds in retaining all relevant information is said to have the *Markov property*. Mathematically speaking, the dynamics of how the env responds at time *t + 1* to an action taken at *t* will depend on what had happened so far:

$$P_r\left\{R_{t+1} = r, S_{t+1} = s' | S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}, R_t, S_t, A_t\right\} \tag{16}$$

If the state signal has the Markov property then the environment's response at *t + 1* depends only on the state and action representations at *t*, thus the environment's dynamics (one-step dynamics) will be defined as:

$$p(s', r|s, a) = P_r\{S_{t+1} = s', R_{t+1} = r|S_t, A_t\} \tag{17}$$

A state signal has the Markov property if and only if Eq. 16 is equal to Eq. 17. The latter allows us to predict the next state and expected next reward given the current state and action (Sutton and Barto (2018)).

### 1.2.4 Markov Decision Process

A task that satisfies Markov property is called *Markov decision process*. If the state and action spaces are finite, then is called *finite Markov decision process (finite MPD)* and are defined by their action and state spaces and the one-step dynamics of the env (Eq. 17). From the one-step dynamics one can easily derive the expected rewards for state–action pairs $r(s,a)$, the state-transition probabilities $p(s'|s,a)$ and the expected rewards for state–action–next-state triples $r(s,a,s')$

### 1.2.5 Value Functions

A value function estimates the return for the agent to be in a given state (to perform a given action in a given state). The rewards the agent can expect to receive in the future depend on what actions it will take. Accordingly, value functions are defined with respect to particular policies.
A policy is a mapping from each state $s \in S$ and each action $a \in A$, to the probability $\pi(a|s)$ f taking action a when in state s. The value of a state $s$ under a policy $\pi$, denoted $v_\pi(s)$, is the expected return when starting in $s$ and following $\pi$ thereafter. $q_\pi(s,a)$ is the expected return starting from $s$, taking the action $a$ , and thereafter following policy $\pi$.
The *Bellman equation for* $v_\pi(s)$ expresses a relationship between the value of a state and the values of its successor states. The Bellman equation averages over all the possibilities, weighting each by its probability of occurring.

### 1.2.6 Optimal Value Functions

Roughly speaking, solving a reinforcement learning problem consists in finding a policy that achieves a lot of reward over the long time run. The *optimal value functions* assign to each state, or state–action pair, the largest expected return achievable by any policy. A policy $\pi$ is optimal ($\pi_*$) if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in S$ and $\pi' \in \Pi$. The *Bellman optimality equations* are special consistency condition that the optimal value functions must satisfy and that can, in principle, be solved for the optimal value functions, from which an optimal policy can be determined with relative ease.

## 2 Problem statement and discussion of the instance generation

The objective of this section is to design a non-pharmaceutical interventional reactive strategy to mitigate the diffusion of COVID-19. The strategy is based on performing massive testing in order to detect positive individuals in the network. Note that the tests are a limited resource, so we seek to maximize our goal of disease contention while minimizing the total use of COVID tests. We define these strategies as reactive because they will depend on the status of the environment.

The environment is set to be the SEPIA diffusion model stated in Part IV and the agent decides which action takes based on the state of the network. However, we can not predict the behaviour of this environment, we cannot predict how many infected cases will be reported in the next iteration neither whether our strategy will be effective. The Reinforcement Learning technique is the most suitable to the problem stated above: define a policy to take an action based on the state of an uncertain environment to maximize a reward.

Each simulation step is analogue to a day and the environment calculates the transmission dynamic of each of the nodes within the network. The agent decides an action to take and after all the dynamics of a day are calculated receives an observation and a reward.

As said, we seek to define a strategy to mitigate the diffusion of the COVID-19 while minimizing the number of tests performed at each time step. However, we will first propose a RL interface where the number of tests are fixed at each time step to test the effectiveness of different RL algorithms.

Reinforcement learning tasks can be divided into two type of tasks: first one, a task in which the agent-environment interaction naturally breaks down into a sequence of separate episodes (episodic tasks) or one in which it does not (continuing tasks). We then define the Task described in this report as an episodic task, where an episode is defined as a limited bunch of steps that ends (terminal step) when the agent identifies for three times in a row no infected nodes, inferring that the pandemic has been contained.

### 2.1 Visualization tools

From now on we will use in this document two tools to visualize the results of a simulation and will help us to understand and analyze what happened in each simulation. Both tools are a self-developed inherited class from `DiffusionTrend` class in `NDLIB` software package that uses `Bokeh`, a `Python` data visualization library that provides `html` graphics.

A A first graph (example in Figure 28) including the evolution of the people infectious (Presymptomatic, Symptomatic and Asymptomatic statuses) in an stacked area chart and two lines that represents the trends of the percentage of people that are Quarantined and the trend of the percentage of people that have died. The code can be found here: Visualization tool 1

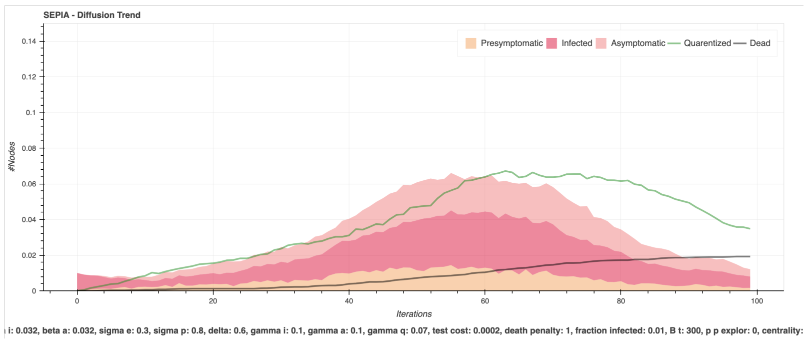


Figure 28: Visualization tool one

B A 2-axis graph (example in Figure 29) including a dashed line representing the evolution of the people infectious (Presymptomatic, Symptomatic and Asymptomatic statuses) compared with the estimation of infectious nodes calculated from the observations that the RL agent perceives from the environment and a stacked chart representing the distribution of tests between exploitation, exploration and pure exploration at each simulation step. The stacked chart uses a different scale than the trends and is represented in the right-hand side of the graph. The code can be found here: Visualization tool 2

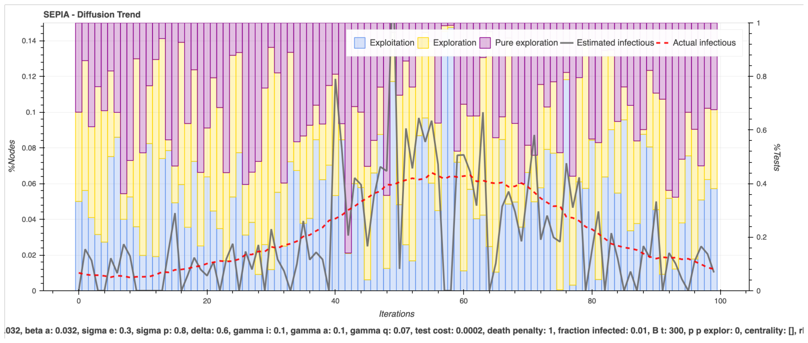


Figure 29: Visualization tool two

## 2.2 Base RL model with fixed number of tests

The action space, i.e. the possible actions that the agent can take, is based on three possible sets of base actions:

- *Test for exploitation*: it is a set of tests done to individuals in the neighborhood of a registered ill node.
- *Test for pure exploration*: it is a set of tests done to randomly chosen people among the network. The goal of these test is to estimate the real number of ill people.
- *Test for exploration*: it is a set of tests done to people in the network related to a particular policy. In this context, we mean by policy a strategy to identify the important nodes within the network, such as we did in Part III with the ten different strategies for Critical Node Detection. This is done to keep under surveillance people that are particularly important for the network.

Let us define then the action space as:

$$A_t = (x_1(t), x_2(t), x_3(t)), \tag{18}$$

where $x_1(t)$ is the number of nodes for exploitation, $x_2(t)$ is the number of nodes for exploration and $x_3(t)$ is the number of nodes for pure exploration at time $t$. This is a analogue reasoning as how Reinforcement Learning algorithms behaves, we know that testing a node in the neighborhood of an already registered ill node is more effective than testing a node with no information about his surrounding. Whereas, to discover this infected nodes the agent must explore before and test random nodes. Another exploratory behaviour but more conservative is to test nodes that the agent knows that are important within the network according to some centrality among those discussed in Part III, for commodity we will use always the degree centrality. It is to be expected that the agent will behave more exploratory at the early steps of the simulation.

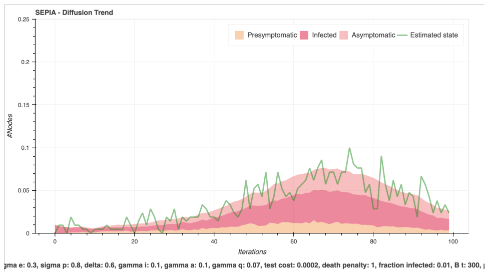
Given a set $A$, we call its cardinality (i.e. the number of element) $|A|$. Let us define:

$$\hat{s}_t = \frac{|\{\text{ill pure exploration test}\}|}{|\{\text{pure exploration test}\}|} \tag{19}$$

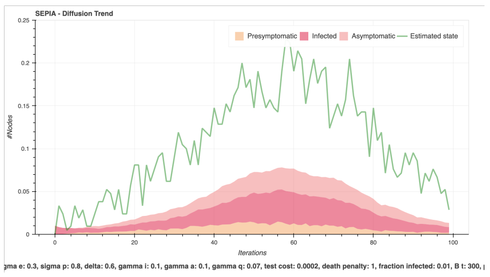
In Eq. (19) we estimate the state of the system in terms of number of ill people. Where we define ill people as the set of people who is either Presymptomatic, Symptomatic or Asymptomatic. We estimate it by using the result of some randomly picked nodes, pure exploratory tests, and not considering the results of tested people when doing exploitation or exploration in order to not overestimate the situation. If we estimate it also considering the tests done to people in the neighborhood of a detected ill person, which is more likely to be infected, the estimation of the state will be biased and we will be overestimating the number of infected people.

