The University of Cape Town

Faculty of Science

Department of Mathematics and Applied Mathematics

July 2023

Applied Mathematics Masters Thesis

Submitted for the degree of Master of Science

Optimizing COVID-19 Control Measures Using Multi-Objective Deep Reinforcement Learning

by

Arinze Lawrence Folarin

MAM5001W

Supervisors: Associate Professor Jonathan Shock and Dr Ndivhuwo Makondo

Declaration of Authorship

I, Arinze Lawrence Folarin, declare that this thesis titled *Optimizing COVID-19 Control Measures Using Multi-Objective Deep Reinforcement Learning*, and the work presented in it are my own. I confirm that:

- This work was done while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this has always been clearly attributed.
- Where I have used figures and/or diagrams from the work of others, the source is always given. With the exception of such figures/diagrams, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: Signed by candidate

Date: 21/06/2023

Abstract

A crucial area of global research is the hunt for efficient non-pharmaceutical methods to stop the spread of diseases. Recent research has shown that reinforcement learning can be a helpful tool in the medical industry to address challenging and delicate issues. The goal of this study is to improve COVID-19 control measures through the use of multi-objective deep reinforcement learning techniques. The results of two case studies, one using a Pareto conditioned network on COVID-19 data from Belgium and the other using a Deep Q-Network, Goal-DQN, and Non-dominated Sorting Genetic Algorithm (NSGA-II) on COVID-19 data from France, are evaluated using both binomial (Stochastic) and Ordinary Differential Equation mathematical models. The study highlights the potential of multi-objective deep reinforcement learning as a method of optimizing public health interventions by shedding light on the optimum COVID-19 control methods for various scenarios and models. Findings show that the suggested strategies are efficient in figuring out the best preventive actions by striking a balance between two crucial choice difficulties encountered when trying to stop the spread of COVID-19 in particular areas. This study makes a substantial contribution to the ongoing fight against pandemics like the COVID-19 event.

Acknowledgements

I would like to express my heartfelt appreciation to the individuals who played a significant role in my life during the past two years.

Jonathan Shock - who was my examiner for my first masters thesis and later accepted to be my supervisor. I am sincerely at a loss for words when it comes to expressing my gratitude towards you. Your unwavering support and encouragement have propelled me to become a resilient individual, especially when it comes to my research pursuits. You consistently strive to comprehend every situation I bring forth and reassure me that everything will be alright, always encouraging me to include you in the process. I must commend you for being an exceptional supervisor. I genuinely express my heartfelt appreciation for your efforts in providing me with funds to extend my studentship to this year. I had believed that it had come to an end last year, but your support has allowed me to continue my educational journey, and for that, I am truly grateful.

Ndivhuwo Makondo, who was my supervisor for my first masters thesis and also agreed to work together with me again. I genuinely value your dedication and invaluable contribution that led to the success of the work. I aspire to have the opportunity to meet you in person someday and express my gratitude firsthand.

Olutayo Oluseun, my lovely and ever supportive wife. She has been a tremendous source of mental support, continuously offering prayers and motivation to keep me going when I encounter challenges along the way and feel tempted to give up on the project.

I am filled with profound gratitude towards the University of Cape Town and African Institute for Mathematical Sciences (AIMS) for their invaluable contributions to the financial support of this program. Their generous assistance has been instrumental in making this opportunity possible for me. I sincerely appreciate their kindness and support, and I earnestly aspire to one day repay them in kind.

To my parents, a big thank you for your prayers and unwavering parental support. I hope that one day, I will have the opportunity to repay your kindness twofold and truly show my appreciation for all that you do.

Contents

1 Introduction

The need to take necessary steps to prevent and control the transmission of infectious diseases, including HIV/AIDS, influenza, malaria, and COVID-19, has become increasingly urgent due to the global concern about their spread. To achieve this goal, two main strategies can be implemented: pharmaceutical interventions and non-pharmaceutical interventions. Implementing both pharmaceutical and non-pharmaceutical interventions is imperative for efficiently preventing and managing the spread of infectious diseases, especially in times of global health crises. Vaccines and antiviral medications are useful pharmaceutical interventions, but their accessibility and availability can be limited during epidemics or pandemics. Therefore, organizations like the World Health Organization (Organization et al., 2019) advise relying on non-pharmaceutical interventions such as travel guidance, hand hygiene, and face masks to help suppress the spread of infections.

The COVID-19 outbreak which resulted from the severe acute respiratory syndrome coronavirus, was a significant public health emergency that has since been brought under control. In December 2019, it started in Wuhan, China, and quickly expanded to every continent. According to the World Health Organization (WHO) (Organization, 2023), the total number of confirmed cases globally has exceeded 765 million as of April 30th, 2023, with over 6.9 million reported deaths. The disease is primarily transmitted through respiratory droplets and symptoms can range from moderate to severe, including fever, coughing, and breathing difficulties. In extreme cases, acute respiratory distress syndrome, pneumonia, and even death are possible.

On January 30, 2020 and March 11, 2020, respectively, the WHO labeled the outbreak a global health crisis and a pandemic (Organization, 2020). Governments and health organizations worldwide implemented various measures to contain the virus, including lockdowns, school closures, travel bans, and mask mandates. It is important to acknowledge that the cooperation between governments and organizations has successfully led to the implementation of effective measures to prevent and control the spread of COVID-19. The COVID-19 epidemic significantly impacted the global economy and had significant negative impacts, resulting in business closures and the loss of millions of jobs, in addition to its human toll. The pandemic underscored the need for readiness in the face of future outbreaks, as well as the importance of efficient virus-controlling measures and vaccines. The epidemic brought about significant changes to the world, affecting how we live, work, and communicate with each other. The COVID-19 pandemic has resulted in a widespread outbreak of illness and death, as well as causing significant disruptions to economies and societies. However, the efforts to manage the pandemic

have been successful in curbing the virus spread, emphasizing the importance of ongoing vigilance in managing the situation. One potential strategy for achieving this goal is to utilize deep reinforcement learning (DRL) algorithms. DRL involves training neural networks to make decisions based on feedback from the environment. In this study, we will examine the use of DRL algorithms in controlling the spread of COVID-19. Specifically, we aim to determine how these algorithms can balance various objectives, such as reducing infections while minimizing the negative impacts of control measures on the economy and society.

Addressing the pandemic has relied heavily on ongoing implementation of public health procedures include examination, traceability of contacts, and isolation, as well as the creation of vaccinations and medications.

The use of non-pharmaceutical interventions (NPIs) is a well-established approach for stopping the transmission of infectious diseases. However, policymakers face different challenges in implementing NPIs effectively, including identifying the most efficient intervention, assessing the cost of different strategies, determining the best timing for implementation, and evaluating various scenarios and variables. Decision-makers often struggle to balance different goals, such as implementing NPIs and containing the epidemic. To overcome these challenges, recent studies suggest applying Optimal Control Theory, as reported in studies by (Richard et al., 2021; Kantner and Koprucki, 2020).

Optimal control theory is a useful tool in the examination of epidemiological models (Richard et al., 2021; Uddin et al., 2020; Kantner and Koprucki, 2020) and population data (Kwak et al., 2021) related to infectious diseases. Iterative schemes such as Runge-Kutta can be employed to solve the optimality system of malaria epidemiology (Agusto et al., 2012). BOCOP (The optimal control solver) together with MATLAB can be used to examine how Central Bank actions affect the economy (Kostylenko et al., 2017). Different optimal control techniques can be employed to optimize single or multiple objectives. Another approach that has gained popularity is reinforcement learning, which combines sophisticated epidemic models with automatic learning of preventative tactics and sequential decision-making.

The authors in (Reymond et al., 2022a) recently found a set of solutions that roughly corresponds to the Pareto front using cutting-edge techniques like the Pareto Conditioned Networks algorithm in order to lessen infections/hospitalizations and the social burden brought on by mitigation measures during the initial Belgian COVID-19 pandemic. The study presented the decision problem as a multi-objective problem, and the EpidemiOptim Python toolbox was used to enable communication between epidemiologists and optimization researchers (Colas et al., 2021).The toolbox employs the standard interface of OpenAIGym to convert epidemiological models and cost functions into optimization challenges.

1.1 Research Goals

The goal of this research is to study the potential of deep reinforcement learning methods, such as Deep Q-Networks and Actor-critic algorithms, along with multi-objective optimization techniques like NSGA-II and Pareto-conditioned networks, to address an optimization problem related to COVID-19. The objective is to optimize the objective functions involved by formulating the problem as a reinforcement learning task. This approach enables us to implement the algorithms mentioned above to tackle the optimization problem. In this context, we are examining two different COVID-19 problems. Firstly, we consider a situation where the mitigation measure involves alternating lockdowns, with the goal of minimizing the costs associated with controlling the spread of the disease in a targeted population. The cost associated with this problem consists of both health costs and economic costs. Health costs result from the strain on the healthcare system if the spread of the virus is not effectively controlled and can lead to increased hospitalizations and deaths. While lock-downs, social isolation, and travel restrictions impede the transmission of the virus, they also have an adverse effect on the economy since they result in job losses and decreased economic activity. Secondly, we consider a COVID-19 problem in which social mitigation measures have an impact on infection and hospitalization rates as well as creating a social burden for a targeted age-structured population.

The objective of the research is to address the following inquiries: Which algorithm would be most appropriate for the two mentioned COVID cases if we apply the algorithms and assess their Pareto fronts (the set of best feasible compromises among all objectives) to determine which one performs better using the evaluation metrics? Furthermore, our study aims to explore whether it is appropriate to consider problems related to COVID-19 control as multi-objective reinforcement learning problems. By applying various algorithms and assessing their Pareto fronts, we hypothesize that the Pareto Conditioned Network algorithm will demonstrate superior performance for the two COVID cases in question. Furthermore, we predict that approaching these problems as multi-objective reinforcement learning problems will yield better results than DQN.

1.2 Thesis Outline

The thesis is structured into four chapters. Chapter 1 serves as an introduction, providing a background of the work, research goals, and hypotheses. In Chapter 2, a comprehensive literature review is conducted to examine past works related to this research and identify key factors to guide the research goals. Chapter 3 presents the experiments and results, while Chapter 4 offers the conclusion and recommendations, followed by a list of references.

2 Literature Review

The objective of this chapter is to provide a comprehensive understanding of the research by presenting essential concepts. As the research focuses on a disease-related matter, we will start by exploring infectious disease modeling and epidemiological models. Furthermore, we will delve into the approaches and algorithms that will be investigated to tackle the challenge of minimizing disease transmission within a specific population.

2.1 Infectious Disease Modelling

According to (Quah, 2016), an infectious disease refers to an illness caused by a virus or its toxic byproducts that can be spread to a susceptible host through various means such as contact with an infected person or animal, or a contaminated inanimate object. A host's susceptibility to the disease, the environment, and the biological characteristics of the pathogen all have an impact on how the infectious agent interacts with it. In addition, as described by (da Costa et al., 2019), social and cultural factors can facilitate or inhibit the host's exposure to disease origins and its spread in the population.

Everyone can contract an infectious disease. A person may be at a higher risk if their immune system is compromised or if they go to areas where specific highly transmissible diseases are common.

According to Davis and Lederberg (2001), infectious diseases are a leading cause of death globally, accounting for more than 13 million deaths each year. Individuals in developing countries bear the greatest burden of morbidity and mortality, particularly infants and children, who account for about 3 million deaths each year from malaria and diarrheal diseases alone. In wealthy countries, underprivileged minorities and indigenous communities are disproportionately affected by infectious illness mortality. Infectious diseases not only endanger human health, but they also have serious economic repercussions. For instance, the Ebola virus outbreak in West Africa cost the region close to six billion dollars, with the outbreak's overall worldwide economic impact pegged at over 15 billion (Ross et al., 2015). Policymakers must develop control techniques to slow or stop the spread of infectious diseases, especially in the absence of efficient drugs or vaccinations. Epidemiological modeling is essential for forecasting outbreak development and reducing their impact.

In the event of disease outbreaks or epidemics, infectious disease modeling is critical for developing rational prevention and control strategies as well as health policy.

Infectious disease modeling has developed into a crucial tool for improving decisions regarding public health for infectious disease control, according to various research (Lessler and Cummings, 2016; Alexander et al., 2012; Sah et al., 2021; Royce and Fu, 2020). These models

concentrate on illness progression within the body, disease transmission pathways, predicting the course of an outbreak, and evaluating epidemic control methods. In reality, during the early stages of the 2009 influenza pandemic, decision-makers in national outbreak response units and the WHO outbreak response team both used mathematical modeling (Van Kerkhove and Ferguson, 2012). Important challenges concerning how to construct a mathematical model of an outbreak and how models might direct public health actions at different stages of an outbreak are raised by these modeling efforts. Figure (1) presents a summary of the steps involved in modeling an infectious disease.

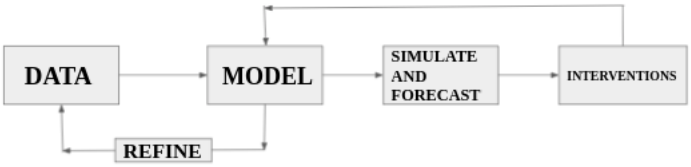
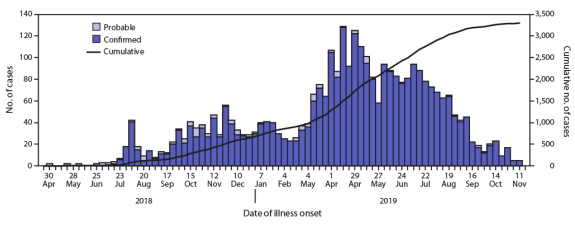


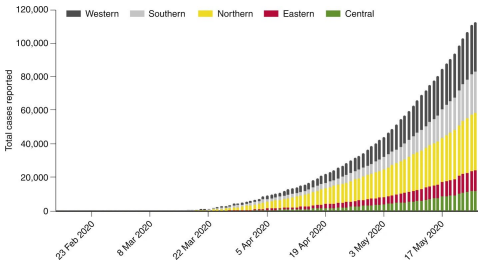
Figure 1: Infectious Disease Modelling

DATA

To estimate disease transmission and control actions, infectious disease modelers need real-world data. They typically gather information to analyze the transmission of a disease in the population once an outbreak or epidemic has been confirmed. These statistics are frequently gathered from a variety of sources, including as demographic data, illness monitoring, and immunization programs. Modelers are able to have an impact on health policy decisions by combining all these data sources into a single model. For instance, during the COVID-19 epidemic, Xu et al. (2020a) employed a mathematical model to guide public health policy. The first stage in creating a mathematical model of an outbreak is to analyze outbreak data, which includes elements like the daily incidence of cases, as shown in figures (2a) and (2b).



(a) Weekly Onset and Cumulative Number of Confirmed and Probable Cases of Ebola Virus Disease— Democratic Republic of the Congo, April 30, 2018–November 17, 2019 (Aruna et al., 2019)



(b) COVID-19 Cases in Africa by Region, February 15 to May 22, 2020 (Massinga Loembé et al., 2020)

MODEL

The next stage is to build a mathematical model that represents the underlying epidemiology of the system after analyzing the data gathered during an outbreak, as shown in figures (2a) and (2b). This entails analyzing the curve/shape of the observed data distribution for the epidemic (as shown in figure (2a)) and creating a mathematical model to approximate it. A number of presumptions about the inhabitants and organization of the real world, as well as estimations of the model input parameters, form the basis of the model. If the model does not precisely reflect the observed data, we start with the most fundamental assumption about how an infectious illness spreads and then refine the model until it roughly approximates the underlying dynamics of the outbreak. We should keep improving our model until it closely matches the findings of the dynamics of the infectious diseases we are modeling in the actual world. The common quantitative models for characterizing the transmission and development of infectious diseases will be discussed in the section that follows.

SIMULATE AND FORECAST

While solving mathematical models analytically can be difficult, they are frequently solved numerically with the aid of a computer. In order to do this, a computer simulation of the mathematical model must be created. Data must then be collected and used to enhance the simulation model (Wang et al., 2022). Having the right data is essential for the simulation model to deliver findings that are accurate and acceptable. With simulation, we can forecast future results, including the anticipated daily caseload. Yet in order to do so, it is essential that our models go through calibration, an iterative process of refining and comparing the model with the real system, and validation, which entails comparing the model's behavior to the real-world system.

INTERVENTIONS

Once we have a model that can reasonably predict the number of cases we will likely see in the future, we then incorporate control interventions into the model to investigate how different control interventions might change that number. Many computer simulations demonstrate that the adoption of an intervention approach affects the management of the COVID-19 epidemic's transmission dynamics (Mondal and Khajanchi, 2022).

In summary, infectious disease models have a wide range of applications, including predicting disease outbreaks, assessing the efficacy of preventive measures, and assisting in public health decision-making. It's crucial to remember that infectious disease models have some restrictions, and their forecasts do not always match actual results. As a result, model results should be

carefully considered in addition to data from other sources.

2.1.1 Epidemiological Models

The discussion in this section will center on epidemiological models and how they might be used to understand and stop the spread of illnesses. An epidemic occurs when an infectious disease spreads widely and quickly among many people. The study of epidemics, illnesses generally, and even non-disease health situations is known as epidemiology. Epidemiological models try to capture how a disease develops over time and how it spreads. In the study of epidemiology, a model can only be regarded as genuine if it can accurately reflect the disease's natural history, including elements like the length of infectivity, the length of illness, and any relevant epidemic or pandemic dynamics.

Epidemiology models show how a disease develops over time and how it spreads. These models must accurately depict the disease's natural history, including its duration of infectivity, incubation period, and sickness, in addition to any pertinent dynamics associated with epidemics or pandemics, in order to be accepted as valid. The model categorizes people into different epidemiological states, such as vulnerability, resistance, sickness, and contagiousness, to reflect the interactions between various variables, such as the illness's duration, latent period, and incubation period proportion of asymptomatic cases, case-fatality rate, recovery rate, immunity, and reinfection rate. The disease's natural history, which explains how individuals move from being vulnerable to incubating to ill to recovering to dying, is the source of the epidemiological states, which describe how a disease spreads and affects a population. Based on information about the disease's history and individual health data, including age and comorbidities, the model's transmission rate and force of infection are crucial elements. The set of differential or difference equations that describe the dynamics of the model theoretically can be solved using a method of repeated iterations. The model is validated by simulating the infection's natural development and including healthcare interventions, and it should be able to replicate real-world data to confirm its accuracy. Without needing all the details from the previous data, a good epidemiological model can mimic prior endemic, epidemic, or pandemic scenarios after it has been constructed. If the model can faithfully mimic the characteristics of previous pandemics or epidemics, it is considered valid. Finally, the model can be used to simulate various interventions and deviations from the epidemic's or pandemic's natural trajectory.

Many applications of epidemiological models exist, such as cost-effectiveness and cost-benefit studies, which show the benefits of shifting funds from ineffective techniques to initiatives that are both more effective and less expensive. They are helpful for creating health plans, putting epidemiological interventions into action, and offering instruction and training. Epidemiological

models have been utilized extensively in studies to better understand and manage particular diseases. For instance, the authors of (Macdonald et al., 1957) used a proposed malaria epidemiological model to understand and control the disease, while the authors of (Bent et al., 2018) applied a Gaussian Process regression to simulate malaria in a stable population in order to stop malaria from spreading. Likewise, zoonotic disease and other parasitic infections with definitive hosts were the subject of mathematical models provided by Nasell and Hirsch (1971), as well as models for measles that were examined by (Bartlett, 1960) and Muench (1959)). For other viral infections, (Elveback et al., 1964) introduced the Reed-Frost model in 1964, and (Waaler and Piot, 1969) employed an epidemiological model to calculate the efficacy of tuberculosis prevention efforts in 1969. As shown by numerous studies like (Colas et al., 2021),(Reymond et al., 2022a),(Giordano et al., 2020),(Arino and Portet, 2020),(Khoshnaw et al., 2020), and (Calafiore et al., 2020), epidemiological models are increasingly being used to study and manage the transmission of COVID-19.

2.1.2 Types of Epidemiological Models

Epidemiological models of infectious diseases can be seen in the following forms;

Deterministic or compartmental models

The conventional Susceptible-Infected-Recovered (SIR) framework, which entails allocating diverse populations to various subgroups and is widely used when dealing with large populations, provides the foundation for these models, which require a set of ordinary differential equations. To develop deterministic/compartmental models, which allow us to forecast the future, differential equations are used. These models are very helpful for comprehending how a disease spreads across a population.

Stochastic or random models

As epidemiological research expanded and occasionally dealt with smaller groups, the element of chance and variation grew more significant, necessitating the necessity for a probability model (Omram, 2001). The stochastic model depicts an unpredictable scenario, which is important when there are few infectious individuals present or when other variables, such as the environment, spread, recuperation, births, or deaths, introduce variability and have a big impact on how an epidemic develops. In many ecosystems, stochastic models allow for random fluctuations in dynamics, which take into account environmental and/or external perturbations and account for things like exposure risk and the infectious vector itself.

Agent-based models

These simulations represent the behaviors and interactions of people within a population. They are helpful in understanding how a person's conduct affects how a disease spreads. Here, agents interact and make decisions according on pre-established rules that take into account their environment, the behaviors of other agents, and their own preferences and traits.

Dynamical models

These models explain how a disease spreads over time using mathematical formulae. They are helpful for forecasting a disease's future spread and comprehending how it develops. We also have statistical models that examine data on the transmission of disease using statistical methods. They are helpful for identifying trends in the data and forecasting upcoming outbreaks.

The population can be divided into numerous classes that alter over time in an epidemiological model (that is, as the disease progresses, individuals move through the different stages of infection). They are categorized as being removed/recovered R(t), infectious I(t), and susceptible S(t).

- Susceptible S(t): Those who have not yet developed the sickness are the demographic being discussed here. Some people lack immunity because they haven't had the disease before or received a vaccination for it. They might then contract the disease themselves after coming into contact with other affected people. The population's vulnerability to infection might change over time as some people contract the disease while others develop protection through vaccination or other means.

- Infectious I(t): These people are those who are now afflicted with a disease and have the capacity to infect others. They already have the illness, and they are either getting well or getting sick right now. These people could or might not exhibit symptoms, but they have the potential to spread the illness through their relationships.

- Removed/Recovered R(t): This group includes people who have either recovered from the illness or have been wiped out from the population as a result of passing away or other circumstances. They can no longer spread the illness to others. These people have either recovered from the infection or have passed away from it.

Simply said, the model seeks to depict how a disease develops in a person, from the initial stage of susceptibility to infection to the stage of infectiousness and ability to spread the disease, and finally to the last stage of recovery or death. In the sections that follow, the various permutations of this transition will be further explained.

2.1.3 Mathematical Compartment Models for Infectious Diseases

Understanding and predicting the spread of infectious diseases can be done with the help of mathematical models. Although research on mathematical epidemiological models began in the early 1900s, its roots can be found in the mid-1800s, when efforts were made to apply physics to the biological sciences and as mathematical statistics and probability theory were developing (Schank and Twardy, 2009). The study on smallpox vaccination by Daniel Bernoulli (1700–1782) is often regarded as the foundation of mathematical epidemiology. Smallpox was widespread in the 18th century, and variolation—an injection technique that produces long-lasting protection against smallpox—was developed, though it came with a minor chance of infection and mortality (Brauer, 2017). The first compartmentalized model was developed by Kermack and McKendrick (1927) to analyze infectious epidemic disease dynamics.

Susceptible-Infectious-Recovery (SIR) Model

The SIR model is based on a set of three non-linear first-order differential equations. It is predicated on the idea that everyone in the population is initially equally susceptible to the disease at the start of an epidemic. Those who become infected either recover or are expelled from the population after a given amount of time. The following rate equations shows the different compartment for the SIR model

$$\frac{dS}{dt} = -\beta S(t)I(t) \tag{1}$$

$$\frac{dI}{dt} = \beta S(t)I(t) - \alpha I(t) \tag{2}$$

$$\frac{dR}{dt} = \alpha I(t) \tag{3}$$

where $\beta S(t)I(t)$ term indicates that in a unit of time, an infected member of the population establishes enough contact to infect βN others. β is the average number of contacts sufficient for transmission of the infection and α is the transfer rate from infected (I) to recovered (R).

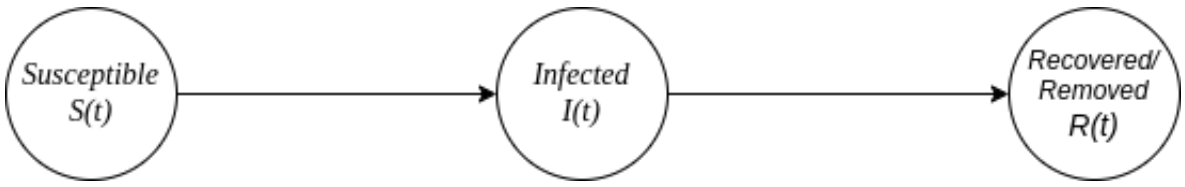


Figure 3: Basic SIR Model Flow Chart

Susceptible-Exposed-Infectious-Recovered (SEIR) Model

This model demonstrates the long latency time when infected but not yet contagious individuals are present in the exposed (E) compartment. The SEIR model is created to account for the exposed period that happens after transmission of infection from susceptible persons to potentially infectious members but prior to the onset of symptoms and infection spread. This model has been used in a variety of situations, as seen in (Biswas et al., 2014), influenza (Saito et al., 2013; Li et al., 1999; Lekone and Finkenstädt, 2006; Shah and Gupta, 2013; Radulescu et al., 2020; Mwalili et al., 2020; López and Rodo, 2021; Zisad et al., 2021; Li and Muldowney, 1995; Diaz et al., 2018), COVID-19 (He et al., 2020), dengue fever (Syafruddin and Noorani, 2012) The rate of equations for the SEIR model:

$$\frac{dS}{dt} = -\frac{\beta IS}{N}, \tag{4}$$

$$\frac{dE}{dt} = \frac{\beta IS}{N} - \gamma E \tag{5}$$

$$\frac{dI}{dt} = \alpha E - \alpha I, \tag{6}$$

$$\frac{dR}{dt} = \alpha I - \mu R \tag{7}$$

where:

- β is an average number of contacts sufficient for transmission of the infection.
- γ is the average latency period (incubation time).
- α is the transfer rate from infected to recovered.

Age-Structured SIR Model

Extensive research has shown that the impact of diseases, including COVID-19, can significantly vary across different age groups. COVID-19 dynamical models have been extensively explored to understand its varying effects on the population at large. The age-structured SIR model is a mathematical epidemiological approach that partitions a population into distinct age groups, allowing the study of infectious disease transmission. It observes how members of the population transit between susceptible (S), infected (I), and removed/recovered (R) states, considering age-specific interactions and characteristics. This model offers valuable insights into how the disease spreads across different age groups as time progresses. In an age-structured model, the integration of an age-contact matrix, C is essential. The age-contact matrix as shown in figure 4 defines the contact rate between each pair of age groups, enabling the study

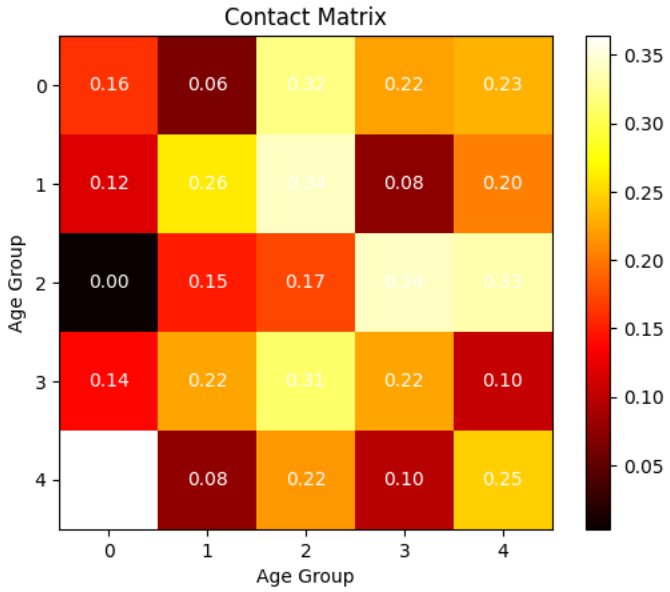


Figure 4: Example of a Age-Structured Contact Matrix showing rates of interaction between age groups. Each element (i, j) of the matrix represents the average frequency of interactions between members of group i and group j.

of interactions and transmission dynamics among different age brackets. A typical rate of equations for the dynamics of the model is given as

$$\frac{dS_k}{dt} = -\frac{\beta S_k}{N} \sum_{j=1}^{n} C_{kj} I_j, \tag{8}$$

$$\frac{dI_k}{dt} = \frac{\beta S_k}{N} \sum_{j=1}^{n} C_{kj} I_j - \gamma I_k, \tag{9}$$

$$\frac{dR_k}{dt} = \gamma I_k. \tag{10}$$

where I(t), S(t), and R(t) are vector-valued functions depicting the age distribution of infectious, susceptible, and recovered members of the population, respectively, by utilizing the k-th coordinate to represent the population count within the respective age category, and N denoting the total population. C_{kj} is the element (k, j) of the contact matrix C, β and α take the same meaning as in the SEIR Model equations above.

2.1.4 Modeling and Simulation of Infectious Diseases

Models are streamlined depictions of actual procedures or things; they are not perfect duplicates of the genuine thing. Compartment models are one type of mathematical model that can be used to forecast and analyze various scenarios. Compartment models have certain limitations, nevertheless, because they presuppose a population's homogenous racial mixing. Control measures including immunizations, antiviral medications, and social seclusion must be tried

out and put into practice if we are to effectively decrease the spread of diseases. The difficulty lies in figuring out how to carry out these laws in a way that protects the vast majority of the population. Computational or analytical modeling is a key strategy for resolving this problem. Computers are used in computational modeling to mimic the transmission of illnesses in different settings among a synthetic population.

2.1.5 Agent-Based Modeling

To mitigate the transmission of infectious diseases, one effective approach is agent-based modeling, also known as individual-based modeling (NEPOMUCENO et al., 2016), which simulates the spread of epidemics. This method represents each member of a population as a dynamic collection of attributes that evolve over time, offering a comprehensive alternative or supplement to traditional epidemiological compartment models. This allows for the modeling of population phenomena as a result of individual or agent interactions. Compartment models, on the other hand, presuppose homogenous mixing within a population. Agents with the ability to operate in a specific environment are used in agent-based modeling. The concept can be viewed as micro-scale simulations in which agents conduct activities and interact with one another and their surroundings over time, simulating real-world processes where the dynamics of the system can be seen. It was first created around the 1970s (Retzlaff et al., 2021). ABMs are used in a variety of disciplines, including business, biology, epidemics, and technology, to name a few (Bonabeau, 2002).

The transmission of infectious diseases can be simulated and modelled using agent-based modeling (ABM), which takes into consideration the specific health state of each person. This modeling strategy keeps account of each person's contacts in their social networks and local communities, including coworkers and classmates. Additionally, the model sets rules for individual agent movement, such as going to and from work or school, and takes into account the spread of the illness to other people. As a result, when used in simulations, ABM provides a more precise depiction of the evolution of disease transmission (Epstein et al., 2002). We can replicate non-pharmaceutical treatments using epidemiological transmission models to capture the intricate dynamics behind infectious disease outbreaks and comprehend such patterns (NPIs). Epidemiological models can forecast the effects of potential intervention strategies and offer information on the spread of certain diseases. To do this, reinforcement learning, an optimum control, provides a technique for automatically learning NPIs in conjunction with intricate epidemic models, which will be described in more detail in the following section.

In our case, we are considering two models derived from French COVID-19 data and Belgian COVID-19 data. The French COVID-19 model effectively portrays the intermittent lockdown strategy, offering insights into the transmission dynamics of COVID-19. In contrast to other

COVID-19 epidemic models that disregard the economic consequences of the pandemic, this model not only considers health-related costs but also the economic impact on the French population. However, a limitation of this model is its inability to account for an age-structured system in which diseases spread through both direct person-to-person contact and indirect contact, a notable deficiency given the crucial role of age in COVID-19 infection dynamics. This is where the second model, the Belgian COVID-19 model, holds an advantage over the French model. These two models were selected due to their remarkable performance in the context of the COVID-19 era, contributing significantly to the reduction of disease transmission in France and Belgium.

2.2 Reinforcement learning

Reinforcement learning is a form of machine learning that focuses on training agents to make decisions through iterative trial-and-error attempts and evaluations. The agent learns to select actions that produce the largest overall cumulative rewards over time by receiving feedback in the form of rewards or penalties. The core of reinforcement learning is the interaction of an agent with its environment. RL dates back to the 1980s and has roots in both optimal control theory and animal learning (Sutton, 1991). The "law of effect" is a psychological and behavioral science principle that contends that behaviors that result in positive outcomes are more likely to be repeated, whereas behaviors that result in negative consequences are less likely to be repeated. Animal learning in the context of RL involves learning through trial and error, similar to this principle. The concept was first put forth by psychologist Edward Thorndike in the late nineteenth century (Thorndike, 1927). It suggests that an action's results affect its likelihood of being repeated in the future, changing an individual's behavior through time. The theory of reinforcement learning (RL) is widely applied in many disciplines, including therapy, education, and training of animals. An agent learns to base decisions on the input it receives from its environment, whether in the form of incentives or penalties, in this idea, which is analogous to the usage of behavior treatments. In real life, agents learn by interacting with their surroundings and changing their behavior in response to incentives or punishments.

Reinforcement learning (RL), unsupervised learning, and supervised learning are the three main subcategories of machine learning. In supervised learning, input data and associated expected outputs are given concurrently as the system is taught using labeled data. The algorithm learns patterns and associations from this labeled data, which enables it to predict outcomes for new data with accuracy. On the other hand, unsupervised learning includes identifying patterns and relationships in unlabeled data. Reinforcement learning doesn't rely on labeled data like supervised and unsupervised learning does. Instead, it relies on trial and

error, where an agent interacts with its surroundings and gains or loses rewards depending on what it does. Positive/good activities in real life are rewarded, while negative/bad actions are penalized, thus the agent gradually learns to choose positive/good actions. This distinguishes between unsupervised learning, in which no label is given, and supervised learning, in which the appropriate behaviors are labeled. In other words, unlike supervised learning, RL delivers the right action based on training knowledge rather than instructions (Sutton and Barto, 2018a). Any issue requiring an RL solution begins with simulating the environment of an agent, and the agent can then assess whether the policy's related action resulted in a high or low reward by building a policy network to direct its actions..

We shall examine significant RL elements that demand discussion in the sections and chapters that follow. The *agent* is in charge of making a decision, communicating that decision to the environment, and getting feedback in the form of observations from the environment and a reward. It has the capacity to perceive and comprehend its environment, gain knowledge through error, and respond appropriately. Anything outside of the learning agent's control refers to *environment*. It is the environment in which the agent carries out particular duties, consents to the agent's actions, and receives incentive signals from the learning agent.There are several distinct kinds of learning environments, such as deterministic ones where the agent's actions and the present state determine the environment's future state and stochastic ones where the agent's actions are unpredictable due to their random nature. Environments can also be categorized as stationary or non-stationary depending on whether the value and dynamism of the reward remain consistent throughout time. In the world of RL, a *policy* acts as a framework that controls how the learning agent behaves at every given moment in order to maximize the desired result. In essence, a policy is a function that links states to possible actions an agent might do when in a certain state, and it can be changed to choose a different action in the future. This function's symbol is π, and the map is given as :

$$\pi : S \to A, \tag{11}$$

The above equation defines a policy, where S stands for a collection of possible states and A stands for a possible set of actions an agent could take when they are in the state $s \in S$. In RL, policies are categorized into two types: stochastic and deterministic policies. A *deterministic policy* instructs the agent to perform a certain action a while in a specified state s. This can be represented by the following equation:

$$\pi : S \to A, \pi(s) = a. \tag{12}$$

The probability of choosing an action a when in state s is provided by a *stochastic policy*, on the other hand.

$$\pi : S \times A \to [0, 1],\ \pi(s, a) = P(a_t = a | s_t = s). \tag{13}$$

The stochastic policy is represented by the equation above, where s_t is the state at time t and a_t is the action executed at time t. The term *optimal* refers to a policy that constantly chooses actions that optimize the expected return for the current state. The *Reward Function* $\mathcal{R}$, when used in relational logic, serves a significant role in outlining the goal of the RL agent.After the learning agent completes a particular action or series of actions that are appropriate for the task at hand, the environment will typically communicate a numerical signal to it as the reward function. This signal can be either positive or negative. States S or state-action (S, A) pairings are mapped by the function $\mathcal{H}$ to rewards R.

$$\mathcal{H} : S \to R, \tag{14}$$

$$\mathcal{H} : S \times A \to R. \tag{15}$$

The *state* in RL, which is a state of the RL problem, refers to the agent's current location in the environment. Both discrete and continuous state spaces are possible.
Value Functions are essential components in RL that serve to reflect the anticipated future reward. Action-value and state-value functions are the two categories that are generally defined for value functions. the projected future reward that an agent can earn from the environment starting from state s while sticking to policy π is represented as the *state-value function*. The symbol for it is $V_{\pi(s)}$:

$$V_\pi(s) = \mathbb{E}\left[G_t | s_t = s\right], \tag{16}$$

where the return at time t is denoted by G_t and the definition is as follows:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{17}$$

The discounted rate parameter γ, where $\gamma \in [0, 1]$, sets the weighting of future rewards in comparison to immediate rewards. The value that the learning agent receives at a particular moment t after performing an action in a particular state is known as the reward (R_t). When beginning from state s, executing action a, and adhering to policy π, the action-value function $Q_{\pi(s,a)}$ represents the anticipated result. It assesses the worth of taking action in a certain

state.

$$Q_\pi(s,a) = \mathbb{E}_\pi \left[R_t | s_t = s, a_t = a \right], \tag{18}$$

$$= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s, a_t = a \right]. \tag{19}$$

where $\mathbb{E}_\pi$ is the expectation given a policy π. *Model:* It is possible to represent the behavior of the environment, and some RL problem-solving strategies call for such models. Model-based methods are the name given to these procedures. On the other hand, there are methods that rely on learning rather than a model of the world. Model-free methods are the name given to these strategies. *Action:* with its actions, the agent can interact with and modify its environment. Either a discrete or continuous action space is possible. A discrete action space has a limited number of possible actions, whereas a continuous action space has an infinite number.