To demonstrate the previous statement we are going to run two simulations in parallel on the same network, a) One considering only tests done for pure exploration to estimate the state of the environment and b) another considering all the tests done (exploitation, exploration and pure exploration) when estimating the state. For this simulations we consider a population of *1e4* people where the average degree is 5. We perform 300 COVID tests at each time, 200 of them for pure exploration and the remaining 100 divided for exploitation and for exploration. Now we compare the trends of the estimated number of infected people vs the actual in both scenarios a) and b). To understand how these Figures are generated as well as how the error between the actual and the estimation is calculated refer to Section 2.2.3.

Figures 30a and 30b represent both simulation for both scenarios described a) and b) respectively, by looking at the difference in the trends in each graph (green vs stacked one) we can see how clearly we are overestimating the state of the environment when considering all the tests done vs when considering only pure exploratory ones. Also we can see it by comparing the error computed which is much higher when considering all type tests. Thus, we demonstrate our previous hypothesis and we propose then a novel and accurate way of estimating the state of the environment.



(a) State estimation vs Actual state when considering only pure exploratory tests. MSE: 1.6e-4



(b) State estimation vs Actual state when considering all tests. MSE: 5.8e-3

Figure 30: State estimation: Considering all tests vs only pure exploratory ones

It is interesting to note that there will always be a deviation in the estimation of the state of the environment due to false positives and false negatives but they represent a small percentage of the overall state and it depends on the parameters $\rho_{np}$ and $\rho_{in}$, which are the probability of not being infected but testing Covid positive and the probability of being infected but testing negative.

The state representation is $(t, \hat{s}_t, B_t)$, where $t$ is the time step of the simulation, $s_t$ is defined in Eq. (19) and $B_t$ is the budget available in time step $t$. However, we consider in this section $B_t$ to be fixed at each time $t$. Since the number of tests is fixed the terms in the action vector must satisfy the following equation:

$$x_3 = 1 - x_2 - x_1 \tag{20}$$

Let us finally define

$$r_t = -|\{\text{dead people}\}_t| \tag{21}$$

Where $r_t$ is defined as the reward given by the environment to the agent at simulation time $t$ and is calculated as minus the number of people that have died at simulation time $t$.

### 2.2.1 Buffer management for better allocation of COVID tests

The idea is to test at each simulation step the most important people within the network. For example when doing tests for exploration, we test the *k* people with the higher number of neighbors. However, if we always test the same *k* people, there will be $\{N - k\}$ people that are never tested, where *N* is the total number of people in the network and $k << N$, meaning that the majority of the people will never be tested and the virus will never be contained.

We introduce, as we did in Part III, the concept of node buffer management. As the definition from a data perspective says, a buffer is used for temporary storage of data that is waiting to be sent. If we extrapolate it to our case, the data that is waiting to be sent is the people that will be tested for COVID. You can think of the buffer as a cyclic array: first the nodes are ordered according some policy, then the *k* first nodes will be picked to test and after being tested they will be placed at the end of the queue and so on while running the simulation.

We need to establish a balance between testing the most important nodes while testing all the nodes in the network. We propose the following strategy: concatenate in the buffer array a) the nodes considered most important according to some strategy and b) all the nodes of the network, so that the most important nodes will be tested twice as often as the nodes considered less important. Refer to Figure 4 and algorithm 4 to better understand it.

When performing for exploration, the important nodes are considered as those whose degree (number of neighbours) is greater than the average degree in the network.

**Algorithm 4** Buffer instantiate and management

```
$T \leftarrow$ final iteration of an episode
$k_i \leftarrow$ set of $K_i$ COVID tests done at iteration i
for each iteration i in $[0, T]$ do
    if i $== 0$ then
        for each node v in Network do
            Compute its degree and assign to $D['v']$
        end for
        $\langle d \rangle \leftarrow$ average value of values in D
        $D_i$ contains each node v in $D$ where $D[v] \geq \langle d \rangle$
        Buffer $\leftarrow concatenate(D_i, D)$
    else
        test nodes in $k_i$
        $k_i' \leftarrow k_i$
        delete subset $k_i$ from Buffer
        Buffer $\leftarrow concatenate($Buffer, $k_i')$
    end if
end for
```

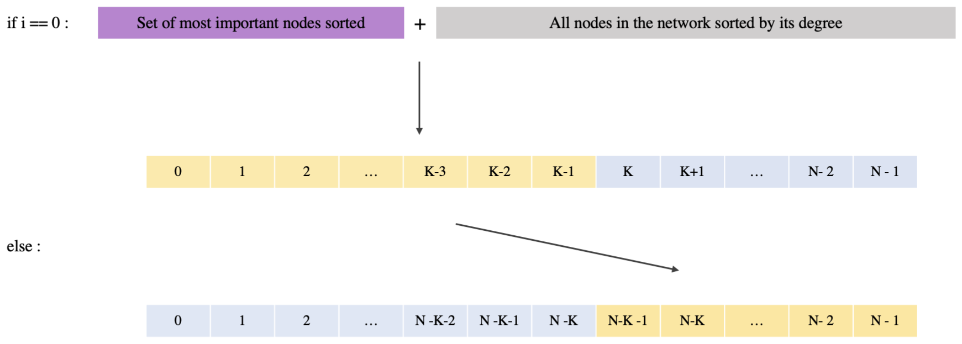


Figure 31: Cyclic buffer management at iteration *i*, *K* is the number of COVID tests done at iteration *i*, *N* is the total number of nodes.

Finally to validate the effectiveness of the novel proposed buffer management method we test it again to a simulation with no buffer and always testing the k-most important nodes in the network. We consider a network of $N$=10000 individuals with an average degree equal to 5. We perform 300 tests at each simulation step and we plot the trends of the simulations in Figure 32, we can conclude that if we use a buffer management technique we are able to mitigate the impact of the COVID pandemic while applying a fairer treatment among the individuals within the network.

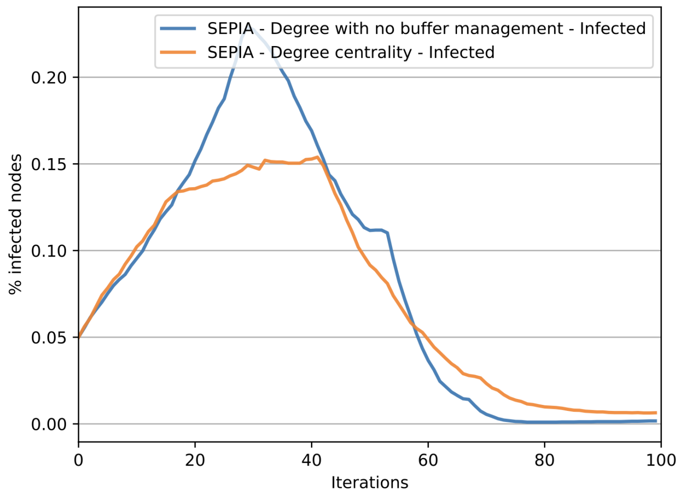


Figure 32: Infected trends when using buffer management vs when always testing the k most important nodes

### 2.2.2 Implementation and results

We implement the Reinforcement Learning interface in python, first we set up the SEPIA `DiffusionModel` class implemented in previous the Part to be an `OpenAI gym` environment. For the agent we use also the python library `OpenAI gym`, a toolkit for developing and comparing reinforcement learning algorithms that provides a standard interface to communicate between learning algorithms and environments.

The action space is a python list of length equal to three, first position of the list, $x_1$ describes the percentage of the total number of tests $B_t$ devoted for exploitation, second position $x_2$ are the percentage of tests for exploration and $x_3$ is the tests for pure exploration. Each of the base actions can take values from range $[0, 1]$, but they will be post normalized to fulfill Eq. (20). As the state is only given by the tests done for pure exploration, that is, nodes selected randomly, we also set the condition to the action space to $x_3 > 0.1B_t$ to be able to have information about the state.

We will test two different Reinforcement learning algorithms:

- `A2C` from `Stable baselines 3` library, is an algorithm for deep reinforcement learning that uses asynchronous gradient descent for optimization of deep neural network. Main reasons for choosing this algorithm is the ability to deal with continuous action spaces, as well as its ability to train feed-forward. It is an on-policy policy search method and uses using $\epsilon$-greedy exploration.
- `PPO` from `Stable baselines 3` library, is a policy gradient method for reinforcement learning that alternates between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent. Also selected for the ability to work with continuous spaces.

**Reinforcement Learning model performance**

To prove the effectiveness of the RL algorithm we test it against three simulations taking fixed basic actions:

A Always testing random people $A = (0, 0, 1)$

B Always doing exploratory tests, i.e., $A = (0, 1, 0)$, testing the k-most important nodes according a centrality

C Always doing exploitation tests, i.e., $A = (x_1, x_2, x_3) = (1, 0, 0)$, testing the neighbors of the people already tested positive

First we compare the performance between scenarios A (Figures 33a1 and 33a2), B (Figures 33b1 and 33b2) and C (Figures 33c1 and 33c2). We see in Figures 33b2 and 33c2 that even doing always tests for exploration or exploitation, there are a number of tests devoted for pure exploration, those represented in purple, since they are needed to estimate the state of the environment. Performing always exploiting, that is, testing people in the neighbourhood of already COVID+ tested, leads to the highest reward, that is, the minimum number of deaths. However, starting performing pure exploitation is meaningless as there is no data about detected COVID positives, which is the basis of the performance of contact tracing algorithm. In fact if we look at Figure 33c1, the number of detected COVID positive people (quarantined trend) up to iteration number 10 is negligible, and thus it needs to be combined with whether exploration or pure exploration one. This is an evidence on the need of build an interaction agent-environment.

We set up the Reinforcement Learning agent as the above explained A2C RL algorithm and let the model train by interacting with the environment a total number of iterations equal to *8e4*. The whole agent training process is all about getting to the highest expected return possible, in this case, to minimize the total number of deaths. If this metric goes up throughout the training, means that the agent is learning well.

With the defined goal in this problem is difficult to define what return to expect more than minimize the number of died people to determine what is a good score. In order to evaluate the results obtained we compare a simulation of the trained policy with a Random agent baseline, that is an agent that always takes random actions, and that the rewards in the learning curve goes up.

Figures 34b1 and 34b2 show the trends of the simulation of an episode and the distribution of COVID tests after training the agent. To validate the trained policy we compare it to the Random agent baseline (Figures 34a1 and 34a2).

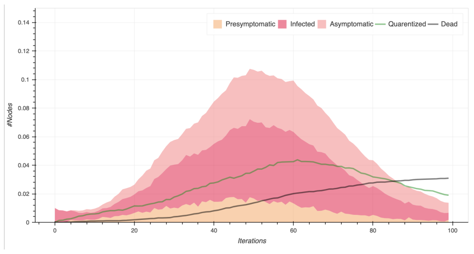
First evidence is that the total reward is maximized, where the total reward is:

$$R_{A2C} = -148 \text{ deaths} > R_{RAB} = -178 \text{ deaths} \tag{22}$$

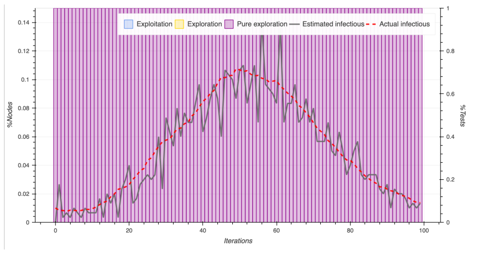
being $R_{A2C}$ the total reward of the trained RL agent and $R_{RAB}$ the total reward of the Random agent baseline, proving the effectiveness of the training RL agent. We can also observe how the infectious trends are decreased when using the trained agent, meaning that when using the trained agent, the impact of the COVID pandemic is mitigated.

Finally, if we look at Figure 35 we see how the mean reward increases as the agent interacts with the environment. In the early iterations, the mean reward falls below 200 died people and after 5e4 iterations, it establishes around 140 to 150 deaths in a single episode. Thus, we prove our hypothesis of the agent being able to learn by interacting with the environment and maximazing the total reward.

Now we take a closer look at Figure 34b2 to see how the tests are distributed at each simulation step. In the first iterations (up to iteration #20) the tests are distributed between testing for pure exploration and for exploration. However, from iteration 20 onwards, the COVID tests are done for pure exploitation since is the strategy that standalone leads to the highest reward, as we demonstrated in Figure 33 and it performs less pure exploratory. Moreover, as we expected, at the early steps of the episode, the agent behave less exploiting since with no data on already tested positive it is meaningless to perform pure exploiting.

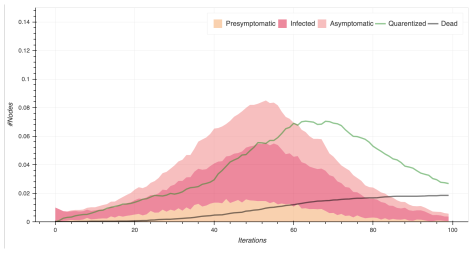


(a1) Diffusion trends when performing always for pure exploration. Total deaths: 309

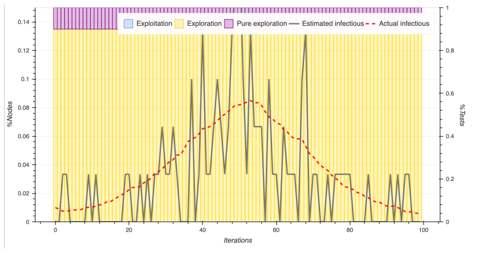


(a2) Distribution of tests and actual infectious trend vs estimated infectious trend

(a) Always doing tests for pure exploration

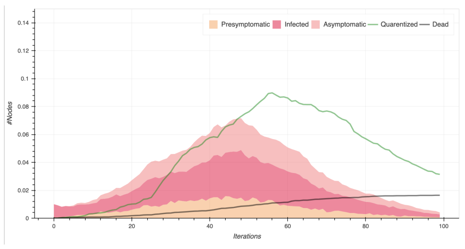


(b1) Diffusion trends when performing always for exploration. Total deaths: 185

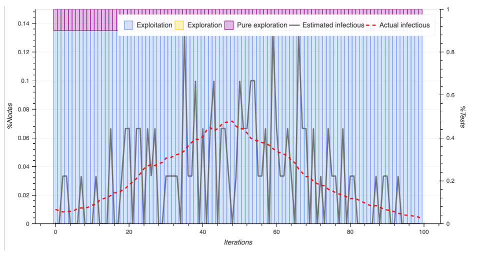


(b2) Distribution of tests and actual infectious trend vs estimated infectious trend

(b) Always doing tests for exploration



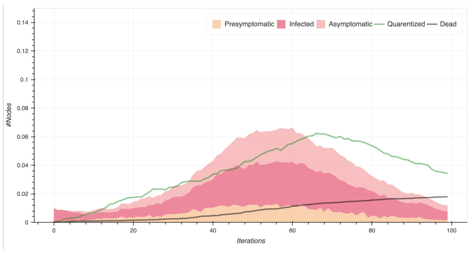
(c1) Diffusion trends when performing always for exploitation. Total deaths: 164



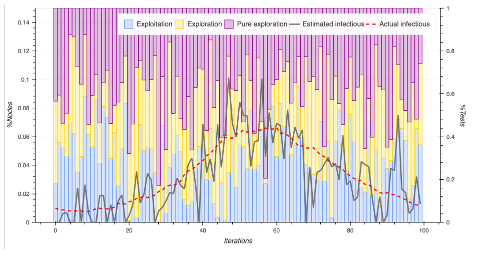
(c2) Distribution of tests and actual infectious trend vs estimated infectious trend

(c) Always doing tests for exploitation

Figure 33: Comparison of the trends when the agent is not trained. Number of people in the network is 1e4 and number of COVID tests done at each iteration is 300

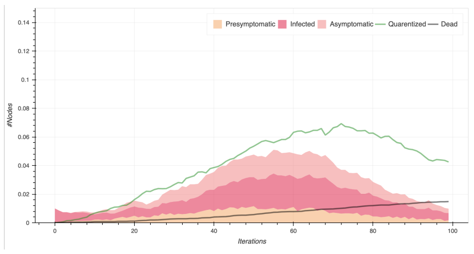


(a1) Diffusion trends when the agent takes random actions. Total deaths: 178

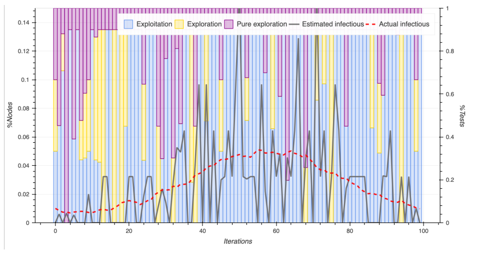


(a2) Distribution of tests and actual infectious trend vs estimated infectious trend

(a) Random agent baseline



(b1) Diffusion trends of the the trained policy. Total deaths: 148



(b2) Distribution of tests and actual infectious trend vs estimated infectious trend

(b) Trained agent

Figure 34: Trained agent performance vs Random agent baseline. Number of people in the network is 1e4 and number of COVID tests at each iteration is 300

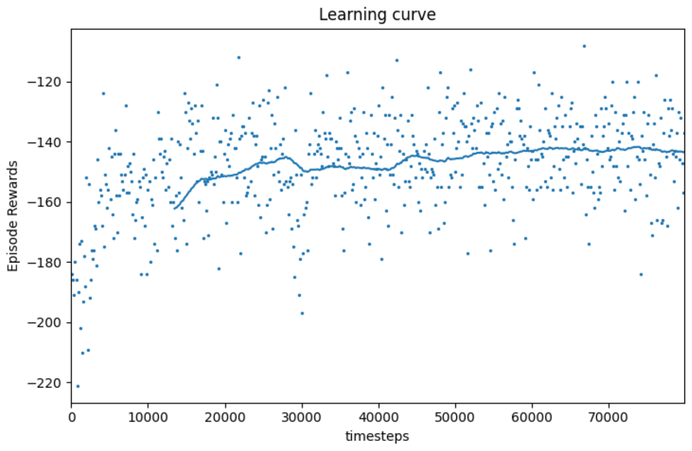


Figure 35: Learning curve of the A2C RL agent

### 2.2.3 State estimation

We have seen that the agent devotes the most of the COVID tests to exploit and few of them to pure exploration. This leads to the highest total reward. However, this has a drawback as the estimation of the state of the network is poor due to not performing enough pure exploratory tests. If we take a closer look at the solid gray line in Figure 34b2, the observation set the agent receives, we can see that there are several steps where the estimation of the state of the network is 0, meaning that there is no nodes infected, which is actually not right if we compare them to the dashed red line in Figure 34b2, that represent the actual number of infectious people. Then we are underestimating and overestimating the state of the network.