There is no one-size-fits-all method of problem-solving in RL; the formulation of a problem depends on its unique requirements and restrictions. In order to properly address RL problems, many formulations and strategies may be needed. Tasks with a start and finish (terminal state) are known as episodic tasks. The episodes are viewed as separate activities that require the agent to engage with the environment as it changes from one state to another. According to Sutton and Barto (2018b), this means that regardless of the agent's state in the previous episode, the entire procedure must be redone in order to start a new episode, and the agent must start from scratch.

In RL, an agent's primary goal is to achieve the greatest expected return, which is determined differently for each task. For episodic activities with a starting and an ending state, the expected return can be calculated using the equation below:

$$G_t = R_{t+1} + R_{t+2} + ... + R_{t+k} + ... + R_T, \tag{20}$$

where T is the last time step and G_t is the return at time t, which is the total of rewards accrued after time t. *continuous or non-episodic tasks* are tasks that are not episodic. They are non-terminal tasks that enable the interaction between the agent and environment to continue continuously.

Therefore, the return for a continual task can be evaluated using

$$G_t = R_{t+1} + R_{t+2} + ... + R_{t+k} + R_{t+k+1} + ... \ . \tag{21}$$

The return G_t equal ∞ when added together. An agent's goal is to achieve the greatest

expected return, which should not be an infinity as shown above but rather a value that is emphatically bounded. By incorporating a discounted rate $\gamma \in [0, 1]$, this scenario may be managed, and the return for a continuous problem is then evaluated using

$$G_t = R_{t+1} + \gamma^2 R_{t+2} + ... + \gamma^k R_{t+k} + \gamma^{k+1} R_{t+k+1} + ... \tag{22}$$

$$= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{23}$$

2.3 Markov Decision Process

Decision-making is the main topic of the discipline of reinforcement learning. The multi-armed bandit dilemma, in which an agent continuously chooses behaviors to maximize the rewards provided by the environment, is a fundamental issue in real-world learning. This problem is a simplified form of a Markov decision process, which is a mathematical framework for modeling and solving RL problems. It has only one state. A state is considered to have the Markov property when it can change into future states without being influenced by its past events and only uses the information available at the moment. If a state meets this requirement, it can be referred to as Markov.

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t] \tag{24}$$

A probability distribution function, $\mathbb{P}$ explains the likelihood of changing from one state to another, and the variable S_t represents the state at time t. The next state in the sequence only depends on the present state and not on the past states because a Markov process is a stochastic process without memory. It is modeled as a sequence of arbitrary states labeled as S_1, $S_2, \cdots$, satisfying the Markov property. A tuple (S,P) that represents a finite collection of states (S) and a state transition probability matrix (P), which quantifies the likelihood of changing from one state to another, defines a Markov process:

$$P_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s] \tag{25}$$

where $s, s' \in S$ and t is the time.

A Markov Decision Process (MDP) is a problem in reinforcement learning where the agent must make sequential decisions by acting and changing states in order to maximize rewards. In RL situations, actions not only influence current rewards but also future rewards through later states, MDPs offer a mathematical foundation for addressing such difficulties.

Mathematically, an MDP is a tuple (S, A, τ, R, γ) where S is the set of all possible states,

A is the set of all possible actions, τ is a transition function that defines the probability $\tau(s', s, a) = Pr(s'|s, a)$ of transitioning to state s' when the current state is s and action a is taken, R is a reward function that defines the reward $R(s, a)$, and $\gamma \in [0, 1]$ is a discount factor. Maximizing the overall reward the agent receives from the environment at each time step is its goal. This necessitates taking into account both short-term benefits and long-term cumulative rewards, giving rise to an essential RL concept known as the return.

MDPs can be used to define decision-making issues for both episodic and continuous tasks. In an episodic task, the agent-environment interaction can be separated into naturally occurring episodes, but in a continuous task, it never ends and there are no distinct episode borders. Because a continuous task is ongoing forever and the return G_t has the potential to grow infinitely large, maximizing the return becomes difficult.

We present the idea of discounted rewards to address this problem, where the agent chooses behaviors to maximize the total amount of discounted rewards it will receive in the future. The parameter γ in equation (22) represents the discount rate, which is a crucial factor in defining the present value of future rewards. Its value ranges from 0 to 1, and it helps to balance the importance of immediate rewards against those received in the future.

A Reinforcement Learning (RL) task requires an agent to use a variety of policies to direct it in maximizing the anticipated long-term payoff. A policy π maps states to probabilities for selecting each potential course of action. To put it another way, if we are in state $S_t = s$, the chance that the agent would choose action $A_t = a$ while employing policy π at time t is given by $\pi(a|s)$.

RL methods estimate a value function, which can either be a function of states or a function of state-action pairings, to complete the decision-making process. Since the order of rewards is random, both methods involve altering the agent's policy based on its experience in estimating the values.

Model-free learning and model-based learning are the two basic computing techniques that make up RL. In model-free learning, the agent assigns a value to each action without having access to the reward function or transition probability matrix. Through the use of algorithms like Q-learning and policy optimization, the agent gains knowledge through experience. A model of the environment is built by the agent via model-based learning, which gives it access to the transition probability matrix, reward function, state space, and action space. Model-based learning uses dynamic programming, which is used in Atari video games as one example. To comprehend how agents can create good or nearly optimum policies without being aware of the model of the environment, we will concentrate on model-free methods like the Monte Carlo approach and the temporal difference method.

Model-free Algorithms

These techniques enable the learning of an optimal value function or policy without the need for an environment model. The Monte Carlo Method and the Temporal Difference Method are two instances of what are known as model-free algorithms. In contrast to Dynamic Programming (DP), which uses predicted mean, the MC Method predicts the value function using empirical mean return (state or state-action pair).
By gathering a sample of episodes of experience produced by a policy π, indicated as

$$S_1, A_1, R_2, ..., S_k \sim \pi$$

The MC Method solves both the prediction problem (predicting the value function given a policy) and the control problem (identifying the ideal value function and policy).
The state value function $v(s)$ or state-action value function $q(s, a)$ is then updated using either the First-visit (checking if a specific state is first visited in an episode) or Every-visit (updating each time step t a particular state is visited in an episode) methods.
Temporal Difference (TD) learning is a model-free algorithm commonly used in solving reinforcement learning problems. TD can learn directly from incomplete episodes of experience without requiring knowledge of MDP transitions and rewards. Unlike the Monte Carlo (MC) method that requires a complete episode of experience to estimate the return G_t, TD uses bootstrapping with a target of $R_{t+1} + \gamma v(s_{t+1})$, where γ is the discount factor and $v(s_{t+1})$ is the estimated value of the next state, instead of using G_t.
Two important TD algorithms used for the control problem are SARSA (State-Action-Reward-State-Action) and Q-learning TD algorithms. SARSA is an on-policy TD control algorithm that learns an action-value function instead of a state-value function. This algorithm considers the transition from state-action pair to state-action pair and estimates the values of state-action pairs by learning $q_\pi(s, a)$ for all states s and actions a with respect to the current behavior policy π, while simultaneously changing the policy π towards greediness with respect to q_π.
The SARSA algorithm is so-named because it takes into account each element of the quintuple of events $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, which corresponds to the current state S_t, current action A_t, reward R_{t+1}, new state S_{t+1}, and future action A_{t+1}. The values of the state-action pairs are updated using:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big]$$

The Q-Learning algorithm is another TD control method used to learn the action-value function, but unlike SARSA, it directly approximates the optimal action-value function, q_*,

regardless of the policy being followed. This means that the Q-Learning algorithm does not need to estimate the value of the next state-action pair based on the current behavior policy. To update the action-value function in Q-Learning, the following formula is used:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

2.4 Deep Learning

Machine learning techniques called deep learning use neural networks with three or more layers. Layers are used to process data, and each layer gradually extracts information before sending them to the layer below. In order to provide a more thorough representation, later layers integrate the low-level information that the previous levels have extracted. Deep learning can be conceptualized mathematically as a function $f : X \to Y$ parameterized by $\theta \in \mathbb{R}^{n_\theta}$ where $n_\theta \in \mathbb{N}$, then

$$y = f(\mathbf{x} : \theta) \tag{26}$$

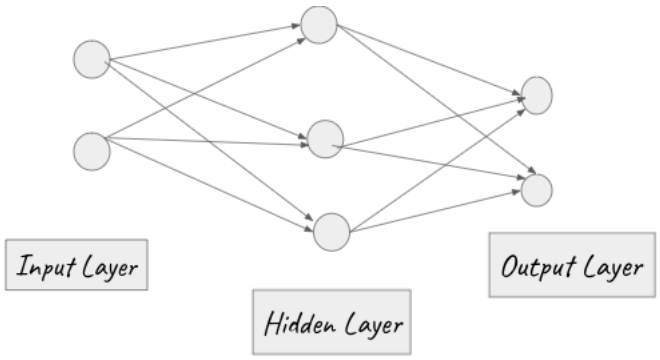


Figure 5: Example of a neural network with one hidden layer

A neural network with a fully connected layer that accepts input values x in column vector form with a size of $m \in \mathbb{N}$ is shown in Figure (5). The input feature is transformed nonlinearly by a parametric function W_1 of size $k \times m$ and a bias term b_1 of size k, followed by another nonlinear transformation, to produce the values of the hidden layer.

$$h = A(W_1 \cdot \mathbf{x} + b_1) \tag{27}$$

The activation function, represented by A, is included in the equation above. This procedure is crucial in neural networks because it translates inputs into a single output for processing in following layers by performing a non-linear modification at each neuron. We will briefly discuss a few of the most popular activation function types used in deep learning, of which

there are numerous varieties.

ReLU Function

The rectified linear unit (ReLU), which performs well on a variety of prediction tasks and is simple to construct, is a common deep learning tool. The maximum of an element $t = W_1 \cdot x + b_1$ and 0 defines ReLU, a straightforward nonlinear transformation.

$$ReLU(t) = \max(t, 0). \tag{28}$$

Sigmoid Function

The sigmoid function is an activation function that converts inputs from the $\mathbb{R}$ domain to outputs that lie within the interval (0, 1). Any input value between $(-\infty, \infty)$ is compressed to a value between those two ranges (0, 1).

$$sigmoid(t) = \frac{1}{1 + exp(-t)} \tag{29}$$

When the output of neural networks is to be interpreted as probabilities for binary classification tasks, sigmoid activation functions are frequently used. Additionally, a specific version of the softmax function is the sigmoid function.

Softmax Function

The softmax function is widely used to return the likelihood of each class in multi-class classification tasks. It converts a vector of numerical input values ($\mathbb{R}$) into a probability distribution, making certain that the total probability equals 1. The softmax function scales the input values so that they fall between 0 and 1, even if the input values may be in the range of $(-\infty, \infty)$. The following equation gives the definition of the softmax activation function:

$$softmax(t_i) = \frac{exp(t_i)}{\sum_j exp(t_j)} \tag{30}$$

Either the components of the input vector t or the outputs from the neurons in the output layer correspond to the values t_i. A huge number of parameters in deep learning must be learned from a given dataset. By minimizing a specified loss function that assesses how well the network performed on the dataset, these parameters are learned. Minimizing the loss function is a crucial step in training a neural network, although it can be difficult in reality.

Gradient Descent

The process of optimization involves minimizing a function, often referred to as the objective function. In many cases, optimization algorithms are designed to minimize the objective function. However, there are scenarios where it may be necessary to maximize the objective function by flipping its sign. Deep learning revolves around finding the most suitable model from a limited dataset. The primary goal is to minimize the loss function, which serves as the objective function for the optimization algorithm. This is achieved by training the model using a specific set of training data. Through this training process, the deep learning model gradually reduces the training error, which represents the discrepancy between the predicted output and the actual output. Ultimately, the ultimate objective is to minimize the generalization error, which reflects the difference between the anticipated output on unseen data and the actual output. One of the commonly employed optimization algorithms in deep learning is gradient descent. The primary objective of gradient descent is to locate a stationary point where the derivative of the loss function becomes zero, enabling the minimization of the loss function. This is achieved by starting from any location and iteratively moving in the direction opposite to the negative gradient (the vector of partial derivatives of the loss function with respect to input variables) until reaching a stationary position. Gradient descent, often referred to as GD, is employed to address optimization challenges in deep learning. Its purpose is to determine a collection of parameters (**W,b**) that minimizes the objective function and yields favorable results on the training set, given a deep learning model, an objective or cost function $J(\mathbf{W,b})$, and a training dataset D.

In deep learning optimization, the fundamental concept is to initialize weight and bias parameters with random values and then iteratively update them to minimize the objective function $J(\mathbf{W,b})$. The process of gradient descent is repeated until convergence is achieved. This procedure involves calculating the gradient (a vector of partial derivatives) of the objective function at a specific location and adjusting the parameters by moving in the direction of the steepest descent, aiming to reach a minimum point.

$$\mathbf{W} = \mathbf{W} - \alpha \frac{d}{d\mathbf{W}} J(\mathbf{W}) \tag{31}$$

$$\mathbf{b} = \mathbf{b} - \alpha \frac{d}{d\mathbf{b}} J(\mathbf{b}) \tag{32}$$

where

- α is the learning rate
- $-\frac{d}{d\mathbf{W}} J(\mathbf{W})$ is the search direction for the stationary point.

2.4.1 Forward Propagation

Forward propagation is the process of carrying out the computations of a neural network starting from the input layer and proceeding towards the output layer (also known as forward pass). Consider a neural network that uses sequential operations and a single hidden layer. Suppose that $\mathbf{x} \in \mathbb{R}^d$ is the input example and that there is no bias term in the hidden layer. The intermediate variable can then be written as follows:

$$\mathbf{z}_1 = \mathbf{W}^{(1)}\mathbf{x} \tag{33}$$

The input example $\mathbf{x} \in \mathbb{R}^d$ is converted into an intermediate variable $\mathbf{z}_1 \in \mathbb{R}^h$ using the weight parameter of the hidden layer, indicated by $\mathbf{W}^{(1)} \in \mathbb{R}^h \times d$. The hidden activation vector, which has a length of h, is produced by passing this intermediate variable through the activation function $\mathbf{A}$.

$$\mathbf{h} = \mathbf{A}(\mathbf{z}_1) \tag{34}$$

The hidden layer output $\mathbf{h}$, an intermediate variable, is processed to produce the output layer variable of a neural network. The output layer variable is created as a vector of length q assuming that just the weight parameter of the output layer exists, denoted as $\mathbf{W}^{(2)} \in \mathbb{R}^{q \times h}$.

$$\mathbf{z}_2 = \mathbf{W}^{(2)}\mathbf{h} \tag{35}$$

Depending on the type of problem and a certain activation function, $\mathbf{G}$ would produce the desired result.

$$\hat{\mathbf{y}} = \mathbf{G}(\mathbf{z}_2) \tag{36}$$

The result generated by the deep learning model is denoted by $\hat{\mathbf{y}}$.

2.4.2 Loss/Cost Function

The difference between the predicted result and the expected result is measured using a loss function to assess the performance of a neural network. The loss function can be used to determine the gradients needed to update the weights. The average of all losses is computed in order to obtain the cost function. Deep learning employs a variety of loss function types, some of which are used frequently as follows:

Mean Squared Error

When the expected and predicted outcomes are continuous numerical values, the mean squared error is frequently used. The squared difference between the predicted value $\hat{\mathbf{y}}$ and the expected

value $\mathbf{y}$ is used to calculate the loss function for this.

$$L = (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2 \tag{37}$$

and the cost is calculated as the average over all losses for the individual examples

$$J = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2 \tag{38}$$

Binary Cross-Entropy

A measure of the dissimilarity between two probability distributions is the cross-entropy, which is the negative log-likelihood. When there are two alternative outcomes, binary cross-entropy is frequently used. We consider the expected outcome $\mathbf{y}$ and the predicted outcome $\hat{\mathbf{y}}$ to calculate the loss function for binary cross-entropy, and the formula is provided below:

$$L_b = -(\mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i)) \tag{39}$$

and the cost function is given as

$$J = -\frac{1}{N}\sum_{i=1}^{N} L_b \tag{40}$$

where N is the total number of examples we are considering.

Categorical Cross-Entropy

A loss function is used to evaluate the difference between two probability distributions, especially when there are numerous classes to which an example could belong and the model needs to choose the best one. This is done using the categorical cross-entropy loss function, which combines the formula below to quantify the loss of an example:

$$L_c = -\sum_{i=1}^{M} \mathbf{y}_i \cdot log(\hat{\mathbf{y}}_i) \tag{41}$$

Based on the target value y_i that corresponds to the $i-$th scalar value in the model output $\hat{\mathbf{y}}_i$, the number of scalar values in the model output M, and the loss of an example, this equation estimates the loss.

$$J = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} \mathbf{y}_{ij} \cdot \log(\hat{\mathbf{y}}_{ij}) \tag{42}$$

2.4.3 Backward Propagation

A neural network technique for computing the gradient of its parameters is back-propagation. By caching intermediate variables required to calculate the gradient with respect to certain parameters, it uses the chain rule in calculus to send information from the output layer to the input layer in reverse order. In order to improve the generalization of the model and reduce the error rate, the goal of training a neural network is to update the weights based on the error rate from the previous iteration.

The generalization and error rates of the model can be enhanced, and their rates can be minimized, increasing the model's dependability. In order to compute the gradient with regard to the parameters during the back-propagation technique, intermediate variables or partial derivatives are kept.

The goal of back-propagation in the previous example was to compute the gradients $\frac{\partial J}{\partial W^{(1)}}$ and $\frac{\partial J}{\partial W^{(2)}}$ given the two parameters $W^{(1)}$ and $W^{(2)}$. The chain rule is used to achieve this, and one intermediate variable's and parameter's gradients are computed individually. Since we must start from the output of the computational graph and work backward towards the parameters, the calculations are carried out in the reverse order of forward propagation. Here is an explanation of the back-propagation procedure using the aforementioned example.

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial \mathbf{z}_2}{\partial W^{(2)}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{z}_2} \cdot \frac{\partial J}{\partial \hat{y}} \tag{43}$$

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial \mathbf{z}_1}{\partial W^{(1)}} \cdot \frac{\partial h}{\partial \mathbf{z}_1} \cdot \frac{\partial J}{\partial h} \tag{44}$$

2.5 Deep Reinforcement Learning

Deep reinforcement learning is a method that combines the research of deep learning and reinforcement learning. It can be divided into three types: model-based, value-based, and policy-based, and entails utilizing a neural network to approximate the value function, policy, or model. A value function that may be utilized to construct a policy is what value-based strategies seek to create. Within this category, while the deep Q-network (DQN) algorithm uses neural networks as function approximators, Q-learning and its variant fitted Q-algorithms use parameterized function approximators. This latter program has outperformed humans in its ability to master many ATARI games from pixel input. We will quickly go through DQN and some of its upgrades.

2.5.1 Deep Q-networks

The Deep Q-network (DQN) technique was first introduced in 2015 by (Mnih et al., 2015), which displayed outstanding performance on numerous ATARI games in an online environment by learning from raw pixel data. Due to its capacity to handle challenging sequential decision-making issues, reinforcement learning has grown in popularity recently. The DQN approach combines deep neural networks and off-policy Q-learning in a model-free way so that the network may automatically extract pertinent information from input data. A single forward pass can choose the best action for a given state using the DQN's probability distribution across all feasible discrete actions.The algorithm's use of experienced replay and frozen target network methods, as described in the next section, helps to prevent unstable learning due to correlated input/output from occurring.

DQN Experience Replay

The DQN algorithm utilizes experience replay to mitigate the problem of high correlation between actions and states during the learning phase. This technique helps stabilize the network's weights, preventing divergence or oscillation. This entails putting an experience dataset into a memory space that has room for N experiences. The current state, the action made in that state, the reward obtained, and the ensuing subsequent state make up each experience. A batch of events is randomly chosen from the memory space during training and utilized to update the network. To guarantee that the network receives independent data, the DQN technique accomplishes this in order to eliminate correlations in the data sequences. Additionally, periodically reliving earlier experiences aids in adjusting for any shifts in the data distribution. Equation (45), which defines the loss function used by the DQN algorithm, minimizes the squared temporal-difference error when updating the network.

$$L_i(\theta_i) = \mathbb{E}\big[(R_{t+1} + \gamma \max_a Q(S_{t+1}, a, \theta_i^*) - Q(S_t, A_t, \theta_i))^2\big] \tag{45}$$

Frozen Target Network

Equation (45) introduces the frozen target network method, also utilized by the DQN algorithm. The Q-network and the target network each comprise two networks with similar architecture but differing weights. Every K time steps, the target network's parameters are synchronized with those of the Q-network, while a continuous update based on the loss function from equation (45) is done by the Q-network. The target network's weights are maintained constant for K time steps and changing policies are smoothed out, leading to more stable learning.

Algorithm 1 Deep Q-Network (DQN) Algorithm

procedure DQN(parameters)
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target-action value function $\hat{Q}$ with weights $\theta^* = \theta$
for episode 1 to M **do**
for $t = 1$ to T **do**
Using a good action selection method like epsilon-greedy; select a random action a_t with probability or select $a_t = \arg\max_a Q(s, a; \theta)$
Execute action a_t in the environment and observe reward r_t and the next state s_{t+1}
Store the transition (s_t, a_t, r_t, s_{t+1}) in D
sample random minibatch of transitions (s_t, a_t, r_t, s_{t+1}) from D
Set output $y_k = \begin{cases} r_k & \text{if episode terminates at } j+1 \\ r_k + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \theta^*) & \text{otherwise} \end{cases}$
Perform a gradient descent step on $(y_k - Q(s_k, a_k; \theta))^2$ with respect to the network parameter θ
After some time steps K, reset $\hat{Q} = Q$
end for
end for
end procedure

2.5.2 Optimization Techniques for DQN

Advanced optimization techniques have been developed for the Deep Q-Network (DQN) algorithm to train agents to make optimal decisions in complex environments, such as in video games where the goal is to attain the highest achievable score. These techniques use sophisticated mathematical algorithms to determine the best action to take at each step of the game, taking into account the game's complexity and multitude of choices. Put simply, optimization techniques for DQN empower agents to learn and make decisions that lead to the best possible outcomes in challenging and intricate environments.

Replay Memory

During the exploration of the environment, our agent takes actions, but the neural network is not trained continuously. Rather, all the agent's experiences are accumulated and saved in a memory. Subsequently, the model is trained by selecting samples from the stored experiences, either randomly or in sequence. This approach is based on the notion that while neural networks can recognize variations in the environment, they may suffer from overfitting and not perform effectively in unfamiliar scenarios.

Target Network

Deep Q Learning employs two neural networks instead of one to improve performance and address the issue of moving targets (the difficulty in training a neural network to precisely approximate specific values (e.g., Q-values) while the network's parameters continuously change throughout the learning process). Although not mandatory, using two networks is beneficial. When an agent utilizes the same neural network to compute both Q values and target values, it may result in moving targets. To circumvent this problem, a separate neural network, known as the target network, with fixed parameters that are regularly updated is used.

Action Selection

When selecting actions, the agent must balance exploration and exploitation. Various methods, such as softmax and epsilon-greedy, can be used for this purpose. Softmax action selection is the preferred method because it automatically stops exploration after a predetermined period of time.

To summarize, the DQN algorithm operates in the following manner: the agent perceives the current state of the environment. If a randomly generated integer is less than or equal to epsilon, the agent acts randomly; otherwise, DQN predicts the Q values and selects the action with the highest Q value. The agent uses a short-term data storage to store a terminal variable, the next state, and reward. Once enough samples are collected in memory, the agent trains the DQN by selecting a batch of experiences. The set of current states is used as features, and the target values are the labels. The main network is periodically updated with the target network.

2.5.3 Double DQN

The Double Q-learning strategy, which solves the issue of overestimation by dividing the maximum operation in the target into two stages: action selection and action assessment, is the origin of the Double DQN approach. The same values are utilized for selecting and assessing actions in classical Q-learning and DQN, which may lead to selecting inflated values and hence too optimistic value estimates. By separating the selection and assessment processes, double Q-learning solves this issue of overestimation by separating the maximum operation in the target into two stages: action selection and action evaluation. The same values are utilized for selecting and assessing actions in classical Q-learning and DQN, which may lead to selecting inflated values and hence too optimistic value estimates. By separating the selection and assessment processes, double Q-learning solves this issue. Double Q-learning specifically entails learning two value functions by randomly selecting one of two value functions to update

for each experience, resulting in two sets of weights indicated by θ and $\theta*$. The greedy policy is determined by one set of weights at each update, and its value is determined by the other set. The aim in Double Q-learning is defined as follows in order to further delineate the selection and evaluation processes in Q-learning:

$$Target(Y) = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_{a} Q(S_{t+1}, a, \theta_i), \theta_i^*) \tag{46}$$

Double DQN addresses the problem of overestimating Q-values, which is common during the early learning stages when incorrect actions may have an inflated value. To tackle this issue, Double DQN utilizes two distinct sets of parameters, denoted as θ and θ^*, to calculate the TD-target. This approach fosters more stable learning and helps prevent the overestimation of Q-values. By replacing the loss function from equation (45) with equation (47), Double DQN can easily accommodate this modification, leveraging its existing two networks.

$$L_i(\theta_i) = \mathbb{E}\big[(R_{t+1} + \gamma Q(S_{t+1}, \arg\max_{a} Q(S_{t+1}, a, \theta_i), \theta_i^*) - Q(S_t, A_t, \theta_i))^2\big] \tag{47}$$

To sum up, the Double DQN algorithm uses two neural networks, one that is continuously updated, called the "online" Q Network, and another that is periodically updated, called the "target" Q Network. By running an argmax over all the state-action values, the online Q Network is used to determine the the cost of every action that might be taken in a particular state. The target value for the action that maximizes value is then determined by using the target Q Network, and the action is chosen. This process is used to prevent over-estimation of Q-values and stabilize the target values.

The Double DQN is an improvement on DQN. (Van Hasselt et al., 2016) showed that DQN suffers from substantial overestimations of Q-values for some actions in some games in the Atari 2600 domain, and was able to use the double DQN algorithm to not only solve the overestimation problem faced by DQN but was able to show better performance on several games compared to DQN. Robotic vehicle navigation using double DQN-based control was used to navigate paths with steep turns on their own. Unexpectedly, this method considerably decreased the amount of time it takes for the car to stabilize and reduced overshooting at turns, especially at greater forward speeds. A little trade-off resulted in a tiny boost in the rising time and steady-state error, though according to (Zhang et al., 2019).

2.5.4 Actor-Critic

In the Actor-Critic method (a policy gradient approach) in which the value function $v(s)$ is employed to train and update the policy parameters denoted as θ. The critic computes the expected value for a given input state, reflecting the value function $v(s)$, whereas the actor reflects the present policy and given an input state s, produces an action a. Both networks frequently update the TD-Error. The assessment of the predicted outcomes of the current and incoming states by the Critic, which contribute to the TD-Error, influences the update of the Actor. Figure (6) provides an illustration of this method.
Algorithm (2) shows the step-by-step process of (Advantage) Actor-Critic algorithm with the factor multiplying the log-likelihood of the policy

$$A^{\pi}(s,a) = R^{t}(s,a) - V^{\pi}(s) \tag{48}$$

in line 11 is known as the advantage of the action a in s, which adjusts the agent's action magnitude, with both positive and negative values influencing encouragement or discouragement of the chosen action and this can appear in other two forms as follows

- **TD-Error or TD advantage estimate**

$$A^{\pi}(s,a) = r(s,a,s') + \gamma V^{\pi}(s') - V^{\pi}(s) \tag{49}$$

- **n-step advantage estimate**

$$A^{\pi}(s,a) = \sum_{k=0}^{n-1} \gamma^{k} r_{t+k+1} + \gamma V^{\pi}(s_{t+n+1}) - V^{\pi}(s_t) \tag{50}$$

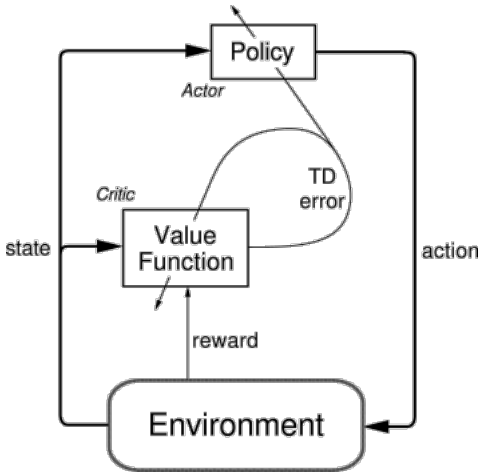


Figure 6: Actor-Critic Architecture (Sutton and Barto)

Algorithm 2 Advantage Actor-Critic Algorithm

procedure ACTOR-CRITIC(neural net)
    Initialize neural network parameterization of policy $\pi(\omega)$
    Initialize neural network parameterization of $V(\theta)$
    while True **do**
        Initialize θ, ω, $V(\text{terminal}, \theta) = 0$
        Initialize s_0
        Observe an episode $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T, s_T$ following $\pi_\omega(\cdot)$
        $R_T = 0$
        for $t = T$ to 0 **do**
            $R_{t-1} = r_t + \gamma V(s_t, \theta)$
            $\nabla J = \nabla J + \gamma^t (R^t - V(s_t, \theta)) \nabla_\gamma \log \pi(a_t, s_t, \omega)$
            $\nabla L = \nabla L + \gamma^t \nabla_\theta (R^t - V(s_t, \theta))^2$
            $\omega = \omega + \alpha \nabla J$
            $\theta = \theta + \beta \nabla L$
        end for
    end while
    return Policy and value function parameters ω, θ
end procedure

RL has made significant advancements in generalization, representation, and efficiency on both theoretical and technical levels, making it increasingly applicable to real-world issues across a variety of fields. Some of the applications encompass simulation training, supply chain optimization, autonomous vehicles, data analysis, and even art. To date, RL methods have been successfully used in a number of healthcare sectors. In a research by Yu et al. (2021), they categorized the application domains into three categories: personalized treatment strategies, automated medical diagnosis, and other general domains such as patient care optimization and scheduling

2.6 RL's Application to Healthcare

However, when it comes to RL's application to healthcare, some researchers have raised concerns about the reliability of policies suggested by RL algorithms. This is because we are dealing with human beings who may only tolerate minimal trial and error, unlike most RL algorithms.

In (Yanez et al., 2019), the authors focus on the design of settings for disease modeling, state representations, intervention strategies, and reward functions. They present the problem of selecting optimal disease interventions as an RL problem. Before adapting an epidemiological model to fit into an RL environment, it is essential to first develop a disease model that is naturally suited for analyzing different control policies. This ensures that various control policies can be analyzed on epidemiological models of the disease using RL. The models are created as a virtual or simulated epidemic/pandemic situation that is closely related to

the disease, which is referred to as the environment in RL. In simple terms, the learning agent interacts with the environment (modified epidemiological model with Markov decision processes) to identify good or close to the best control policies to decrease disease spread in the shortest possible period. Such epidemiological models can be seen in (Colas et al., 2021; Reymond et al., 2022b; Bent et al., 2018; Khadilkar et al., 2020; Kompella et al., 2020), and in order to discover the best timing for HIV testing and retention in care rates at 5-year intervals between 2015 and 2070, Khatami and Gopalappa (2021) used RL. In this work, we are interested in Coronavirus disease (COVID-19) and how RL can be applied on a number of NPIs to reduce the spread of the disease. COVID-19, an infectious disease caused by the SARS-CoV-2 virus since its detection in 2019 has seen quite a number of NPIs, the common ones being travel bans, curfews, social distancing, bans of social gathering, face masks, increased hygiene, remote working, school closures, and lock-downs as mentioned in (Perra, 2021).

There are various instances where Reinforcement Learning (RL) has proved to be beneficial in the healthcare field, such as cancer treatment design and reduction of malaria spread. (Zhao et al., 2009) used a temporal-difference learning method called Qlearning with approximation functions like Support Vector Regression (SVR) and Extremely Randomized Trees (ERT) to identify optimal policies for a specific cancer treatment. (Bent et al., 2018) used agent-based methods such the Genetic algorithm, the Upper/Lower Confidence bound algorithm, and Batch Policy Gradient to formulate the problem of decreasing the spread of malaria as a stochastic multi-armed bandit problem. They suggested good policies that can reduce the spread of malaria for a targeted population.

However, the traditional RL approach has limitations in handling sequential decision problems with complex state and action spaces, which led researchers to develop a new approach called Deep Reinforcement Learning (deep RL). (Mnih et al., 2015) In their pioneering work, Mnih et al. (2015) introduced the deep Q-network (DQN), a revolutionary agent that integrates deep neural networks with reinforcement learning. This innovation allows the DQN to handle continuous high-dimensional state and action spaces with remarkable efficiency Additionally, Libin et al. (2020) investigated the use of deep RL to learn prevention strategies for reducing the spread of a viral outbreak. To predict when to close and reopen schools, as well as the relevant contact matrices for school breaks and terms, they used the Proximal Policy Optimization deep RL algorithm on a stochastic age-structured SEIR model with 379 patches, each patch representing an administrative region in Great Britain.

In (Kwak et al., 2021), deep RL algorithms like Dueling Double Deep Q-Network (D3QN) and Double DQN were utilized to let agents look for public health measures to stop the COVID-19 virus's proliferation. The study was only concerned with benefits to population health, ignoring

the negative social and economic consequences. The Susceptible-Infectious-Recovered-Dead (SIRD) model served as the foundation for the epidemiological model under consideration. SIRD simulates scenarios to determine how reinforcement learning can reduce the burden of COVID-19.

(Kompella et al., 2020) utilized RL algorithm (discrete-action Soft Actor Critic) to optimize mitigation policies that minimized the economic impact while not exceeding the capacity of the hospital. They suggested a new agent-based pandemic simulator which implements a modified SEIR (susceptible, exposed, infected, recovered) infection model that can represent fine-grained interactions between people in specific areas in a community. (Kumar et al., 2021) compared the prediction efficiency of deep RL against a modified Long Short-Term Memory (LSTM) and Logistic Regression (LR) model to predict likely COVID-19 cases and deaths, the count of newly affected individuals, losses and cures in the next days. In terms of error rate, the deep RL algorithms used outperformed the LSTM and LR model. They further used deep RL to optimize results based on COVID-19 symptoms.

Colas et al. (2021) introduced EpidemiOptim, a Python toolbox that generate an optimization problem from the combination of epidemiological models and cost functions via OpenAI Gym. They applied RL based algorithm, DQN, and an evolutionary based algorithm, NSGAII, on a SEIR model for COVID-19 to find optimal policies for dynamical on-off lockdown control under the optimization of death toll and economic recess. The epidemiological model considered here is a python implementation of an extended SEIR model fitted to French data.

Deep multi-objective RL and a cutting-edge technique called Pareto Conditioned Networks (PCN)(Reymond et al., 2022b) was used in (Reymond et al., 2022a) to mitigate the spread of COVID-19 by training a set of decisions that estimates the Pareto front of the decision problem. In order to reduce COVID-19 situation (like hospitalizations and infections) and societal burden, they looked at exit strategies in Belgium after the initial SARS-CoV-2 epidemic wave using a modified version of the SEIR compartmental model (discrete-time stochastic model that incorporates an age-structured population).

2.7 Multi-objective Optimization

Making decisions and choices is an inevitable part of life, and everyone desires to make them as good as possible. In the past, decisions were usually based on intuition, common sense, or chance. However, in today's world, mathematical modeling and programming provide solutions to everyday judgments and compromises. In multi-objective optimization problems, conflicting objectives arise that cannot be solved by a single optimal solution. These problems are prevalent in various fields, such as health, robots, bridges, planning and pricing production systems, and wireless sensors.

Multiple objective functions are measured in various units, such as time and money, which typically compete and clash with one another. For instance, in the medical industry, optimizing the drug's potency necessitates taking into account negative side effects and synthesis costs according to (van der Horst et al., 2012; Rosenthal and Borschbach, 2017). As a result, Pareto-optimal solutions are a class of optimal solutions that come from multi-objective optimization problems. In such circumstances, there is no solution that can be deemed superior to another in terms of all objectives.

Comparing the objective function values of different solutions in single-objective optimization issues makes it simple to assess which option is best. The goodness of a solution in multi-objective optimization problems is instead decided by dominance. If there are two solutions to certain objective functions, x_1 and x_2, we say that x_1 dominates x_2 if it is strictly better in at least one goal and equal to or better than x_2 in all objectives.