Let's try to prove the hypothesis that the more tests for pure exploration the better the estimation of the state. To better illustrate this, for the same network topology we will run several simulations in parallel by fixing the number of COVID tests devoted for pure exploration and increasing them in each simulation. Then we will plot the estimation done by the algorithm with the actual state of the environment.

The nodes that can be detected in the network, that is, what we consider the actual state of the network are the aggregated number of nodes which statuses are "Pre-symptomatic" or "Infected" or "Asymptomatic"

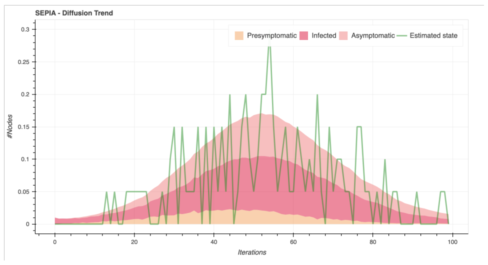
Lets assume that we are only doing tests for pure exploration in order to not unbias our estimation. Thus, the action taken at each step will be $a = [0, 0, 1]$. We also calculate the Mean Square Error (MSE) between the actual value and the estimation done by the algorithm, defined as:

$$MSE = \frac{1}{N}\sum_{t=1}^{T}(s_t - \hat{s}_t)^2 \tag{23}$$

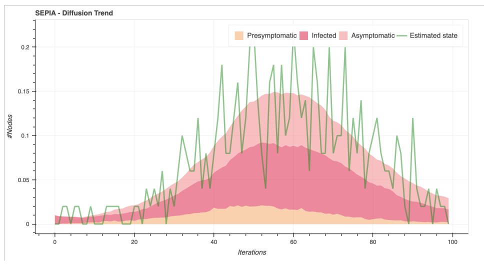
If we take a look at Figure 36 and 37 we can see how the estimation is much better as long as the number of COVID tests for pure exploration performed increases. However, devoting that much number of tests to pure exploration will lower the overall performance of the RL algorithm.

To better estimate the state, the Partially observable Markov decision processes (POMDPs) methods allow for the modeling of problems that have hidden state. Hasinoff (2002) introduces the use of memory to maintain an internal state. A good idea is to introduce some form of memory, so that the agent can attempt to use its past experiences to disambiguate aliased states and act appropriately.

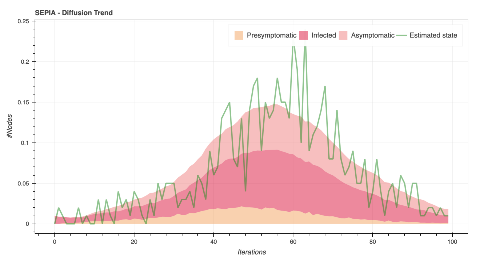
Another option will be to extend the state space creating a dictionary of observations, that is, include more information of the state of the network to the observation such as the cumulative value of positive detected nodes or the number of individuals under quarantine, which are variables known describing the state of



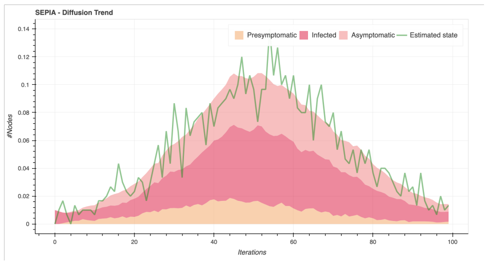
(a) Number of tests K= 30, MSE = 3.04e-3



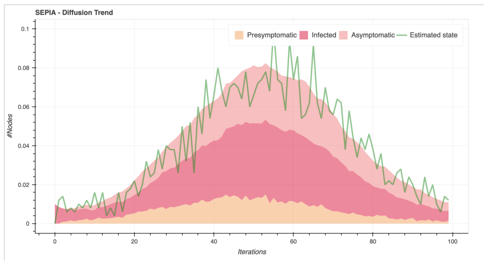
(b) Number of tests K= 50, MSE = 1.28e-3



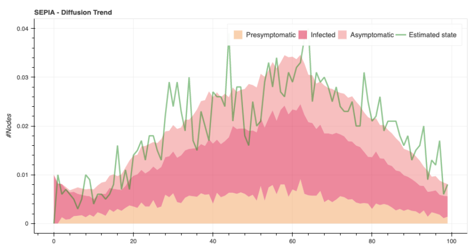
(c) Number of tests K= 100, MSE = 8.98e-4



(d) Number of tests K= 300, MSE = 1.43e-4



(e) Number of tests K= 500, MSE = 8.57e-5



(f) Number of tests K= 1000, MSE = 2.24e-5

Figure 36: State estimation for different number of tests for pure exploration in a network with 10000 nodes

the environment.

From now on in this document for simplicity we chose to implement a rule to perform a fixed number of pure exploratory tests at each simulation step to estimate the state of the network. The number of tests for pure exploratory have to make a trade off between fair estimation of the network while not losing performance on the contention strategy of the virus as they are randomly allocated. If we look at Figure 36, sub-figure 36c seems to be the fairest estimation with the lowest number of tests. Thus we will devote $\frac{\#\text{tests for pure exploration}}{\#\text{Nodes in the network}} = \frac{100}{10000} = 1\%$ of the nodes will be devoted for pure exploration.

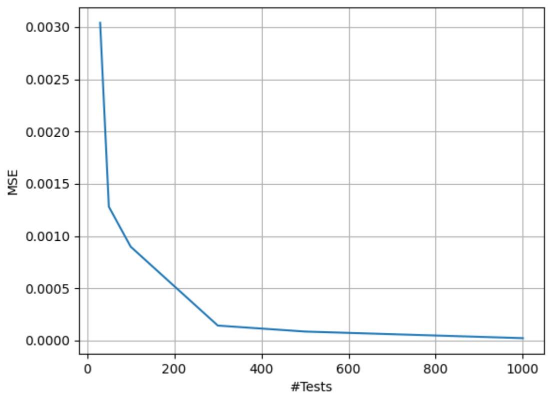


Figure 37: Number of tests vs MSE

## 2.3 Optimizing the number of tests using a RL model

### 2.3.1 Limited number of tests per step

The objective of this section is to design a reactive non-pharmaceutical intervention strategy to contain the diffusion of the disease with the caveat of limited resources. Thus we will try to optimize the use of the COVID tests by adapting it to the state of the environment.

Let us define:

$$A_t = (x_1(t), x_2(t), x_3(t), B_t), \tag{24}$$

as the action space, where $x_1(t), x_2(t)$ and$x_3(t)$ are defined as in the previous section. We now introduce a fourth base action to the Action vector, $B_t$, defined as the number of COVID tests that the agent performs at step $t$ and it varies from iteration to iteration, we also limit the maximum number of COVID tests that the agent can perform at each time step to $B$.

We also define:

$$x_1(t) + x_2(t) + x_3(t) = B_t \leq B, \tag{25}$$

as the total number of tests done at time step $t$.

We have demonstrated in the previous section the need to perform a minimum number of pure exploratory tests in order to let the RL agent estimate properly the state of the environment. We also have discussed and demonstrated the low performance of doing pure exploration when aiming to contain the spread of the pandemic. Keeping in mind that COVID tests are scarce and costly we will devote the minimum number of pure exploratory tests that allows us to make a fair estimation of the environment. We then set $x_3$ constant over time and equal to a $1\%$ of the nodes in the network. As we simulate a network with N, the number of nodes, equal to 10000, the number of tests for pure exploration is 100.

Then the agent only need to take two actions depending on the state perceived:

- $B_t$: The number of COVID tests to do at each simulation step or iteration, ranging from 100 to B
- From those tests, how many are for exploitation and how many for exploration

Let us define

$$A(s) = (a[0], a[1]), \tag{26}$$

as the set of actions possible in state s.

Finally, we have:

$$\begin{gathered} A_t = (x_1(t), x_2(t), x_3(t), B_t), \\ x_1(t) = (1 - a[0])a[1]B \\ x_2(t) = a[0]a[1]B \\ x_3(t) = 100 \\ B_t = a[1]B \end{gathered} \tag{27}$$

The environment and the observation are defined as in the previous section.

Each unitary test has a cost c. It is worthwhile noting that the test is characterized by a confusion matrix:

$$\begin{bmatrix} \rho_{ip} & \rho_{np} \\ \rho_{in} & \rho_{nn} \end{bmatrix} \tag{28}$$

Let us define:

$$R_t = -(cK_{t-1} + pD_{t-1}) \tag{29}$$

as the reward at time step $t$ dependent on $A_{t-1}$ and $S_{t-1}$. The reward function is composed by two terms the cost of doing a unitary test, $c$, multiplied by the number of tests done at time t-1, $K_{t-1}$, and the number of victims, $D_{t-1}$, weighted by the penalty rate $p$ of an individual dying. The cost of the tests and the penalty rate are adjusted and fixed from experience by doing several simulations.

The goal of a classic Reinforcement learning problem is to maximize the episodic return defined as the sum of the rewards:

$$R = R_0 + R_1 + .. + R_T \tag{30}$$

### 2.3.2 Implementation and results

The code of the environment can be found here: RL environment: Optimization of tests

Now we implement our environment adjusting the action space function and creating the reward function. We now test the implemented Reinforcement Learning agent-environment with the above described POP RL algorithm. We train the agent during 2e5 steps to estimate the optimum policy $\pi$.

We validate the trained Reinforcement Learning agent in two different ways, as described above, a) by comparing it to a Random Agent Baseline and b) by evaluating the learning curve of the training process.

Figure 38 represent the evolution of the episodic rewards. The total mean reward, defined as the sum of the individual rewards in an episode defined in Eq. 29, went from -400 to achieve a reward below -335. We can also see how the mean reward goes up. This clearly demonstrate how the RL agent learns from interacting with the environment.

On Figure 39b we can see plotted a comparison on the distribution of the tests during an episode for both, the Random Agent Baseline and the trained POP RL agent. The reward obtained was higher in the trained RL algorithm, being

$$R_{POP} = -332.91 \text{ deaths} > R_{RAB} = -387.45 \text{ deaths} \tag{31}$$

This proves the effectiveness of the Reinforcement Learning and how it optimizes the use of the COVID tests while containing the spread of the virus.

Now we take a closer look to the distribution of the tests on an episode after the agent was trained, depicted in Figure 39b. We are going to ignore the pure exploration tests (those in purple) as they are fixed at each time step, we will focus only in exploiting and exploratory ones.

We can see that in the early iterations of the episode when the virus is not spread yet and the agent perceives that the state of the environment is below the 2%, the agent test at a lower rate, indeed the agent only performs COVID tests on 3 out of the first 15 iterations and only in 1 of them deciding to do more than the 50% of the available tests.

Then from iteration 15 to 70 where the agent perceives that more than the 3% of the population is infected, the agent decides to do COVID tests on 33 out of the 55 subsequent iterations. Deciding in 26 out of the 33 iterations to spend more than the 50% of the available tests per iteration. Finally, in the last 20 iterations, the agent perceives that the state of the environment is 0% in almost every iteration meaning that there is no infected individuals within the network, the agent decides to test only in 5 out of the 20 iterations.

We can also observe how the agent decides to do almost always tests for exploitation, that is, to test the neighborhood of already positive tested ones as we have demonstrated during the document that are more effective. Always that there is a spike of infected people, the agent has the ability mitigate it and revert the trend, we can see it at iterations 16, 28, 39 and so on

We have reached our goal of designing a strategy that optimizes the allocation of the COVID tests.

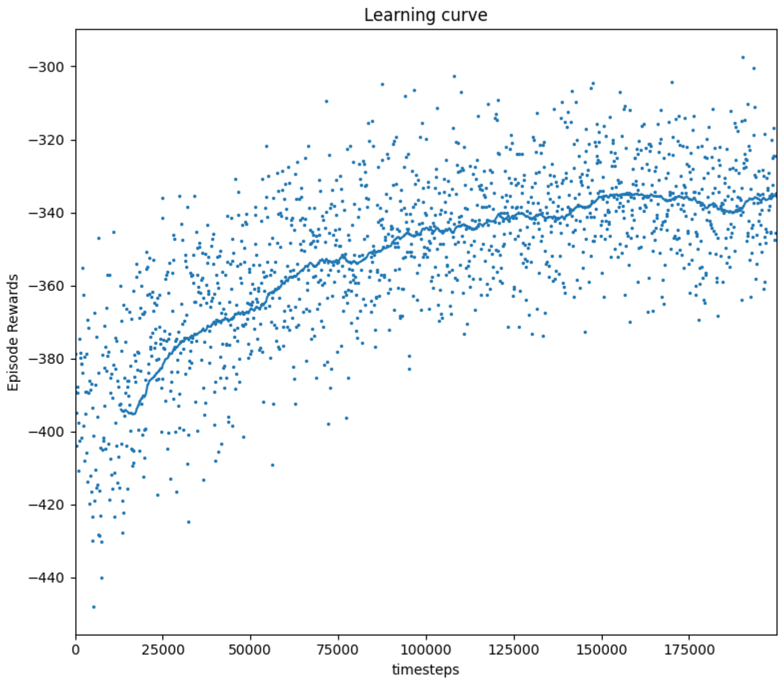
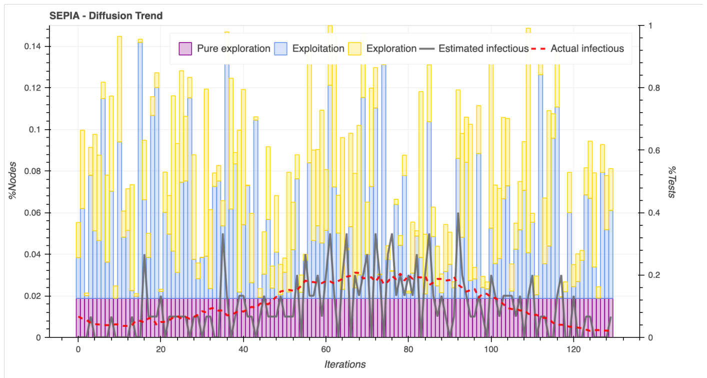
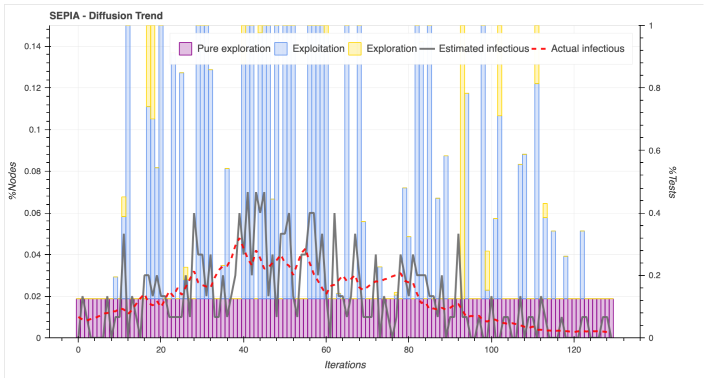


Figure 38: Learning curve of the POP RL algorithm



ıma p: 0.8, delta: 0.6, gamma i: 0.1, gamma a: 0.1, gamma q: 0.07, test cost: 0.005, death penalty: 1, detected reward: 0, fraction infected: 0.01, B t:

(a) Tests distribution of the Random Agent Baseline. Reward: -387.45



ıma p: 0.8, delta: 0.6, gamma i: 0.1, gamma a: 0.1, gamma q: 0.07, test cost: 0.005, death penalty: 1, detected reward: 0, fraction infected: 0.01, B t:

(b) Tests distribution of the trained RL POP agent during 1e5 iterations. Reward: -332.91

Figure 39: Validation of the RL agent to optimize the number of tests done

# Bibliography

M. Alene, L. Yismaw, M. A. Assemie, D. B. Ketema, W. Gietaneh, and T. Y. Birhan. Serial interval and incubation period of covid-19: a systematic review and meta-analysis. *BMC infectious diseases*, page 257, 2021. doi: 10.1186/s12879-021-05950-x.

S. T. Ali, L. Wang, E. H. Y. Lau, X.-K. Xu, Z. Du, Y. Wu, G. M. Leung, and B. J. Cowling. Serial interval of sars-cov-2 was shortened over time by nonpharmaceutical interventions. *Science*, 369(6507):1106–1109, 2020. doi: 10.1126/science.abc9004. URL `https://www.science.org/doi/abs/10.1126/science.abc9004`.

A.-L. Barabási. *Network science.* Cambridge University Press,, 2016.

A. Beigi, A. Yousefpour, A. Yasami, J. F. Gomez-Aguilar, S. Bekiros, and H. Jahanshahi. Application of reinforcement learning for effective vaccination strategies of coronavirus disease 2019 (COVID-19). *Eur Phys J Plus*, 136(5):609, May 2021.

U. Bharti, D. Bajaj, H. Batra, S. Lalit, S. Lalit, and A. Gangwani. Medbot: Conversational artificial intelligence powered chatbot for delivering tele-health after covid-19. *5th International Conference on Communication and Electronics Systems*, pages 870–875, 2020.

P. Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2:113–120, 1972.

B. Buonomo and R. D. Marca. Effects of information-induced behavioural changes during the covid-19 lockdowns: the case of italy. *Royal Society Open Science*, 7: 201635, 2020. doi: https://dx.doi.org/10.1098/rsos.201635.

R. S. Burt. Social contagion and innovation: Cohesion versus structural equivalence. *American Journal of Sociology*, 92:1287 – 1335, 1987.

Y. Carson and A. Maria. Simulation optimization: Methods and applications. *Proceedings - Winter Simulation Conference*, page 118–126, 1997. doi: https://10.1145/268437.268460.

D.-B. Chen, H. Gao, and L. Lü. Identifying influential nodes in large-scale directed networks: The role of clustering. *PLoS ONE*, 2013. doi: https://doi.org/10.1371/journal.pone.0077455.

J. Chen, K. Li, Z. Zhang, K. Li, and P. S. Yu. A survey on applications of artificial intelligence in fighting against covid-19. *arXiv e-prints*, 54(8), oct 2021. ISSN 0360-0300. doi: 10.1145/3465398. URL `https://doi.org/10.1145/3465398`.