The following are mathematical formulations for multi-objective optimization problems having a number of objectives and a number of equality and inequality constraints.

$$\begin{aligned} &\max / \min \quad f_i(\mathbf{x}) \quad i = 1, ...N_{objectives} \\ &\text{subject to} \quad g_k(\mathbf{x}) = 0 \quad k = 1, ..., K \\ &\qquad\qquad\quad h_l(\mathbf{x}) \leq 0 \quad l = 1, ..., L \end{aligned} \tag{51}$$

where

- $f_i(\mathbf{x})$ is the objective functions.
- $\mathbf{x}$ is a decision vector that represents a solution.
- $N_{objectives}$ is the number of objectives.
- K and L are the numbers of equality and inequality constraints respectively.

To tackle multi-objective problems, there are two approaches as described by Dey and Choudhury (2016). The first method includes turning a multi-objective problem into a single-objective problem using a *scalarization function*. The disadvantage of this method is that when the scalarization function is non-linear, the transformation could not be valid. Weights are used in the scalarization strategy to aggregate various objective functions into a single solution. There are different types of weights such as equal weights, rank-order centroid weights, and rank-sum weights. For instance, two objective functions (obj_1 and obj_2) can be transformed into a single objective function (O) through scaling and transformation, as shown below

$$O = \Big((1 - \lambda) * \mathrm{obj}_1\Big) + \Big(\lambda * \mathrm{obj}_2\Big) \tag{52}$$

where the weight $\lambda \in [0, 1)$. The scalarization function is a traditional multi-objective optimization technique that combines each objective that has already been pre-multiplied by a weight to create a single objective. The weighted sum technique is a frequent name for this strategy, which selects an objective's weight in proportion to its relative relevance. Obtaining a Pareto-optimal solution in the objective space is challenging. Some research studies, like the work of (Syswerda, 1991), have used weight vectors in the fitness function to adjust values during resource scheduling evaluations based on penalties. The difficulty of swiftly and precisely moving the working position of a robotic system to a designated position was addressed by the authors in (Jakob et al., 1992) in the context of job planning. They used a weighted sum method that took into consideration many goals to accomplish this. Their strategy stressed creating a path for the robot to go along that was both smooth and effective in addition to focusing on avoiding obstacles. Another study by (Jones et al., 1993) implemented weighted genetic operators in a Genetic Algorithm (GA) to produce hyper-structures from a collection of crystalline formations. Wilson and Macleod (1993) included this strategy into a GA to create IIR filters without multipliers in an effort to reduce the filter's response inaccuracy and implementation cost. Another MOO method, the $\epsilon-$ Constraint method (Chircop and Zammit-Mangion, 2013), restricts all objectives except one to user-specific values. This approach can be used to solve convex or non-convex problems, but it is important to choose the ϵ vector carefully so that it lies between the minimum and maximum values of each objective function. In order to tackle multi-objective optimization issues while reducing computing cost, Quagliarella (1998) proposed connection this technique with a hybrid GA.This method was employed by Ranjithan et al. (1992) to address groundwater pollution containment issues. Goal programming method have been instrumental in addressing MOO challenges in the industrial context and its development is given credit to Charnes and Cooper (Charnes and Cooper, 1957) and Ijiri (Ijiri, 1965) who developed the method for a linear model. The decision-maker in this technique must establish targets or goals for each purpose. These values are introduced to the problem as more restrictions. After that, the goal function will attempt to reduce the absolute discrepancies between the targets and the true values.

The simplest formulation of this strategy is as follows:

$$\min \sum_{i=1}^{k} |f_i(\bar{x}) - T_i|, \text{ subject to } \bar{x} \in \mathcal{F} \tag{53}$$

where the decision-maker's aim or goal for the ith objective function, $f_i(\bar{x})$, is denoted by the letter T_i, and the feasible region is denoted by the letter $\mathcal{F}$. Therefore, the goal is to lessen the total absolute value of the disparities between the desired and actual values. Unless prior knowledge of the shape of the search space is available, this strategy's disadvantage is that

the decision-maker is given a challenging task of choosing the proper weights or preferences for the goals that will remove the problem's inherent differences. To enhance the brightness of six atomic spectral lines for minor constituents, while considering specific experimental conditions, (Wienke et al., 1992) integrated this strategy with a genetic algorithm. To optimize two-dimensional trusses and create a two-dimensional mechanism, Adeli (1994) also employed goal programming in conjunction with a genetic algorithm. The second approach involves using *multi-policy algorithms* or evolutionary multi-objective optimization (EMO) algorithms (Deb et al., 2002) and SPEA2 (Zitzler et al., 2003) which create a group of multi-objective solutions instead of focusing on just one solution at a time such as Pareto approach and NSGA-II respectively. The "Pareto method" is another name for these algorithms, which make the MOOP simpler by avoiding the need of challenging mathematical equations (more on this will be covered later). The algorithms employ a population-based methodology, where multiple solutions take part in each generation, and a new set of solutions is evolved after each generation. According to (Deb, 2011), this technique is effective since it doesn't require any derivative information, is comparatively simple to apply, is flexible, and has a wide range of applications. Choosing the best Pareto-optimal solution from a list of potential solutions is the challenging element of multi-objective optimization issues. In a single simulation run, the approach can uncover numerous distinct solutions that represent trade-offs between various criteria and do not dominate one another. Therefore, it becomes clear that the best method to complete the initial work is to use a Pareto-based strategy, such as an evolutionary algorithm. In our case we are using reinforcement learning to solve multi-objective problems.

2.8 Multi-objective Reinforcement Learning

In RL, Multi-objective reinforcement learning (MORL) is a framework that considers multiple reward signals, each corresponding to a specific objective. The objective of MORL is to learn policies that can optimize multiple criteria simultaneously. Unlike single-objective RL where a sole optimal solution is sought, in MORL, the aim is to identify a collection of trade-off solutions that strike a balance between the objectives. The objective is to obtain a collection of top-performing trade-off solutions that Pareto dominate all other solutions, while being incomparable to each other.

In MORL, regular MDPs are generalized to multi-objective markov decision processess or multiobjective markov decision processes (MOMDPs). As opposed to scalar rewards, MOMDPs provides a vector of rewards, i.e.,

$$R(s,a) = (R_1(s,a), ..., R_n(s,a)), \tag{54}$$

In the above equation, n signifies the number of objectives being considered. When it comes to MORL, the value function dependent on the state associated with a state $\mathbf{s}$ takes the form of a vector.

$$V^{\pi}(s) = (V_1^{\pi}(s), V_2^{\pi}(s), \cdots V_n^{\pi}(s)) \tag{55}$$

Due to the presence of multiple objectives in the environment, it is possible for different policies to be optimal for different objectives. To determine optimality in multi-objective optimization, the Pareto dominance relation is commonly utilized, as demonstrated in (Van Moffaert and Nowé, 2014).

The authors in (Mossalam et al., 2016) proposed Deep Optimistic Linear Support Learning (DOL) to tackle multi-objective choice problems with high dimensionality, especially when the relative importance of the objectives is uncertain. DOL leverages high-dimensional input features to compute the convex coverage set, which encompasses all possible optimal solutions that are a convex combination of the objectives.

The authors in (Xu et al., 2020b) introduced an evolutionary learning algorithm called Prediction-Guided MORL algorithm to address control problems with competing goals. The objective is to come up with a collection of control measures that are suitable for various objective desires, known as the Pareto-optimal set. To achieve this, they extended a popular reinforcement learning algorithm, Proximal Policy Optimization (PPO), and showed that optimum performance trade-offs for multi-objective control of robots can be achieved by representing the optimal policies as a Pareto set, which comprises of different policy groups. Their approach was applied to continuous robot control problems, and they were able to demonstrate the effectiveness of their algorithm in approximating the Pareto set.

A new temporal difference learning algorithm that integrates the Pareto dominance relation into a reinforcement learning method called Pareto Q-learning, was presented by the authors in (Van Moffaert and Nowé, 2014). The algorithm is applicable to both episodic environments with deterministic and stochastic transition functions. The algorithm was evaluated in the Pressurized Bountiful Sea Treasure (PBST) environment, a task characterized by a predefined sequence of episodes with specific initial conditions, involving an agent controlling a submarine to discover underwater jewels by exploring underwater on a 10×11 grid. The jewels have values that significantly increase as the distance from the initial location grows. When compared to several single-policy MORL algorithms, such as the Chebyshev scalarized MORL algorithm, the algorithm outperformed them. This demonstrates that opting for actions that dominate in a specific region does not necessarily ensure the global Pareto optimality of the overall combination of chosen actions in each state, i.e., the policy.

Multi-objective reinforcement learning is used when decision problems involve multiple objec-

tives, which are represented by a multi-objective Markov decision process (MOMDP). This MOMDP is defined by a tuple, $\mathbb{M} = (S, A, T, \gamma, \bar{R}, n)$, where S and A are the state and action spaces, T is a probabilistic transition function, γ is the discount factor for future rewards, and an immediate reward function with n dimensions and vector values is called $\bar{R}$. The number of objectives in the problem is denoted by n. Single-objective reinforcement learning is a special case of MORL, with $n = 1$. The goal in MORL is to find a set of non-dominated solutions, represented by the Pareto front, and learn policies that can reach all of these solutions. Once informed, the decision-maker can decide which course of action they choose to take and apply the associated policy. Finding a policy π^* that optimizes the expected sum of discounted rewards is the aim of single-objective RL (where $n = 1$):

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{h} \gamma^t r_t | \pi, s_0\right] \tag{56}$$

In situations where $n > 1$ in the MORL setting, the returns obtained from the sum may not have a clear winner without additional information. For instance, it can be challenging to determine the optimal return between $(0, 30), (15, 15)$ or $(30, 0)$. A solution is considered "dominated" if its values for all objectives are lower than those of another solution. A solution is said to be "Pareto-efficient" if it is not possible to improve one objective without worsening at least one other objective. In the previously mentioned example, all solutions are Pareto-efficient. The "Pareto front" represents the best feasible compromises among all objectives, and it is the collection of all Pareto-efficient solutions. Thus, the Pareto front provides a range of optimal solutions that can help decision-makers understand trade-offs among different objectives. The primary objective is to gain a comprehensive understanding of the non-dominated solutions in the Pareto front. As illustrated in Figure (7), it is necessary to learn multiple unique policies in order to achieve all of the non-dominated solutions. With this knowledge, the decision-maker can make informed choices and select their preferred course of action. Once a decision has been made, the corresponding policy can be put into effect to achieve the desired outcome.

Pareto-dominance

Pareto-dominance is a term used to describe one policy (π) that outperforms another policy (π') across all objectives. Specifically, if the expected return (V_π or the V-value) of policy π is greater than or equal to the expected return, $V_{\pi'}$ of policy π' across all objectives, and there is at least one objective where policy π outperforms policy π', then policy π' is considered dominated and policy π.

$$V^{\pi} >_P V^{\pi'} \iff (\forall : V_i^{\pi} \geq V_i^{\pi'}) \wedge (\exists i : V_i^{\pi} > V_i^{\pi'}) \tag{57}$$

where $>_P$ is the Pareto-dominance operator.
Identifying the collection of policies Π^* that is not outperformed by any alternative policy is the objective:

$$\Pi^* = \{\pi \in \Pi \mid \nexists\, \pi' \in \Pi : V^{\pi'} >_P V^{\pi}\} \tag{58}$$

where the set of all feasible policies is Π. As shown in Figure (8), Π^* is then used to get the Pareto Front $\mathcal{F} = \{V^{\pi} | \pi \in \Pi^*\}$, which serves as the best coverage set for the problem.

Solution set

Any group of V-values generated from a collection of policies constitutes the solution set. A solution set is the collection of all possible solutions to a problem or decision-making scenario. In the context of multi-objective optimization, the solution set is the set of all possible combinations of the objectives that can be achieved. A solution set can be visualized in a multi-dimensional space, with each dimension representing one of the objectives.

Coverage set

Any solution set that solely contains non-dominated V-values is the coverage set. A coverage set is a subset of the solution set that represents the set of solutions that are considered acceptable according to some predefined criteria. The selection of the coverage set is typically based on the decision maker's preferences or the limitations imposed by the problem at hand. For example, a coverage set may include only solutions that are within a certain range of values for each objective, or that meet certain feasibility requirements.
In summary, the solution set represents all possible solutions to a problem, while the coverage set represents the subset of solutions that are acceptable according to some predefined criteria.

2.8.1 Pareto Fronts

To address multi-objective problems, Pareto optimality, named after economist Vilfredo Pareto, has been used to find solutions that benefit some individuals while not harming others. As multiple objectives are involved in the problem, there is no unique solution. Instead, Pareto solutions represent a collection of acceptable trade-off optimal solutions known as the Pareto front. The analytical phase of multi-criteria decision-making process is multi-objective optimization that aims to determine all Pareto optimal solutions to the multi-objective problem. The Pareto set allows decision-makers to select the preferred solution, which is the most desirable. In contrast to single-objective optimization problems that may overlook the trade-off view point, the Pareto set provides a range of options, containing optimum solutions from an overall standpoint. The optimization of policies in sequential decision problems

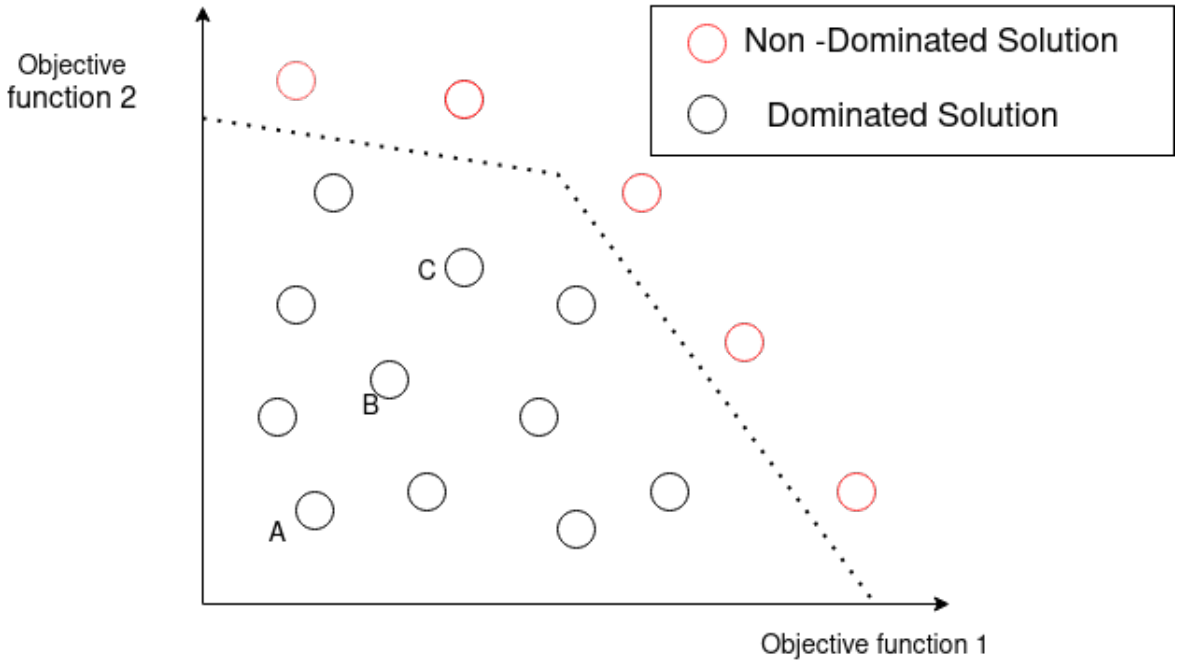


Figure 7: Decision space: Solution space showing dominated and non dominated solutions for a multi-objective optimizatioon problem. Point A is dominated by point B and point C dominates both points A and B

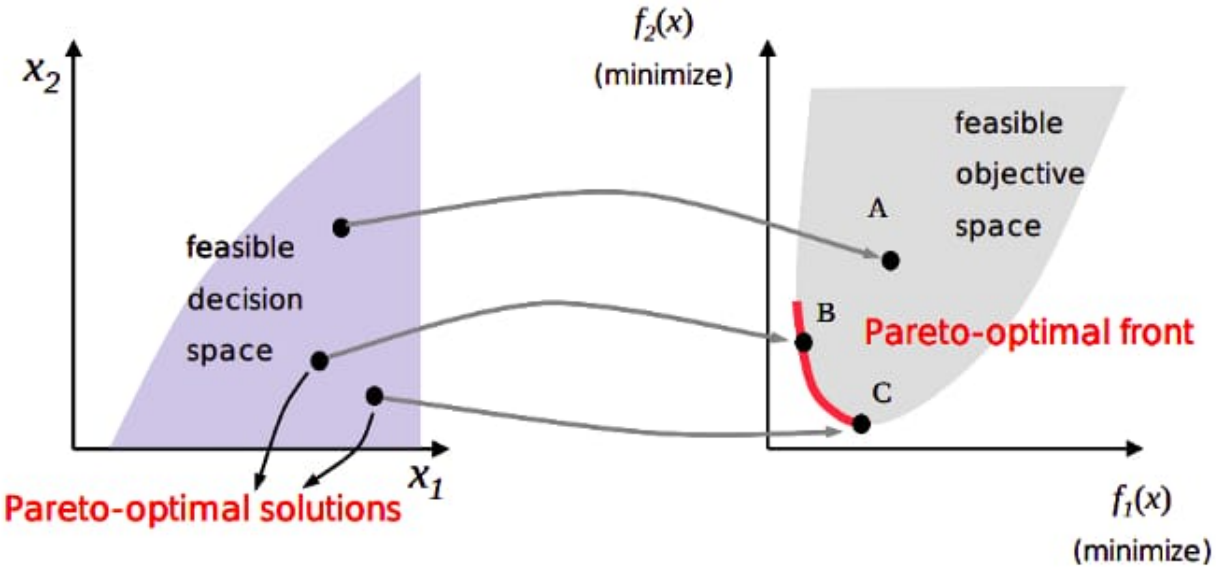


Figure 8: Exploring Decision Space and Objective Space (on left and right respectively): Non-Domination and Pareto Optimality of Solutions B and C (Igor, 2021).

using RL has traditionally been focused on single objectives (Kwak et al., 2021), (Bent et al., 2018). However, addressing problems like epidemic/pandemic mitigation requires balancing various criteria like death, economic costs, attack rate, and societal burden (Colas et al., 2021), (Reymond et al., 2022a). To solve such multi-objective problems, the Pareto front, which includes all the best possible compromises, can be used. (Reymond et al., 2022a) found the Pareto front of a multi-objective decision problem using PCN to optimize attack rate and social burden while (Colas et al., 2021) used DQN and NSGAII to obtain a Pareto front of Pareto-optimal solutions, considering health and economic costs. This work aims to use Soft Actor-Critic, a deep RL algorithm, to find the Pareto front of Pareto-optimal solutions on epidemiological models in (Colas et al., 2021) and (Reymond et al., 2022a), comparing its performance with PCN, DQN, and NSGAII. DQN will be used on the epidemiological model in (Reymond et al., 2022a), while PCN will be employed on the model in (Colas et al., 2021). Assessing the algorithm's efficiency will aid in identifying the best approach for addressing multi-objective optimization problems.