P. Chen, H. Xie, S. Maslov, and S. Redner. Finding scientific gems with google's pagerank algorithm. *Journal of Informetrics*, 1(1):8–15, 2007. ISSN 1751-1577. doi: https://doi.org/10.1016/j.joi.2006.06.001. URL `https://www.sciencedirect.com/science/article/pii/S1751157706000034`.

W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. *Association for Computing Machinery*, 199-208:199–208, 01 2009. doi: 10.1145/1557019.1557047.

Y. Chen, A. H. Wang, B. Yi, K. Q. Ding, H. B. Wang, J. M. Wang, H. B. Shi, S. J. Wang, and G. Z. Xu. [epidemiological characteristics of infection in COVID-19 close contacts in ningbo city]. *Zhonghua Liu Xing Bing Xue Za Zhi*, 41(5):667–671, May 2020.

E. Council. Timeline - council actions on covid-19, 2022. URL `https://www.consilium.europa.eu/es/policies/coronavirus/timeline/`.

B. dictionary. The incubation period, 2022. URL `https://www.biologyonline.com/dictionary/incubation-period`.

L. D. Domenico, G. Pullano, C. E. Sabbatini, P.-Y. Boëlle, and V. Colizza. Impact of lockdown on covid-19 epidemic in Île-de-france and possible exit strategies. *BMC Medicine*, 18:240, 2020. doi: https://doi.org/10.1186/s12916-020-01698-4.

S. Flaxman, S. Mishra, A. Gandy, H. J. T. Unwin, T. A. Mellan, H. Coupland, C. Whittaker, H. Zhu, T. Berah, J. W. Eaton, M. Monod, I. C. C.-. R. Team, A. C. Ghani, C. A. Donnelly, S. Riley, M. A. C. Vollmer, N. M. Ferguson, L. C. Okell, and S. Bhatt. Estimating the effects of non-pharmaceutical interventions on covid-19 in europe. *Nature*, 584:257–261, 2020. doi: https://doi.org/10.1038/s41586-020-2405-7.

L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977. ISSN 00380431. URL `http://www.jstor.org/stable/3033543`.

M. Gatto, E. Bertuzzo, L. Mari, S. Miccoli, L. Carraro, R. Casagrandi, and A. Rinaldo. Spread and dynamics of the covid-19 epidemic in italy: Effects of emergency containment measures. *Proc. Natl Acad. Sci. USA*, 117:10484–10491, 2020. doi: https://doi.org/10.1073/pnas.2004978117.

G. Giordano, F. Blanchini, R. Bruno, P. Colaneri, A. D. Filippo, A. D. Matteo, and M. Colaneri. Modelling the covid-19 epidemic and implementation of population-wide interventions in italy. *Nature medicine*, 26:855–860, 2020. doi: https://doi.org/10.1038/s41591-020-0883-7.

P. Hage and F. Harary. Eccentricity and centrality in networks. *Social Networks*, 17:57–63, 1995.

M. Haines. Frustration grows amid us covid test shortages, 2022. URL `voanews.com/a/frustration-grows-amid-us-covid-test-shortages-/6393440.html`.

S. W. Hasinoff. Reinforcement learning for problems with hidden state. Technical report, Standford University, 2002.

X. He, E. H. Y. Lau, P. Wu, X. Deng, J. Wang, X. Hao, Y. C. Lau, J. Y. Wong, Y. Guan, X. Tan, X. Mo, Y. Chen, B. Liao, W. Chen, F. Hu, Q. Zhang, M. Zhong, Y. Wu, L. Zhao, F. Zhang, B. J. Cowling, F. Li, and G. M. Leung. Temporal dynamics in viral shedding and transmissibility of covid-19. *Nature Medicine*, 26(5):672–675, May 2020. ISSN 1546-170X. doi: 10.1038/s41591-020-0869-5. URL `https://doi.org/10.1038/s41591-020-0869-5`.

L. Hilfiker and J. Josi. Epyestim python libray, 2022. URL `https://github.com/lo-hfk/epyestim`.

Y. Hu, S. Ji, Y. Jin, L. Feng, H. E. Stanley, and S. Havlin. Local structure can identify and quantify influential global spreaders in large scale social networks. *Proceedings of the National Academy of Sciences*, 115(29):7468–7472, 2018. doi: 10.1073/pnas.1710547115. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1710547115`.

D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 137-146, 07 2003. doi: 10.1145/956750.956769.

D. Kempe, J. Kleinberg, and E. Tardos. Influential nodes in a diffusion model for social networks. *Lecture Notes in Computer Science*, 3580:1127–1138, 07 2005. doi: 10.1007/11523468_91.

W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Royal Society*, 115, 1927. doi: https://doi.org/10.1098/rspa.1927.0118.

M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse. Identification of influential spreaders in complex networks. *Nature Physics*, 6:888–893, 2010. doi: https://doi.org/10.1038/nphys1746.

R. L. Kumar, F. Khan, S. Din, S. S. Band, A. Mosavi, and E. Ibeke. Recurrent neural network and reinforcement learning model for COVID-19 prediction. *Front Public Health*, 9:744100, Oct. 2021.

E. Lavezzo, E. Franchin, C. Ciavarella, G. Cuomo-Dannenburg, L. Barzon, C. Del Vecchio, L. Rossi, R. Manganelli, A. Loregian, N. Navarin, D. Abate, M. Sciro, S. Merigliano, E. De Canale, M. C. Vanuzzo, V. Besutti, F. Saluzzo, F. Onelia, M. Pacenti, S. G. Parisi, G. Carretta, D. Donato, L. Flor, S. Cocchio, G. Masi, A. Sperduti, L. Cattarino, R. Salvador, M. Nicoletti, F. Caldart, G. Castelli, E. Nieddu, B. Labella, L. Fava, M. Drigo, K. A. M. Gaythorpe, A. R. Brazzale, S. Toppo, M. Trevisan, V. Baldo, C. A. Donnelly, N. M. Ferguson, I. Dorigatti, A. Crisanti, K. E. C. Ainslie, M. Baguelin, S. Bhatt, A. Boonyasiri, O. Boyd, H. L. Coupland, Z. Cucunubá, B. A. Djafaara, S. L. van Elsland, R. FitzJohn, S. Flaxman, W. D. Green, T. Hallett, A. Hamlet, D. Haw, N. Imai, B. Jeffrey, E. Knock, D. J. Laydon, T. Mellan, S. Mishra, G. Nedjati-Gilani, P. Nouvellet, L. C. Okell, K. V. Parag, S. Riley, H. A. Thompson, H. J. T. Unwin, R. Verity, M. A. C. Vollmer, P. G. T. Walker, C. E. Walters, H. Wang, Y. Wang, O. J. Watson, C. Whittaker, L. K. Whittles, X. Xi, and I. C. C.-. R. Team. Suppression of a sars-cov-2 outbreak in the italian municipality of vo'. *Nature*, 584 (7821):425–429, Aug 2020. doi: 10.1038/s41586-020-2488-1. URL `https://doi.org/10.1038/s41586-020-2488-1`.

N. H. L. Leung, D. K. W. Chu, E. Y. C. Shiu, K.-H. Chan, J. J. McDevitt, B. J. P. Hau, H.-L. Yen, Y. Li, D. K. M. Ip, J. S. M. Peiris, W.-H. Seto, G. M. Leung, D. K. Milton, and B. J. Cowling. Respiratory virus shedding in exhaled breath and efficacy of face masks. *Nature Medicine*, 26(5):676–680, May 2020. ISSN 1546-170X. doi: 10.1038/s41591-020-0843-2. URL `https://doi.org/10.1038/s41591-020-0843-2`.

Q. Li, T. Zhou, L. Lü, and D. Chen. Identifying influential spreaders by weighted LeaderRank. *Physica A: Statistical Mechanics and its Applications*, 404:47–55, jun 2014. doi: 10.1016/j.physa.2014.02.041. URL `https://doi.org/10.1016/Fj.physa.2014.02.041`.

R. Li, S. Pei, B. Chen, Y. Song, T. Zhang, W. Yang, and J. Shaman. Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (sars-cov-2). *Science*, 368:489–493, 2020. doi: https://10.1126/science.abb3221.

L. Lü, T. Zhou, Q.-M. Zhang, and H. E. Stanley. The h-index of a network node and its relation to degree and coreness. *Nature Communications*, 2016. doi: https://doi.org/10.1038/ncomms10168.

E. Mahase. Covid-19: death rate is 0.66% and increases with age, study estimates. *BMJ*, 369, 2020. doi: 10.1136/bmj.m1327. URL `https://www.bmj.com/content/369/bmj.m1327`.

A. Q. Ohi, M. F. Mridha, M. M. Mostafa, and M. A. Hamid. Exploring optimal control of epidemic spread using reinforcement learning. *Sci Rep*, 10(1):22106, Dec. 2020.

W. H. Organization. Coronavirus disease (covid-19): How is it transmitted?, 2022. URL `https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/`.

R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev.*, 86, 2001. ISSN 14. doi: https://doi.org/10.1103/PhysRevLett.86.3200.

J. Peto. Covid-19 mass testing facilities could end the epidemic rapidly. *BMJ*, 368, 2020. doi: https://doi.org/10.1136/bmj.m1163.

R. Preidt. When is a person with covid-19 most infectious?, 2021. URL `https://www.webmd.com/lung/news/20210827/when-is-a-person-with-covid-19-most-infectious#1`.