2.8.2 Pareto Conditioned Network (PCN)

In tackling MORL (Multi-Objective Reinforcement Learning) problems, two approaches are commonly employed, namely the utilization of single-policy and multi-policy algorithms (Vamplew et al., 2011). Single-policy algorithms aim to learn a singular, optimal policy based on a specific set of objectives, while multi-policy algorithms concentrate on learning a collection of policies that cover all potential goal functions, often known as a coverage set.

The PCN (Pareto Conditioned Networks) algorithm, as presented by (Reymond et al., 2022b), is a multi-policy approach that employs a supervised learning approach to enhance the policy. It deviates from the conventional temporal difference learning technique in reinforcement learning (RL) to address the moving target problem associated with neural networks as function approximators. In contrast to traditional supervised learning, where the true target is known beforehand, RL faces the difficulty of not knowing the target (such as the best course of action for the policy). By utilizing a supervised learning approach, PCN effectively circumvents this moving target problem. As the agent's behavior improves progressively, the target action it aims for can undergo changes, often resulting in the challenge of fine-tuning and a fragile learning process. To address this, the utilization of this technique contributes to achieving stable learning. The Pareto Conditioned Network (PCN) is a machine learning model that leverages the principles of Pareto optimization. Pareto optimization is a mathematical approach used to determine the optimal balance among multiple objectives. PCN is an architecture specifically designed to optimize multiple objectives simultaneously, based on the notion that a neural network can be trained to optimize numerous goals at once. This is achieved by

training the network with a set of auxiliary objectives alongside the primary ones, enabling the network to learn the optimal trade-off between different objectives.

The input tuple (s, h, R) in PCN represents the state s, planned horizon h, and desired return R. PCN utilizes a single neural network for processing this input. At the beginning of the episode, the decision maker selects the values of h and R to be executed.

Each action a_i in the set A corresponds to a distinct output in the PCN neural network. These outputs indicate the network's level of confidence in the action's ability to achieve the desired return within the specified number of timesteps. This can be likened to a classification problem, where the network learns to assign the label a_i to the value (s, h, R).

To enable PCN to learn the mapping from input to label, a labeled dataset containing training examples is essential, similar to classification tasks. Unlike classification, the dataset used by PCN is not static. PCN actively learns about its environment through exploration, resulting in an evolving dataset that improves over time as it gathers high-quality trajectory data.

Ensuring that only relevant experiences are retained during training is crucial for PCN to achieve Pareto-dominating solutions. This is accomplished by reducing the dataset size and selectively preserving tuples from different parts of the objective-space that possess the desired returns (R).

PCN finds applications in various fields, including image and video synthesis, image and video compression, as well as solving multi-objective optimization problems.

2.8.3 Multi-Objective Metrics

Metrics are primarily employed to assess the performance and efficacy of an algorithm. When dealing with multi-objective problems particularly when seeking Pareto-optimal solutions, analyzing and contrasting the learned coverage sets of various algorithms is not a simple procedure because one algorithm's output may dominate another in one area of the objective space but not another. In (Reymond et al., 2022a), the authors evaluated coverage sets by comparing hospitalizations with social burdens while taking into account four different versions of PCN algorithms, using three different metrics, namely Hypervolume (Zitzler and Thiele, 1999), the $\epsilon-$indicator (I_ϵ) (Zitzler et al., 2003), and $\epsilon-$mean indicator $(I_{\epsilon-\text{mean}})$ (Reymond et al., 2022b).

Hypervolume Indicators

The Hypervolume metric is a widely adopted approach for evaluating outcomes in multi-objective optimization. It quantifies the portion of the goal space that is dominated by the coverage set learned by the algorithm, relative to a predetermined reference point. In order

to guarantee a successful volume measurement, the reference point is selected as the lower boundary. Larger hypervolumes indicate a more dominating coverage set. The hypervolume denotes the extent of the objective space encompassed by the coverage set. As all other potential solutions are outperformed by the coverage set, the hypervolume represents the highest achievable value for the Pareto front, as no further improvements can be made to this volume.

$\epsilon-$indicator (I_ϵ):

This metric assesses how closely a coverage set $\prod^*$ is located to the Pareto front F. By ensuring that there is a corresponding V-value, $V^{\pi'}$ in the coverage set, for each V^π on the Pareto front that is no more than *epsilon* smaller than V^π, the ϵ-indicator measures the distance between the coverage set and the Pareto front.

$$I_\epsilon = \inf_{\epsilon \in \mathbb{R}} \{\forall\ V^\pi \in \mathcal{F}, \exists V^{\pi'} \in \prod^* : ||V^\pi - V^{\pi'}||_\infty \leq \epsilon\} \tag{59}$$

- When the value of ϵ approaches 0, it signifies that the obtained solutions closely align with the true Pareto front, implying a highly accurate approximation.
- A relatively small ϵ value suggests a reasonable approximation, although some distance remains between the obtained solutions and the true Pareto front.
- Conversely, a large ϵ value indicates significant disparity between the obtained solutions and the true Pareto front, implying a poor approximation.

The epsilon indicator measures how closely the solutions generated by a multi-objective optimization algorithm approximate the true Pareto front, which represents the optimal balance among conflicting objectives.

$\epsilon-$mean indicator ($I_{\epsilon-\mathbf{mean}}$)

This represents a modification of the I_ϵ metric, which seeks to provide an upper limit estimation for this value by incorporating an additional assumption. This metric assumes an equal likelihood of selecting each point on the Pareto front as the optimal choice based on a randomly sampled utility function. The computation requires averaging the ϵ values that have been calculated using the same approach as the I_ϵ metric.

2.8.4 Genetic Algorithms

The genetic algorithm(GA), which draws inspiration from Charles Darwin's theory of natural evolution, employs a search heuristic whereby the fittest individuals are selected for repro-

duction to produce the next generation of offspring. Genetic algorithms were developed to explore and optimize solution spaces, where conducting a comprehensive search would be impractical, as well as locations where traditional search methods prove ineffective. The genetic algorithms (GAs) used as function optimizers aim to increase fitness values corresponding to the optimization targets. In general, evolutionary computing approaches, and particularly GAs, have demonstrated remarkable empirical success in solving various challenging design and optimization tasks. GAs start with a randomly initialized population of potential solutions, typically represented as a string (chromosome). The search space is narrowed down by a selection operator, while the crossover and mutation operators generate new candidate solutions.

An optimization and search method based on the ideas of natural evolution is known as a genetic algorithm. It imitates natural selection to identify the best solution to a problem. The algorithm begins with a population of potential solutions, referred to as individuals or chromosomes. It then employs genetic operators, such as selection, crossover (recombination), and mutation, to produce new generations of solutions. The quality of the new solutions is evaluated and selected using a fitness function.The process of using genetic algorithms is repeated until a satisfactory resolution is achieved. Genetic algorithms are heavily relied upon in several industries, such as machine learning, artificial intelligence, engineering, and finance.

2.8.5 Non-dominated Sorting Genetic Algorithm

The NSGA-II method is highly effective in solving multi-objective optimization problems. Three key characteristics of NSGA-II are well known: a clever approach to sort solutions, a speedy way to determine distance, and a straightforward way to compare them. It works well for addressing a variety of goals and locating the most effective solutions. It accomplishes this by classifying solutions into several tiers according to how effective they are. It is widely used in various fields, such as engineering, finance, and biology, due to its effectiveness (Deb et al., 2002). Algorithm (3) shows the pseudocode for NSGA-II. The first step is population initialization, where the population is set up based on the problem range and constraints. The process of non-dominated sorting involves the classification of a given population into different levels based on non-domination criteria. This method helps to distinguish solutions into various degrees of non-domination, where a solution is regarded as dominating another if it outperforms it in all objectives. Solutions that are not dominated by any other solution are referred to as non-dominated solutions and are placed in the first level of non-domination. The second level comprises solutions that are dominated by one or more solutions in the first level, and the process is repeated until every solution has been assigned to a level of non-domination. After sorting the solutions, the next step is to calculate the crowding distance for each solution

in the non-dominated front. This step is crucial as it helps to preserve diversity within the non-dominated front by assigning a crowding distance metric that measures how densely solutions are located within a given region. The next generation of solutions is chosen based on a combination of their rank and crowding distance. The crowding distance plays a critical role in the algorithm as it measures the distance between neighboring solutions in the front and maintains diversity in the population. The idea behind crowding distance is that solutions that are farther apart in the front are more diverse and have a greater chance of yielding better outcomes. The algorithm can examine more options and avoid becoming stuck in a local optimum by maintaining a diverse range of solutions in the population.

In multi-objective optimization, the genetic operators applied to the population include selection, crossover, and mutation. Selection chooses individuals from the current population to generate the next generation, while crossover combines genetic information from two individuals to create a new solution, and mutation randomly alters a small portion of an individual's genetic information. These operations are repeated to create new solutions and maintain diversity in the population. As these solutions evolve over time, they will converge to a group of non-dominated solutions, which represent a set of trade-off solutions for the multi-objective optimization problem.

The recombination and selection step involves combining the offspring population and the current generation population, and selecting individuals for the next generation. Every position in the new generation is filled one after the other until the population exceeds the size of the previous generation. The ultimate population of solutions is composed of Pareto fronts, which are independent of any other solutions and show the trade-offs between various objectives.

In this chapter, we have explored essential concepts, including infectious disease modeling, epidemiological models, various types of compartment models for infectious diseases. We expanded our discussion to include reinforcement learning and deep learning techniques, such as DQN and Double DQN, and their applications in healthcare.

Moreover, we delved into multi-objective optimization and reinforcement learning, examining algorithms like Pareto conditioned networks and NSGA. Moving forward, the next chapter will focus on our case study, specifically the application to COVID-19, and two special cases utilizing models fitted on data from France and Belgium.

Throughout the next chapter, we will present the mathematical models utilized and the reinforcement learning architecture, encompassing the actions and states involved, as well as the reward function.

Algorithm 3 NSGA-II (Poloni et al.)

procedure NSGA-II(number)
    Initialize population(number)
    while Termination criteria not met **do**
        Elitism selection technique()
        Preserving top-performing solutions in each generation.
        Genetic operation()
        Creating new solutions through crossover and mutation of parent individuals.
        objective evaluation()
        Assessing the fitness or performance of solutions based on multiple conflicting objectives.
        Fast non-dominated sorting()
        Efficiently partitioning solutions into multiple Pareto fronts based on dominance relationships.
        Crowding distance assignment()
        Quantifying the diversity of solutions within each front by measuring their distances in the objective space.
    end while
end procedure

3 Case Study: Application on COVID-19

In this chapter, we delve into our case study, focusing on the application of a multi-objective deep reinforcement learning algorithm to tackle COVID-19. We will explore two specific cases of the disease that have drawn our attention and examine the essential elements necessary to formulate the control problem as an optimization challenge within the reinforcement learning framework. The COVID-19 disease is brought on by the SARS-CoV-2 virus, which was discovered for the first time on January 7, 2020. To prevent the spread of the virus, various actions have been taken globally, including total lockdowns, social distancing measures, closure of businesses and educational institutions, banning of public gatherings, and travel restrictions. Governments in many countries have taken these measures to counter the pandemic, as reported by (Kraemer et al., 2020).

3.1 The COVID-19 Epidemic: Optimizing On-Off Lockdown Policies

The authors in (Colas et al., 2021) presented a toolbox named EpidemiOptim, which serves as a platform to bridge the gap between epidemiologists and optimization researchers. Using a standard interface used by optimization practitioners (OpenAI Gym) in Python, Epidemiological models and costs are transformed into optimization challenges by the toolbox. Optimization techniques such as genetic algorithms and reinforcement learning algorithms are utilized in this toolbox to determine the optimal strategy for implementing lock-down measures.
By implementing lock-down measures, the aim is to minimize both the spread of the virus and the economic and social/health impacts. To illustrate the effectiveness of the EpidemiOptim toolbox in adjusting dynamic lock-down policies by minimizing death toll and economic costs, the authors used the SEIR model for COVID-19. The toolbox uses DQN and NSGA-II.
In response to the pandemic, numerous countries implemented stringent lockdown measures that encompassed mobility limitations, limiting physical contact, school and business closures, the cancellation of public events, and prohibitions on gatherings of all sizes. For instance, the lockdown in France lasted for 55 days, from 17th of March to 11th of May. During this period, approximately 690,000 lives were saved, but the nation's economy still endured a significant setback, with GDP declining by at least 10.1 percent according to Mandel and Veetil (2020). This case study examines how various tactics were refined over the course of a year using RL and EA algorithms. The environment analyzed in this epidemic case includes an epidemiological model, defined state and action spaces, and specific cost functions. Three elements are important to define the optimization problem which are the cost/reward function,

the epidemiological model, state spaces and the action spaces which are captured by the environment in which the learning agent(control policy) interacts with. These elements will be discussed in more detail in the following sections.

Example: Simulating the Spread of Infectious Diseases (SIR Model Visualization)

As an illustrative example, we simulate the spread of infectious diseases using the SIR model with the specified parameters: Infection rate (beta) = 0.25, Recovery rate (gamma) = 0.01, Population = 5000, and Initial number of infected individuals = 100. The simulation is set to run for 1000 days, allowing us to visualize the population dynamics of infected, susceptible, and recovered individuals over a period of time.

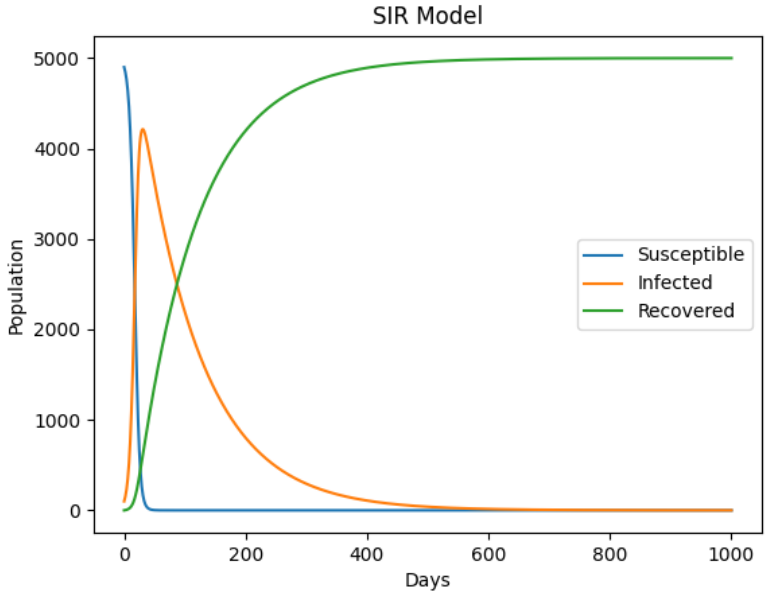


Figure 9: Population Dynamics showing the spread of an infectious diseases over time (SIR Model). The simulation is done using a python code that uses a simple Euler's method to numerically solve the differential equations of the SIR model and simulate the dynamics of susceptible, infected, and recovered individuals over time. The resulting plot shows how, over the course of the 1000 days, the population in each compartment changes. The aim is to recover quickly and return the population to normal by finding effective solutions and policies, with the ultimate goal of eradicating the disease.

The Epidemiological Model (EpidemiOptim)

The epidemiological model controls how the simulation of the epidemic develops. It produces the next state s_{t+1} from the epidemic's present state s_t and the agent's current action a_t, using the transition function $T : S, A \to S$, S represent state space and A stands for action space. The epidemiological model being examined is a well-structured mathematical model designed to depict the changing patterns of the COVID-19 pandemic over time, serving as the foundation for defining the states in the control problem. Specifically, we utilize the SEIRAH model, an extension of the SEIR model discussed in the previous chapter. This model has been effectively employed to the French data gotten from Prague et al. (2020) The SEIRAH model captures the dynamics of various compartments, including the susceptible population (S), exposed individuals (E), infectious cases (I), removed cases (R), non-ascertained asymptomatic cases (A), and hospitalized cases (H). The dynamics among these compartments are characterized by a set of six ordinary differential equations (ODE).

$$\begin{cases} \frac{dS}{dt} & = -\frac{bs(I+\alpha A)}{N}; \quad b = b_0 \exp(\beta_{w1} + \beta_{w2} + \beta_{w3} + \beta_{w'}) \\ \frac{dE}{dt} & = \frac{bs(I+\alpha A)}{N} - \frac{E}{D_e} \\ \frac{dI}{dt} & = \frac{rE}{D_e} - \frac{I}{D_q} - \frac{I}{D_I} \\ \frac{dR}{dt} & = \frac{(1+A)}{D_I} + \frac{H}{D_h} \\ \frac{dA}{dt} & = \frac{(1-r)E}{D_e} - \frac{A}{D_I} \\ \frac{dH}{dt} & = \frac{I}{D_q} - \frac{H}{D_h} \end{cases} \tag{60}$$

where

- b_0 is the rate of transmission among confirmed cases prior to the implementation of a lockdown.
- r is the proportion of actual cases correctly identified and reported in a population.
- The transmission ratio between I and A is α.
- D_e represents the duration of latency or incubation, measured in days.
- D_I represents infectious period measured in days.
- The time in days from the start of I to H is denoted by the variable D_q.
- The length of the hospital stay, in days, is D_h.
- N is the population size.

- β refers to the impacts of the lockdown during the first, second, third, and subsequent weeks (β_k) for $k = 1, 2, 3, w'$ respectively.

State and action spaces

The state variable represents a set of information utilized by the decision-making strategy, including the states of the SEIRAH epidemiological model, boolean indicators for previous and current lockdown states, normalized total costs (health and economic) up to timestep t, and the level of the transmission rate. This state variable enables the control policy to make informed decisions. A list of potential actions a_t is provided, suggesting actions that can be taken to mitigate the epidemic For example, implementing a lockdown can effectively slow down and contain the spread of the epidemic. The action is depicted as follows:

$$a_t = \begin{cases} 0 & \text{if action is off-lockdown} \\ 1 & \text{if action is on-lockdown} \end{cases} \tag{61}$$

Cost functions

A major factor in the formation of optimization problems is the development of cost functions. Controlling epidemic situations can frequently be handled as multi-objective problems, requiring algorithms to optimize numerous cost functions rather than just one. These costs can be determined per timestep or for the entire episode.
The health cost

$$C_h(t) = 0.005R(t)$$

can be assessed either over the course of a whole episode (year) or for each change in environment (weekly).
The economy cost

$$C_{eco}(t) = Y_0 - AK_0^{\gamma_k}((1 - u(t))\lambda(N - G(t)))^{1-\gamma_k}$$

where

- The term $R(t)$ represents the size of the population that has either recovered or been removed from the group under consideration at a specific time, t.
- $G(t)$ is the population's size that is sick, secluded, or deceased.
- N — population size.
- $u(t)$ — extent of the lockdown's partial unemployment.

- K_0 is the economy's pre-outbreak capital stock, which will be taken to have remained stable during the epidemic.
- γ_k is the capital elasticity.
- A is the total factor productivity.
- Y_0 is the Gross Domestic Product (GDP) before pandemic.

3.2 Investigating COVID-19 Mitigation Policies' Pareto-front

In this second case, we examined the initial phase of the outbreak of COVID-19 in Belgium, during which a lockdown was implemented to control the spread of the virus. Our investigation aimed to explore various strategies for easing restrictions and transitioning back to normal activities, with the goal of reducing both the amount of COVID-19 infections and hospitalizations, as well as minimizing the societal impact of the preventive measures that were put in place. The authors in Reymond et al. (2022b), assert that in situations where epidemic mitigation involves multiple, and at times conflicting, criteria such as prevalence, mortality, morbidity, and cost, maximizing policies based on a single objective may not yield favorable results. They suggest using a multi-objective strategy to identify the best or most balanced policies for lowering COVID-19 cases and reducing the social burdens associated with the preventative measures put in place during the Belgian COVID-19 outbreak. The authors develop a number of solutions that approximate the Pareto front of the control problem using a cutting-edge method known as PCN. This approach is particularly intriguing due to its ability to address the complexity of the issue and provide valuable insights. In this instance, serial sero-prevalence data and hospitalization incidence data were used to fit the epidemiological model designed by (Abrams et al., 2021), aiming to represent the COVID-19 outbreak in Belgium. This model is a stochastic discrete-time age-structured compartmental model that replicates mitigation strategies. It modifies social distance factors relating to contacts at work, school, and during free time. Drawing upon the epidemiological model established by Abrams et al. (2021), the authors of (Reymond et al., 2022a) developed MOBelCov, a specialized multi-objective epidemiological environment. MOBelCov employs a multi-objective Markov decision process (MOMDP) framework. By incorporating the epidemiological model devised by (Abrams et al., 2021), MOBelCov enables the simulation of state transitions, while its action space combines proportional reductions in school, work, and leisure contacts at each time step. The environment also defines a reward function based on two objectives: the attack rate (proportion of the population affected by the pathogen) and the social burden. They used a cutting-edge MORL technique called PCN to comprehend and explore the trade-offs

between attack rate and societal load. PCN employs a single neural network to learn the Pareto front-corresponding rules. The authors use the continuous action PCN approach to examine the Pareto front of multi-objective COVID-19 plans by looking at the solution set offered by the mitigation strategies. Their analysis reveals that PCN effectively minimizes the social burden, particularly in scenarios with relatively low hospitalization rates. The RL environment considered in this case consists of

The Epidemiological Model

The epidemiological model considered for the Belgian case study is a mathematical model to understand the COVID-19 epidemic's large-scale dynamics. The efficacy of non-pharmaceutical interventions is evaluated within the framework of the model proposed by (Abrams et al., 2021). In order to study transitional measures toward a less severe COVID-19 condition, this model was used to analyze the first spread of Coronavirus 2 in Belgium. It is a discrete-time, stochastic model that encompasses pre-symptomatic, asymptomatic, mild, and severe symptoms of SARS-CoV-2 transmission as well as the history of an older population. The model considers an age-structured population, designed as an extended SEIR structure for $K = 10$ age groups, that is, $[0-10), [10-20), [20-30), [30-40), [40-50), [50-60), [60-70), [70-80), [80-90), [90-\infty)$. $C_{home}, C_{work}, C_{transport}, C_{school}, C_{leisure}, C_{other}$ are different social environments which stands for the home, work, transport, school, leisure, other environments respectively. In the age-structured model, each compartment, including susceptibles (S), exposed (E), infectious (presymptomatic, asymptomatic, mild, severe, hospitalized and in intensive care unit (ICU)), Death (R) recovered (R), is represented by a vector, indicating the number of people in each age bracket (k) at a given time (t) occupying the corresponding compartment. This approach allows for a detailed examination of how the disease spreads within different age groups, capturing the population composition and changes in each compartment over time. Figure A.1 in Reymond et al. (2022a) shows an illustrative representation of the mechanistic model and these changes are mathematically represented by a system of eight Ordinary Differential

Equations (ODEs) known as the Deterministic Model.

$$\begin{cases} \frac{d\mathbf{S}}{dt} & = -\lambda \mathbf{S} \\ \frac{d\mathbf{E}}{dt} & = \lambda \mathbf{S} - \gamma E \\ \frac{d\mathbf{I}^{presym}}{dt} & = y\mathbf{E} - \theta \mathbf{I}^{presym} \\ \frac{d\mathbf{I}^{asym}}{dt} & = \theta p \mathbf{I}^{presym} - \delta_i \mathbf{I}^{asym} \\ \frac{d\mathbf{I}^{mild}}{dt} & = \theta(1-p)\mathbf{I}^{presym} - (\psi + w_2)\mathbf{I}^{mild} \\ \frac{d\mathbf{I}^{sev}}{dt} & = \psi \mathbf{I}^{mild} - w\mathbf{I}^{sev} \\ \frac{d\mathbf{I}^{hosp}}{dt} & = \phi_1 w \mathbf{I}^{sev} - (\delta_3 + \tau_1)\mathbf{I}^{hosp} \\ \frac{d\mathbf{I}^{icu}}{dt} & = (1-\phi_1) w \mathbf{I}^{sev} - (\delta_4 + \tau_2)\mathbf{I}^{icu} \\ \frac{d\mathbf{D}}{dt} & = \tau_1 \mathbf{I}^{hosp} - \tau_2 \mathbf{I}^{icu} \\ \frac{d\mathbf{R}}{dt} & = \delta_1 \mathbf{I}^{asym} - \delta_2 \mathbf{I}^{mild} + \delta_3 \mathbf{I}^{hosp} + \delta_4 \mathbf{I}^{icu} \end{cases} \tag{62}$$

where,

- The compartment denoted as **E** represents the people already exposed to the disease, and they transition to this compartment at a specific rate λ over a certain period of time. $\mathbf{E} = (E_1(t), E_2(t), ..., E_k(t))^T$ is the vector indicating how many people in each age group k were exposed at time t. Other compartments are also vectors representing their members of the population of each group at a particular time.

- The compartment labeled as **S** represents people who are vulnerable to infection. **S** is an array denoting vulnerable people within each age category of the population at time t.

- The compartment denoted as $\mathbf{I}^{presym}$ represents the pre-symptomatic stage, where an exposed individual transitions to after becoming infectious over a specific period of time at a rate of γ.

- When individuals get infected but do not develop any symptoms, there is a probability of p that they will be in the compartment labeled as I^{asym}.

- When individuals become infected, there is a probability of (1 - p) that they will develop mild symptoms, transitioning them to the compartment $\mathbf{I}^{mild}$. In this compartment, they recover at a rate of δ_2. However, there is also a possibility that they will experience a more severe infection, leading them to the compartment $\mathbf{I}^{sev}$ at a rate of ψ.

- Individuals who experience a severe infection are subsequently transferred to a hospital

for treatment, where they are represented by the compartment $\mathbf{I}^{hosp}$. The probability of this transfer is denoted by ϕ.

- Individuals who become critically ill have a probability of (1 - ψ_1) of being directly transferred to the ICU. For the hospital setting, individuals are categorized into two compartments: $\mathbf{I}^{hosp}$ and $\mathbf{I}^{icu}$. Both individuals in the hospital compartment ($\mathbf{I}^{hosp}$) and ICU compartment (I^{icu}) have the potential to recover or experience unfortunate outcomes. The recovery rates for these compartments are denoted as δ_3 and δ_4 respectively. However, there is also a risk of mortality with rates τ_3 and τ_4 respectively for individuals in the hospital and ICU compartments.

- The compartment denoted as R represents the stage where asymptomatic individuals recover. The recovery process occurs at a rate of δ_1.

Due to human nature, interventions to stop the transmission of a virus, such as limiting social contacts or government activities, add uncertainty to the outbreak's course. They investigated a model that contains a stochastic component known as the chain binomial model to account for the unpredictable impacts of these treatments on social interactions in order to ascertain how this uncertainty affects the transmission of disease. The chain binomial model offers a probabilistic viewpoint on an outbreak by taking into account the probabilistic evolution of infected people through discrete generations. This approach views the spread of the disease as a sequence of interconnected events, accounting for uncertainty and randomness in the epidemic's development over time and is given as follows. The interval of time between two consecutive evaluations of the model is denoted by $(t, t+h]$ and is represented as the length h. A stochastic age-structured model that is discretized is fully specified by the following

- $\mathbf{E}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{S}_t(k), p_t^*(k))$, assuming there are $S_t(k)$ vulnerable persons in age group k at time t, this results in the anticipated number of susceptible individuals in age group k who will transition to the exposed state within the time span from t to $t+h$.

$$p_t^*(k) = 1 - (1 - p_t(k))^{I_t} \tag{63}$$

$$p_t^* = 1 - exp\Bigg[- h \sum_{k'}^{K} \beta_{asym}(k, k')\{I_t^{asym}(k')\} + \beta_{sym}(k, k')\{I_t^{mild}(k') + I_t^{sev}(k')\}\Bigg] \tag{64}$$

This gives a probability of infection denoted as $p_t^*(k)$ which pertains to susceptible individuals within age group $k = 1, ..., K$, while the overall count of people infected at time t is represented by I_t. Additionally, $p_t(k)$ represents the likelihood of transmission when an age group k susceptible person comes into touch with an infected person.

- $\mathbf{I}^{presym}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{I}^{presym}_{t}(k), \Delta_{presym})$, where $\Delta_{presym} = 1 - exp(-hp(k)\theta)$

  This refers to the count of people in age group k who moved from the exposed stage to the pre-symptomatic stage within the given time period. In a similar manner, the stochastic transitions in the other compartments are also determined.

- $\mathbf{I}^{mild}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{I}^{presym}_{t}(k), \Delta_{mild}\}\theta])$ where $\Delta_{mild} = 1 - exp[-h\{1 - p(k)$

- $\mathbf{I}^{sev}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{I}^{mild}_{t}(k), \Delta_{sev})$ where $\Delta_{sev} = 1 - exp(-h\psi(k))$

- $\mathbf{I}^{hosp}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{I}^{sev}_{t}(k), \Delta_{hosp})$ where $\Delta_{hosp} = 1 - exp(-h\psi_1\phi(k))$

- $\mathbf{I}^{icu}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{I}^{sev}_{t}(k), \Delta_{icu})$ where $\Delta_{icu} = 1 - exp[-h\{1 - \phi_1(k)\}\omega(k)]$

- $\mathbf{D}^{hosp}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{I}^{hosp}_{t}(k), \Gamma_{hosp})$ where $\Gamma_{hosp} = 1 - exp(-h\tau_1(k))$

- $\mathbf{D}^{icu}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{R}^{icu}_{t}(k), \Gamma_{icu})$ where $\Gamma_{icu} = 1 - exp(-h\tau_2(k))$

- $\mathbf{R}^{asym}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{I}^{asym}_{t}(k), \beta_{asym})$ where $\beta_{asym} = 1 - exp(-h\delta_2(k))$

- $\mathbf{R}^{hosp}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{I}^{hosp}_{t}(k), \beta_{hosp})$ where $\beta_{hosp} = 1 - exp(-h\delta_3(k))$

- $\mathbf{R}^{icu}_{new,t+h}(k) \sim \mathbb{B}(\mathbf{I}^{icu}_{t}(k), \beta_{icu})$ where $\beta_{icu} = 1 - exp(-h\delta_4(k))$

and

$$
\begin{cases}
\mathbf{S}_{t+h}(k) & = \mathbf{S}_t(k) - \mathbf{E}_{new,t+h}(k) \\
\mathbf{E}_{t+h}(k) & = \mathbf{E}_t(k) + \mathbf{E}_{new,t+h}(k) - \mathbf{I}^{presym}_{new,t+h}(k) \\
\mathbf{I}^{presym}_{t+h}(k) & = \mathbf{I}^{presym}_{t}(k) + \mathbf{I}^{presym}_{new,t+h}(k) - \mathbf{I}^{asym}_{new,t+h}(k) - \mathbf{I}^{mild}_{new,t+h}(k) \\
\mathbf{I}^{asym}_{t+h}(k) & = \mathbf{I}^{asym}_{t}(k) + \mathbf{I}^{asym}_{new,t+h}(k) - \mathbf{R}^{asym}_{new,t+h}(k) \\
\mathbf{I}^{mild}_{t+h}(k) & = \mathbf{I}^{mild}_{t}(k) + \mathbf{I}^{mild}_{new,t+h}(k) - \mathbf{I}^{sev}_{new,t+h}(k) - \mathbf{R}^{mild}_{new,t+h}(k) \\
\mathbf{I}^{sev}_{t+h}(k) & = \mathbf{I}^{sev}_{t}(k) + \mathbf{I}^{sev}_{new,t+h}(k) - \mathbf{I}^{hosp}_{new,t+h}(k) - I^{icu}_{new,t+h}(k) \\
\mathbf{I}^{hosp}_{t=h}(k) & = \mathbf{I}^{hosp}_{t}(k) + \mathbf{I}^{hosp}_{new,t+h}(k) - \mathbf{D}^{hosp}_{new,t+h}(k) - R^{hosp}_{new,t+h}(k) \\
\mathbf{I}^{icu}_{t+h}(k) & = \mathbf{I}^{icu}_{t}(k) + \mathbf{I}^{icu}_{new,t+h}(k) - \mathbf{D}^{icu}_{new,t+h}(k) - R^{icu}_{new,t+h}(k) \\
\mathbf{D}_{t+h}(k) & = \mathbf{D}_t(k) + \mathbf{D}^{hosp}_{new,t+h}(k) + \mathbf{D}^{icu}_{new,t+h}(k) \\
\mathbf{R}_{t+h}(k) & = \mathbf{R}_t(k) + \mathbf{R}^{asym}_{new,t+h}(k) + \mathbf{R}^{mild}_{new,t+h}(k) + \mathbf{R}^{hosp}_{new,t+h}(k) + \mathbf{R}^{icu}_{new,t+h}(k)
\end{cases}
\tag{65}
$$

Reymond et al. (2022b) used the above model to create a version of MOBelCov with a stochastic transition function denoted as $\mathbb{B}$(Binomial). The $\mathbb{B}(.,.)$ function signifies the change from one state (the first position) to another state with a specific probability (the second position).

The State Space

The state variables that make up the epidemiological model are included in the multi-objective Markov decision process (MOMDP). The state is a composite representation of the various components within the epidemiological model s_{em} and the social contact matrix C as explained in 2.1.3. s_{em} is a tuple

$$s_{em} = \{S_k, E_k, I_k^{presym}, I_k^{asym}, I_k^{sev}, I_k^{hosp}, I_k^{icu}, H_k^{new}, D_k, R_k\} \tag{66}$$

for each group $k \in \{1, ..., K\}$ described in 3.2, where S encodes the people who are susceptible to infection and E encodes the members of the population who have been exposed to COVID-19, The people infected with COVID-19 are $I_k^{presym}, I_k^{asym}, I_k^{sev}, I_k^{hosp}, I_k^{icu}$ namely pre-symptomatic, asymptomatic, have mild symptoms, are hospitalised, or are in the Intensive Care Unit respectively. Finally, H_k^{new}indicates the number of people in the age bracket k who have recently been hospitalized.
A state in the RL environment known as MOBelCov is as follows:

$$s = s_{em} \cup C \tag{67}$$

C is a contact reduction function that imposes a proportional reduction of work (including transport) p_w, school p_s and leisure p_l contacts, which is implemented as a linear combination of contact matrices a social contact matrix defined as

$$C = C_{home} + p_w(C_{work} + C_{transport}) + p_s C_{school} + p_l(C_{leisure} + C_{other}) \tag{68}$$

models the contacts of the different age-groups which impact the propagation rate of the epidemic.

Action Space

The action space $\mathcal{A}$ is a 3-dimensional continuous vector in $[0, 1]^3$ that impacts the social contact matrix C as described in Equation (68)

$$a = [p_{work}, p_{school}, p_{leisure}] \in \mathcal{A}$$

Due to the possibility of discretizing the action space they considered two variants of the MDPs: Either a 3-dimensional continuous-action MDP, where you can directly decide p_{work},

p_{school} and $p_{leisure}$ or a discretized variant with a pre-selection of possible reductions:

$$[p_{work}, p_{school}, p_{leisure}] \in \begin{cases} work: & 0, 0.3, 0.6 \\ school: & 0, 0.5, 1 \\ leisure: & 0.3, 0.6, 0.9 \end{cases} \tag{69}$$

In the example given for the discretized version above, the action space size $|\mathcal{A}| = 27$ where $a_i \in \mathcal{A}$ could be $[0, 1, 0.9]$.

Transition Function

The transition of the agent in the environment is given as $P(s'|s_{cm}, C)$ which is the probability transition that the agent transits to a new state given the old state $s = s_{cm} \cup C$. At each timestep t, the the model is simulated for one week using the C obtained from action a_t. The compartment model is represented by a set of ODEs which were transformed into a stochastic chain-binomial process. As a result, two versions of the MOBelCov environment were created. One version uses the deterministic ODE model, which is a MDP with a deterministic transition function. The other version uses the stochastic binomial model, which is a MDP with a stochastic transition function.

Reward Function

The attack rate (i.e., infections, hospitalizations) and the burden of the interventions on the population are two objectives that the authors set as part of a vector reward function.
The attack rate in terms of infections is defined as the difference in susceptibles from the current state and next state according to Libin et al. (2021). The authors in (Reymond et al., 2022a) defined the related reward function as the negative attack rate because the cost needs to be minimized.

$$\mathcal{R}_{ARI}(s, a, s') = -(\sum_{k=1}^{K} S_k(s) - \sum_{k=1}^{K} S_k(s')) \tag{70}$$

The reward function to reduce the attack rate in terms of hospitalisations is simply defined as the negative number of new hospitalizations:

$$\mathcal{R}_{ARH}(s, a, s') = -\sum_{k=1}^{K} H_k^{new}(s) \tag{71}$$

For social burden, the authors considered using the missed contacts resulting from the intervention measures. This is quantified by computing the difference between the original social contact matrix $\hat{C}$ and the installed social contact matrix C which is then used in

computing the social burden reward function

$$\mathbb{R}_{SB}(s,a,s') = \sum_{i=1}^{K}\sum_{j=1}^{K}(C-\hat{C})_{ij}S_i(s)S_j(s) + \sum_{i=1}^{K}\sum_{j=1}^{K}(C-\hat{C})_{ij}R_i(s)R_j(s) \tag{72}$$

where $S_k(s)$ represents the number of persons in state s who are susceptible in age group k, and R_k denotes the number of individuals in state s who are recovered in age group k.
Here, the PCN algorithm is used to optimize two multi-objective reward functions namely $[R_{ARH}, R_{SB}]$ and $[R_{ARI}, R_{SB}]$.