T. Qiao, W. Shan, and C. Zhou. How to identify the most powerful node in complex networks? a novel entropy centrality approach. *Entropy*, 19(11), 2017. ISSN 1099-4300. doi: 10.3390/e19110614. URL `https://www.mdpi.com/1099-4300/19/11/614`.

H. Ritchie, E. Mathieu, L. Rodés-Guirao, C. Appel, C. Giattino, E. Ortiz-Ospina, J. Hasell, D. B. Bobbie Macdonald, and M. Roser. Coronavirus pandemic (covid-19). *Our World in Data*, 2020. URL `https://ourworldindata.org/coronavirus`.

G. Rossetti. Ndlib python library, 2022. URL `https://ndlib.readthedocs.io/en/latest/index.html`.

G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31:581–603, 1966. doi: https://doi.org/10.1007/BF02289527.

A. Sandford. Coronavirus: Half of humanity on lockdown in 90 countries. *Euronews*, 2021. https://ourworldindata.org/coronavirus.

S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.

M. Supino, A. d'Onofrio, F. Luongo, G. Occhipinti, and A. D. Co. The effects of containment measures in the italian outbreak of covid-19. *medRxiv*, 2020.03.25.20042713, 2020. doi: https://doi.org/10.1101/2020.03.25.20042713.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

P. Szor. Fighting computer virus attacks. *USENIX*, Aug. 2004. URL `https://www.usenix.org/conference/13th-usenix-security-symposium/fighting-computer-virus-attacks`.

B. Tang, X. Wang, Q. Li, N. L. Bragazzi, S. Tang, Y. Xiao, and J. Wu. Estimation of the transmission risk of the 2019-ncov and its implication for public health interventions. *J Clin Med.*, 9(2):462, 2020. doi: https://10.3390/jcm9020462.

K. A. Walsh, S. Spillane, L. Comber, K. Cardwell, P. Harrington, J. Connell, C. Teljeur, N. Broderick, C. F. de Gascun, S. M. Smith, M. Ryan, and M. O'Neill. The duration of infectiousness of individuals infected with sars-cov-2. *Journal of Infection*, 81(6):847–856, 2020. ISSN 0163-4453. doi: https://doi.org/10.1016/j.jinf.2020.10.009. URL `https://www.sciencedirect.com/science/article/pii/S0163445320306514`.

G. Wittenbaum, A. Hubbell, and C. Zuckerman. Mutual enhancement: Toward an understanding of the collective preference for shared information. *Journal of Personality and Social Psychology*, 77:967–978, 1999. doi: 10.1037/0022-3514.77.5.967.

K. Xiao, H. Yang, B. Liu, X. Pang, J. Du, M. Liu, Y. Liu, X. Jing, J. Chen, S. Deng, Z. Zhou, J. Du, L. Yin, Y. Yan, H. Mou, and Q. She. Antibodies can last for more than 1 year after sars-cov-2 infection: A follow-up study from survivors of covid-19. *Frontiers in Medicine*, 8, 2021. ISSN 2296-858X. doi: 10.3389/fmed.2021.684864. URL `https://www.frontiersin.org/article/10.3389/fmed.2021.684864`.

R. Yang, X. Gui, and Y. Xiong. Comparison of clinical characteristics of patients with asymptomatic vs symptomatic coronavirus disease 2019 in wuhan, china. *JAMA network open*, 3(5):e2010182–e2010182, May 2020. ISSN 2574-3805. doi: https://doi.org/10.1001/jamanetworkopen.2020.10182. URL `https://pubmed.ncbi.nlm.nih.gov/32459353`.

E. L. Yeyati and F. Filippini. Social and economic impact of covid-19. *Global Economy and Development program at Brookings*, 2021. URL `https://ourworldindata.org/coronavirus`.

L. Zhong, C. Gao, Z. Zhang, N. Shi, and J. Huang. Identifying influential nodes in complex networks: A multiple attributes fusion method. In D. Ślezak, G. Schaefer, S. Vuong, and Y. Kim, editors, *Active Media Technology*, volume 8610. Springer, Cham, 2014. doi: https://doi.org/10.1007/978-3-319-09912-5_2.

# Appendix A

# Code summary

This appendix explains how to use the code developed for this thesis and where you can find it.

The code can be found here: Critical Node Problem

The code is divided in two main repositories:

- `node_detection`: contains the code developed to find the N more important nodes within a network to test and mitigate the diffusion of COVID-19. To find the N more important nodes within the network we propose 10 different strategies.
- `gym_covid`: It is an OpenAI gym interface for Reinforcement Learning. The objective is to train a RL agent to learn how to allocate the tests within the network to reduce the number of deaths.

## node_detection

The code can be found here: node_detection

The node_detection repository simulates the diffusion of the COVID-19 pandemic in a graph network. At each each simulation time step finds the $N$ most relevant nodes within the network, performs a test on them and if tested positive they are quarantined to contain the diffusion of the pandemic. Each simulation retrieves the result of a cost function that depends on the number of tests performed and the number of people infected.

**Usage**

In the `main.py` script, define and initialize the graph network that you want to use as well as the number of nodes and the probability of edge creation, in this project the network used was an erdos renyi graph from the Python package NetworkX.

```
g = nx.erdos_renyi_graph(number_nodes, prob_edge)
```

Instantiate the diffusion model and the strategy to use to find the most relevant nodes in the network.

```
#Strategy selection
model = SIR(g)
```

The diffusion model designed is a DiffusionModel class from NDLIB Python software package, refer to DiffusionModel to know how this works and how it can be modified. It is a simple Susceptible-Infected-Recovered model presented in Part III.The different strategies to detect the critical nodes are:

```
#Random nodes selection
model =  SIR(g)
#Node selected according to Degree centrality
model =  SIRDet(g)
#Node selected according to Eccentricity centrality
model =  SIREc(g)
#Node selected according to Betweenness centrality
model =  SIRBe(g)
#Node selected according to PageRank centrality
model =  SIRPar(g)
#Node selected according to Degree Discount centrality
model =  SIRDDB(g)
#Node selected according to VoteRank centrality
model =  SIRVR(g)
#Node selected according to Community Based method
model =  SIRCB(g)
#Node selected according to Contact tracing (MI)
model =  SIRMI(g)
#Node selected according to Contact tracing plus centrality method
model =  SIRDI(g)
```

For an explanation of each of them, please refer to Section 3 in Part III

Specify the parameters of the model by using a Configuration object. This will rule hoe the model behaves.

```
# Model Configuration
cfg = mc.Configuration()
#Infection rate
cfg.add_model_parameter('beta', 0.01)
#Numer pf tests performed at each iteration
cfg.add_model_parameter('tests', test)
#Penalty for a person being infected - cost function parameter
cfg.add_model_parameter('penalty', 1)
#Cost of doing a unitary test - cost function parameter
cfg.add_model_parameter('test_cost', 0.2)
```

```python
#Penalty for nodes being infected at the same time - cost function parameter
cfg.add_model_parameter('infected_penalty', 0.3)
#True positive rate
cfg.add_model_parameter("rhotp", 0.95)
#False positive rate
cfg.add_model_parameter("rhofp", 0.02)
#Initial fraction of nodes infecetd
cfg.add_model_parameter("fraction_infected", 0.05)
model.set_initial_status(cfg)
```

Define the number of iterations and execute the simulation. Please refer to DiffusionModel to understand what happens in each simulation.

```python
#number of iterations
iters=100
iterations = model.iteration_bunch(iters)
```

Finally the trends of the dynamics of the simulation are visualized using the NDLIB package ndlib.viz.bokeh, which is a extension of Bokeh library.

```python
#Diffusion trend visualization
trends = model.build_trends(iterations)
viz = DiffusionTrend(model, trends)
p = viz.plot(width=700, height=500)
show(p)
```

## gym_covid

The code can be found here: gym_covid

The gym_covid repository brings a Reinforcement Learning interface that simulates the diffusion of the COVID-19 pandemic in a graph network. The diffusion model used is a self-proposed model called SEPIA. The agent goal is to learn how to allocate at each simulation step a set of tests to minimize the number of deaths due to COVID-19.

### Usage

In the `main.py` script, define and initialize the graph network that you want to use as well as the number of nodes and the probability of edge creation, in this project the network used was an erdos renyi graph from the Python package NetworkX.

```
g = nx.erdos_renyi_graph(number_nodes, prob_edge)
```

Select and initiate the environment. There are two environment available in this repository, the difference between them is the type of action that the agent takes at each step. The `"gym_covid:covid-v0"` environment receives four actions as described in Section 2.3 in Part V: *Optimizing the number of tests using a RL model* and the `"gym_covid:covid-v2"` environment receives three actions as described in Section 2.2 in Part V: *Base RL model with fixed number of tests*.
The environment is a diffusion model, it inherits the DiffusionModel class from NDLIB Python software package, refer to DiffusionModel to know how this works and how it can be modified. It is a simple Susceptible-Infected-Recovered model presented in Part III.

```
#Environment selection
env = gym.make("gym_covid:covid-v0", graph=g)
```

Specify the parameters of the model by using a Configuration object. This will rule how the model behaves. For a reasoning on how these parameters have been set to imitate the diffusion behaviour of COVID-19 refer to Section 2 in part IV.

```
# Model Configuration
cfg = mc.Configuration()
#fatality rate of people with severe symptoms
cfg.add_model_parameter('alpha_i', 0.014)
#transmission rate of post-latent people
cfg.add_model_parameter('beta_p', 0.16)
#transmission rate of people with severe symptoms
cfg.add_model_parameter('beta_i', 0.032)
#transmission rate of people with no/mild symptoms
cfg.add_model_parameter('beta_a', 0.032)
#latency rate
cfg.add_model_parameter('sigma_e', 0.3)
#post-latency rate
cfg.add_model_parameter('sigma_p', 0.8)
#fraction of infected people with symptoms (severe)
```

```
cfg.add_model_parameter('delta', 0.6)
#recovery rate of people with severe symptoms
cfg.add_model_parameter('gamma_i', 0.1)
#recovery rate of people with no/mid symptoms
cfg.add_model_parameter('gamma_a', 0.1)
#quarentine rate
cfg.add_model_parameter('gamma_q', 0.07)
#Cost of doing a unitary test - cost function parameter
cfg.add_model_parameter('test_cost', 0.0002)
#Penalty for a dead node - cost function parameter
cfg.add_model_parameter('death_penalty', 1)
#Initial fraction of nodes infecetd
cfg.add_model_parameter('fraction_infected', 0.01)
#Numer pf tests performed at each iteration
cfg.add_model_parameter('B_t',1000)
#True positive rate
cfg.add_model_parameter("rhotp", 0.95)
#False positive rate
cfg.add_model_parameter("rhofp", 0.02)
env.set_initial_status(cfg)
```

Initiate the agent and build the Reinforcement Learning model. The trained algorithms used on this document are explained in Section 2.2 in Part V. They are a set of improved implementations of Reinforcement Learning (RL) algorithms based on `OpenAI Baselines`.

```
agent = Agent(env)
# If the environment don't follow the interface, an error will be thrown
check_env(env, warn=True)
# build the RL training model
model = A2C("MlpPolicy", env, verbose=1)
# define the number of simulation steps and train it
time_steps = int(10000)
model.learn(time_steps)
```

Once that the RL model has been trained, run a simulation are taken accordingly with wath the RL model has learn by interacting with the environment. First of all, you need to call the function `env.reset()` whenever a new simulation starts, if not the environment will start as the environment on the last step of the previous simulation. Define the number of simulation time steps (analogue to real days) although the simulation will stop if `done` is True, that is, if there is no infected people in two consecutive steps. You also need to create and object `system_status` to store the information of the network at each step

```
system_status = []
#reset the environment and set the initial state as the initial observation
observation = env.reset()
for _ in tqdm.tqdm(range(0, iters)):
    #the model decides and action based on the previous iteration
    action, _states = model.predict(observation)
    #run a step of the simulation with the action selected by the model
    observation, reward, done, info = env.step(action)
    #Store the information of each iteration
    stem_status.append(info)n
    print("obs=", observation, "action=", action, "reward=", reward, "done=", done)
    #if done is True (there is no infected people in two consecutive time steps)
     finish the simulation
    if done:
        break
```

Finally visualize the diffusion trends given by the simulation. They are done through a self-developed class that inherits the `DiffusionTrend` class in `NDLIB`. You can select the trends that you want to plot by specifying the statuses as a parameter in both methods `env.build_own_trends` and `DiffusionTrend2`

```
iterations = system_status
trends = env.build_own_trends(iterations, [1,2, 3, 4,6,7])
viz = DiffusionTrend2(env, trends, ["Exposed" , "Presymptomatic","Infected",  "
     Asymptomatic", "Quarentized", "Dead"])
p = viz.plot(width=900, height=500)
show(p)
```

You can also build the evolution in time of the Reproduction number estimated on the number of people infected

```
trends = env.build_trends(iterations)
env.get_reproduction_number(trends)
viz = DiffusionTrend1(env, trends)
p1 = viz.plot(width=900, height=500)
show(p1)
```

## Diffusion Model

The diffusion model implemented is a class inherited from the parent class DiffusionModel from NDLIB. Also note that the diffusion model in the gym_covid repository acts as a RL environment, so it also is a child of the Env class from OpenAI gym package. In this document we have presented two different diffusion models, a simple SIR model and the extended SEPIA model. In this section we will explain how a SEPIA object works so an easier model can also been easily understood.

Also note that in this section we are explaining only the Diffusion model, if you want to use it as a RL environment please refer to the next section

As every Python class, first step is to define the __init__ method and the self parameters, the most important are the possible states of the network and the parameters of the network which are defined in the Configuration object in main.py.

```python
def __init__(self, graph, seed=None):
    """
         Model Constructor
         :param graph: A networkx graph object
     """
    super(__class__, self).__init__(graph, seed)
    self.available_statuses = {
        "Susceptible": 0,
        "Exposed": 1,
        "Presymptomatic": 2,
        "Infected": 3,
        "Asymptomatic": 4,
        "Recovered": 5,
        "Quarentized": 6,
        "Dead": 7
    }
    self.parameters = {
        "model": {
            "beta_p": {
                "descr": "transmission rate of post-latent people",
                "range": [0, 1],
                "optional": False},
            "alpha_i": {
                "descr": "fatality rate of people with severe symptoms",
                "range": [0, 1],
                "optional": False},
  #The rest of the parameters are not represented for visual commodity reasons
        },
        "nodes": {},
        "edges": {},
    }
    self.name = "SEPIA"
```

The class works as follows: during each simulation iteration all the nodes in the network are asked to (i) evaluate their current status and to (ii) (eventually) apply a matching transition rule. The iterations are called through the abstract method `iteration(self)` from inherited class Diffusion model or through `step(self, action)` if the models acts also as a RL environment.

These methods first check if the actual iteration is the first or not. If it is the first step, the centrality according the most important nodes is called and stored in the buffer. It is only called once as the topology of the network will not vary during the simulation. The buffer is a circular buffer which store at the end of the buffer the nodes that have already been checked, in order to not to test always the same nodes.

```python
#creates a dictionary with all the nodes and their statuses
actual_status = {node: nstatus for node, nstatus in future.utils.iteritems(self.status
    )}
#check if it is the first iteration
if self.actual_iteration == 0:
    #Ranks the nodes in order of importance according the defined centrality
    centrality = self.nodes_to_check()
    self.node_buffer = centrality.copy()
    self.buffer_capacity = len(self.graph.nodes)
    self.actual_iteration += 1
    delta, node_count, status_delta = self.status_delta(actual_status)
    #stores the information of the network at iteration 0
    if node_status:
        return {"iteration": 0, "status": actual_status.copy(),
                "node_count": node_count.copy(), "status_delta": status_delta.copy(),
                "cost":0, "centrality": centrality.copy()}
```

If it is not the first iteration, then all the nodes in the network are asked to (i) evaluate their current status and to (ii) (eventually) apply a matching transition rule, the transition rules are ruled by the Dynamic equations presented in PartIV.

```python
for u in self.graph.nodes:
    #check the status of the node
    u_status = self.status[u]
    #calculate a random number which will be compared against the transmision rates to
      decide the transition of the node
    eventp = np.random.random_sample()
    #calculate the neighbors of the node
    neighbors = self.graph.neighbors(u)
    if self.graph.directed:
        neighbors = self.graph.predecessors(u)
    #check if the status of the node is 0 ("Susceptible")
    if u_status == 0:
        #check how many of the neighbors are infectious, that is, are Presymptomatic.
     Asymptomatic or Infected
        presympt_neighbors = [v for v in neighbors if self.status[v] == 2]
        sympt_neighbors = [v for v in neighbors if self.status[v] == 3]
        asympt_neighbors = [v for v in neighbors if self.status[v] == 4]
        #estinmate the exposure rate
        exposure_rate = 1- ((1 - self.params['model']['beta_p']) ** len(
     presympt_neighbors) * (1 - self.params['model']['beta_i']) ** len(sympt_neighbors)
      * (1 - self.params['model']['beta_a']) ** len(asympt_neighbors))
```

```
        # check if the node are selcted to be tested, test it and if is a false positive quarantine it
        if u in nodes_to_test and eventp < self.params['model']['rhofp']:
            actual_status[u] = 6 #Quarentized
    #check if the status of the node is 1 ("Exposed")
    elif u_status == 1:
        # check if the node must be tested, test it and if is a false positive quarantine it
        if u in nodes_to_test and eventp < self.params['model']['rhofp']:
            actual_status[u] = 6 #Quarentized

        else:
            if eventp < self.params['model']['sigma_e']:
                actual_status[u] = 2 #Presymptomatic
    #check if the status of the node is 2 ("Presymptomatic")
    elif u_status == 2:
        # test the node if applicable, presympt nodes could infect other nodes and can test positive
        if u in nodes_to_test and eventp < self.params['model']['rhotp']:
            actual_status[u] = 6 #Quarentized
        else:
            if eventp < self.params['model']['sigma_p']:
                symptp = np.random.random_sample() #probability of being symptomatic
                if symptp < self.params['model']['delta']:
                    actual_status[u] = 3 #symptomatic
                else:
                    actual_status[u] = 4 #Asymptomatic
#The rest of the statuses are omitted in this document for simplicity
```

Once that all the nodes have been checked, increase the number of the iteration and return the information of this iteration

```
done = self.done()
if node_status:
    return {"iteration": self.actual_iteration - 1, "status": delta.copy(),
            "node_count": node_count.copy(), "status_delta": status_delta.copy(), "confirmed_cases": nodes_exposed}
else:
    return{"iteration": self.actual_iteration - 1, "status": {},
            "node_count": node_count.copy(), "status_delta": status_delta.copy(), "confirmed_cases": nodes_exposed}
```

There are two other methods that you need to configure, the `done()` to check if the simulation must ends if there has been two consecutive simulation steps with no nodes infected and the `nodes_to_check()` to ranks the nodes in order of importance according to the centrality selected.