Timesteps

At every timestep, the agent has the ability to select an action that will transition it into a new state S' from the present state S. To clarify further, the model will aim to replicate the progression of the epidemic over a span of one week, providing the agent with advanced access to the current status of the epidemic.
In this chapter, we have thoroughly examined the two specific cases of focus, including the essential components of the optimization problem, such as the epidemiological model, as well as the corresponding state and action spaces for each case study. Moving forward to the next chapter, we will delve into the experiments conducted and the results obtained by applying multi-objective and deep reinforcement learning algorithms, such as DQN, Goal-DQN, NSGA, and PCN.

4 Experiments and Results

We will examine the results and importance of the experiments in this chapter. Furthermore, we will explore the rationale for considering certain decision problems as multi-objective problems, enabling us to utilize multi-objective algorithms that effectively balance conflicting policies when addressing the decision problem. We are focusing on applying multi-objective algorithms to two different COVID-19 simulated gym environments.

4.1 Experiments

The objective of reinforcement learning is to improve the agent's actions through interaction with the environment. To evaluate the agent's decision-making, we examine the cumulative reward acquired during these interactions. In our multi-objective decision-making scenario, where our objective is to minimize the impact on both health and the economic cost and in another case where we aim to minimize the attack rate or the number of hospitalized individuals while taking into account the social burden arising from strategies implemented to control the spread of COVID-19, the approaches used to assess the algorithm or learning agent's performance differ, which is the use of Pareto fronts achieved by the algorithms compared to the Pareto fronts of the fixed policies (100 predetermined policies which would serve as a baseline) and other metrics such as hypervolume, epsilon indicator and epsilon mean indicator as discussed in the preceding chapter. The hyperparameter values utilized in the experiments have been adopted from Colas et al. (2021) and Reymond et al. (2022b). It is important noting that modifying some of these values had only a minor impact on the achieved Pareto fronts by the algorithms.

Case 1: The COVID-19 Epidemic: Optimizing On-Off Lockdown Policies

As mentioned earlier in Section (3.1), the authors have created a toolkit called EpidemiOptim, comprising algorithms like DQN (with variations) and NSGA-II, along with an epidemiological model that models the propagation of the coronavirus disease. Additionally, they incorporated an intervention strategy, both on and off policy, to regulate the disease's spread within a particular population. Governments face two primary challenges in managing the disease's transmission: health costs and economic costs. The learning agent must acquire optimal policies that minimize these costs over the long term.

4.2 Algorithms and their Parameters

DQN (Deep Q-Network)

As mentioned earlier, DQN (Deep Q-Network) is a reinforcement learning algorithm that combines Q-learning with deep neural networks. We utilized DQN for this particular problem due to its state-of-the-art capabilities in effectively learning optimal policies for optimization problems formulated as reinforcement learning tasks. The choice of proper parameters and hyperparameters affects the algorithm's performance. As an illustration, the conventional neural network architecture consists of an input layer, many hidden layers, and an output layer. Increasing the number of units per layer can enhance the network's capacity.

The optimization process of the neural network encompasses an optimizer, which is an algorithm employed to update the network's weights during training. The performance of the algorithm is also influenced by the batch size, which denotes the number of training examples used in each iteration. Furthermore, the frequency of updating the target network has an impact. Less frequent updates can enhance training stability but may cause the target network to lag behind the current network.

Exploration, denoted by the ϵ value, determines the exploration rate, ranging from 0 to 1, which dictates the probability of the agent selecting a random action instead of the action with the highest estimated Q-value. The simulation horizon indicates the number of steps or actions taken within a simulated environment before resetting the agent's decision-making process. Its purpose is to reduce computational requirements during agent training. The save policy frequency refers to how often the agent's current policy or set of actions is saved during training. This allows for the potential restoration of the agent's policy or evaluation of its performance at different stages of training. The buffer size determines the memory capacity allocated to store the agent's experiences, particularly the state-action pairs, for the purpose of training. This storage, referred to as the replay buffer, acts as a repository of experiences from which the agent can learn and improve its decision-making abilities. Table 1 presents the hyperparameters utilized in the DQN algorithm, designed to learn effective policies for solving optimization problems. These hyperparameters play a vital role in shaping the algorithm's performance and behavior.

How is the DQN applied in this case?

To implement the algorithm, several components are required, collectively forming the epidemiological environment. These components include epidemiological models, epidemiological states denoted as s_t along with their associated transition probabilities, on and off actions, and the corresponding rewards for each action a_t in a given state at time t. Furthermore,

Hyperparameters	**Values**
Activation	*relu*
Units in hidden layer	64
Optimizer	*Adam*
Loss Function	*MSE*
Batch size	64
Learning rate	0.001
Training frequency	1000000
Discount factor	0.99
exploration (ϵ value)	0.2
Target network update frequency	5000
simulation horizon	364
save policy frequency	1000
Buffer size	1000000

Table 1: The different hyper-parameters used by the DQN algorithm.

we incorporate the aforementioned hyperparameter values to facilitate the execution of the algorithm. Having established all these essential components, we can now proceed to execute the following steps using the algorithm:

- To start the process, we initialize the neural network that will serve as the Q-function, setting the weights randomly. Additionally, we initialize the replay memory with a predetermined capacity. In our case, we utilize two distinct neural networks: one is responsible for generating Q-values for the new state, denoted as $Q_{predict}$, and the other for the subsequent state, referred to as Q_{eval}. The Q_{eval} is used to calculate the Q-target,

$$Q_{target} = \text{reward} + \gamma \max(Q_{eval}(\text{s'},\text{a})) \tag{73}$$

then use $Q_{predict}$ with Q_{target} to get the loss.

$$loss = \frac{1}{N}\sum_{i=1}^{N}(Q^i_{predict} - Q^i_{target})^2 \tag{74}$$

The optimizer is responsible for computing the gradient of the loss function with respect to the weights. It leverages this information to update the weights in a direction that minimizes the loss through backpropagation. For our implementation, we have chosen the *Adam* optimizer. This optimizer is favored due to its ease of use and ability to incorporate moving averages of the gradient and squared gradient. By giving more weight to recent examples in the training data, *Adam* helps prevent the algorithm from getting trapped in sub-optimal minima.

- The agent interacts with the environment by engaging in a predetermined number of episodes, aiming to learn the most effective actions. Within each episode, the agent follows a specific sequence: it chooses an action using epsilon-greedy exploration, executes the action, observes the resulting reward and next state, and stores this experience in the replay memory. Later on, a random batch of experiences is selected from the memory and employed to update the Q-function using Temporal Difference learning. This entire process continues until the episode concludes. Once an episode ends, the agent utilizes the trained Q-function to determine actions that align with the desired behavior within the environment. It is worth noting that the predetermined episodes considered corresponds to both the number of training steps and the simulation horizon.

- The DQN targets only one cost function:

$$\bar{c} = (1 - \beta)c_h + \beta c_{eco} \tag{75}$$

  where c_h and c_{eco} are health and economic costs respectively which the environment generate as a reward for the action selected by the agent. In this experiment, we will train independent policies and take values of β in $[0, 0.05, 0.1, ..., 1]$ and subsequently, integrate separately trained DQN policies to form a group of policies that collectively establish the Pareto fronts.

- In a nutshell, the Deep Q-Network (DQN) is used to train a Q-network that calculates the value of carrying out a certain action (a) in a specified state (s). The value is specified as the anticipated accumulative negative cost $(-\bar{c})$ over a given period of time.

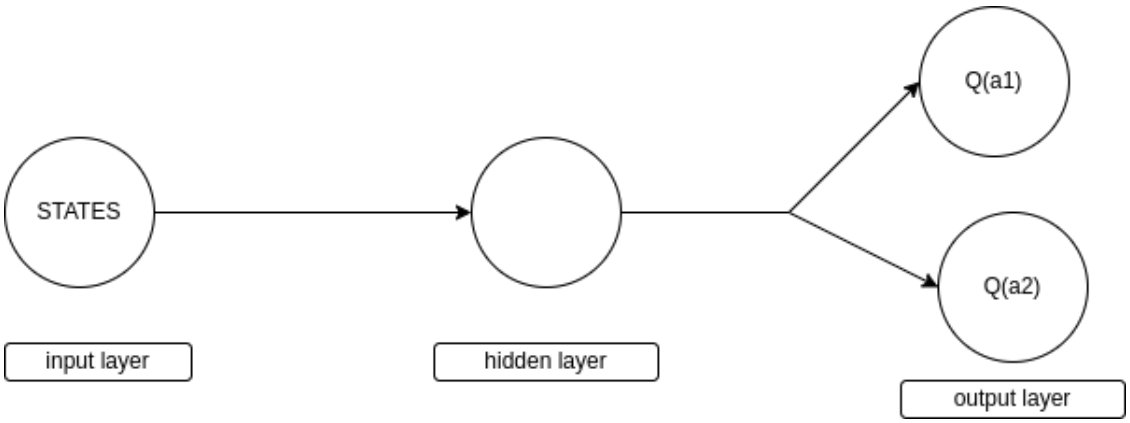


Figure 10: Illustrating the DQN Architecture, where the input layer represents the state space, the hidden layer for processing the data into an output layer which produces action values (Q_{a_1}, Q_{a_2}) for a given state.

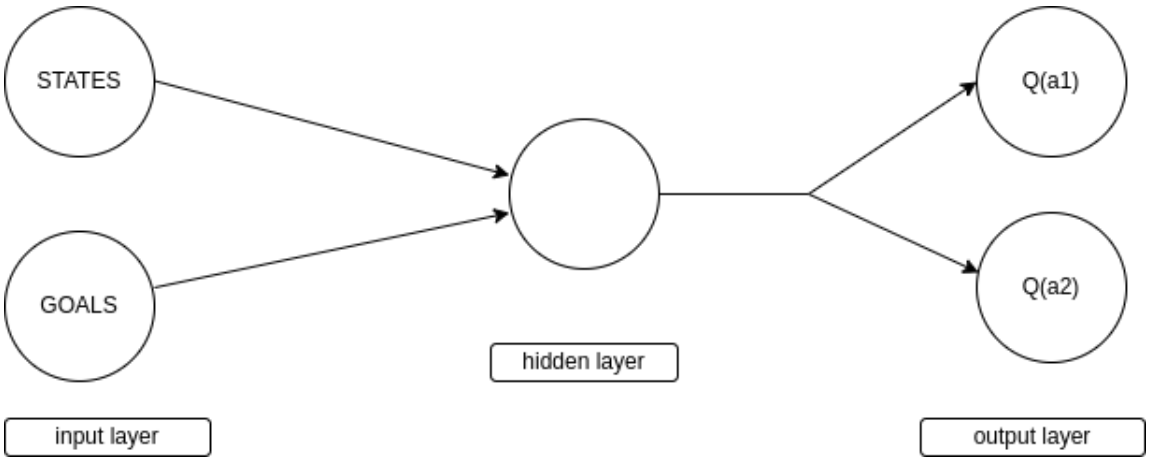


Figure 11: GOAL-DQN Framework where the input layer represents the state space with the goals. the hidden layer and the output layer which produces the action values for a given state and goal.

There is a variant of the DQN algorithm applied on this COVID-19 environment called GOAL-DQN.

GOAL-DQN

GOAL DQN is an adapted version of the Deep Q-Network (DQN) algorithm, where a single policy ($\prod(s, \beta)$) is trained to handle various parameter combinations defined by β. The approach is inspired by the concept introduced by (Badia et al., 2020), where instead of aggregating costs (rewards) at the reward level and Q-function approximation based on this signal, it involves training two distinct Q-networks, one for health costs and another for economic costs. The optimal action is determined by selecting the action that maximizes the combined values of the two Q-functions.

$$\prod(s, \beta) = \arg\max_{a} \ (1 - \beta)Q_h(s, a) + \beta \ Q_{eco}(s, a) \tag{76}$$

where, Q_h represents the Q-value for health cost, and Q_{eco} represents the Q-value for economic cost. During training, the agent randomly selects a target β value from the range of $[0, 1]$. The hyperparameters utilized in the GOAL DQN algorithm are mostly similar to those used in the DQN algorithm, with the inclusion of the goals. In this context, the goal is represented by the β value, which is concatenated with the states the agent should aim for during each interaction when training the network. Here, we evaluate one Goal DQN policy against a set of 100 uniformly selected Pareto goals, and the resulting population undergoes filtering to derive the Pareto fronts.

NSGA-II:

This algorithm approaches the problem of COVID-19 as a multi-objective optimization problem, aiming to discover a collection of non-dominated solutions that strike a balance between minimizing the health cost and the economic cost.

How is the NSGA-II applied in this setting?

We begin by initializing the population of solutions with randomly assigned weights. Each solution represents a unique collection of weights for a Q-network. The fitness of each solution in the population is evaluated by interacting with the learning environment and observing the resulting reward. The solutions are then divided into two groups based on their fitness: non-dominated solutions and dominated solutions. Leveraging its non-dominated sorting ability, the agent creates a front consisting of non-dominated solutions, which are solutions that are not outperformed by any other solution in the population. Additionally, the crowding distance of each solution in the front is calculated. Solutions for the next generation are selected using a combination of non-dominated sorting and crowding distance calculation.For a predetermined number of episodes, this process is repeated. Ultimately, the agent employs the best solution from the population to select actions in the environment, effectively working towards the goal of minimizing the impact on health and the economy. The hyperparameters utilized in NSGA-II can be found in Table 2.

Hyperparameters	**Values**
number of train steps	500000
simulation horizon	364
evaluate and log every	1000
nunmber of generations	20
population size	30
layers	64
policy	neural network
number of evaluations if stochastic	30

Table 2: The different hyper-parameters used by NSGA-II

Case 2: Investigating multi-objective Pareto front Reinforcement learning-based COVID-19 mitigation strategies

As mentioned earlier in Section (3.2), the authors adopted a multi-objective decision approach to achieve balanced policies. They used a deep multi-objective reinforcement learning method

and enhanced the cutting-edge PCN algorithm. The goal was to identify a set of options that approximately balance reducing COVID-19 infections and hospitalizations with taking into account the societal burden brought on by the implemented mitigation measures.

In this experiment, our algorithm was applied to various scenarios involving the approximation of Pareto fronts. These fronts were determined for two different models in the environment: a Deterministic (ODE) model and Stochastic (Binomial) models. The algorithm aimed to learn a set of solutions that capture the trade-off between infections and social burden, as well as hospitalizations and social burden.

For this experiment, we will be utilizing the PCN algorithm with the following set of hyperparameters to guide its learning process and obtain a desirable Pareto front.

PCN

The PCN algorithm was initially designed for discrete actions; however, in this study, the authors extended its application to incorporate continuous actions, considering the nature of the applied mitigation measures that involve continuous values. The specific hyperparameters employed for this experiment can be found in Table 3.

Hyperparameters	**Values**
learning rate	0.001
batch size	256
model updates	50
episodes between updates	10
ER size (in episodes)	1000
initial random episodes	200
reward scaling	[10000, 100]
total training steps	300000
desired return noise	0.2

Table 3: The different hyperparameters used by the PCN algorithm

How is the PCN applied to this decision problem?

The initial step involves determining the objectives, which can either focus on achieving a balance between infections and social burden or between hospitalizations and social burden. In Section 3.2, the state space, action space, and reward function have already been defined. Optimization can be performed on either R_{ARI} or R_{ARH}, as research has indicated a correlation between infections and hospitalizations (Reymond et al., 2022a).

During an *initial random episode*, data will be gathered from the MOBelCov environment, which can be either stochastic or based on ODE models, to train the PCN algorithm. The

collected data will consist of states, actions, desired return, and desired horizon. The network will be trained using this data, with the inputs being states, desired return, and desired horizon, and the outputs being the corresponding near-optimal actions for each objective. The objective of the network is to approximate the Pareto front, which represents the set of optimal trade-off solutions between multiple objectives. This is achieved by minimizing the error between the actual output and the predicted output through a process known as backpropagation. The network undergoes training over multiple episodes to improve its approximation of the Pareto front.The PCN algorithm trains the model using a specified number of training steps referred to as "total training steps". A batch of data from the gathered dataset is sampled during the training phase in order to update the model. The batch size is an essential parameter required for updating the model, as it determines the number of samples to be included in each training batch.

4.3 Results

We show the results of the experiments achieved in this section by applying the algorithms as learning agents on the two cases which are both captured by the OpenAI gym environments.

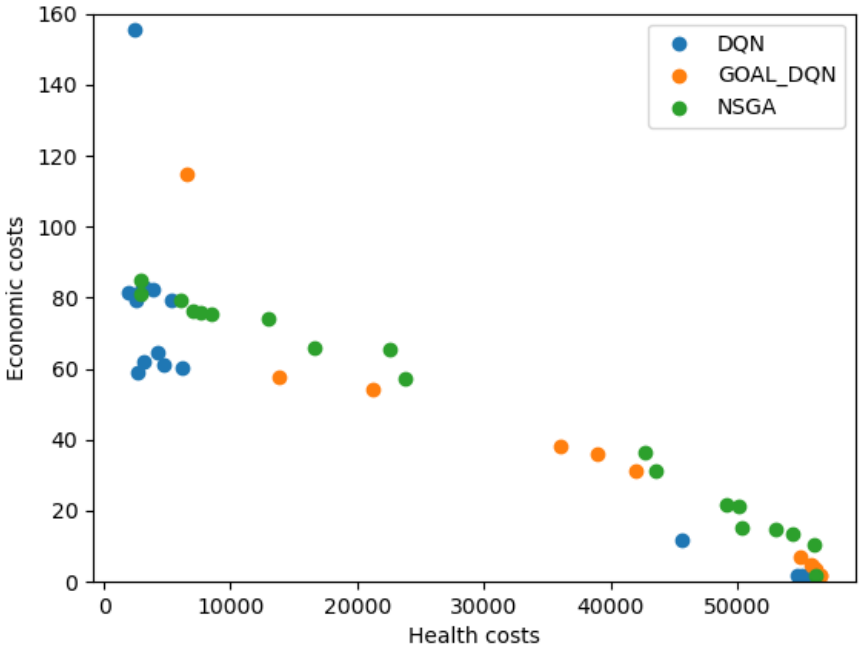


Figure 12: The Pareto plots of policies discovered by DQN, Goal-DQN and NSGA-II algorithms. The plot shows the comparison of the policies discovered by the agents to minimize the Economic and Health costs.

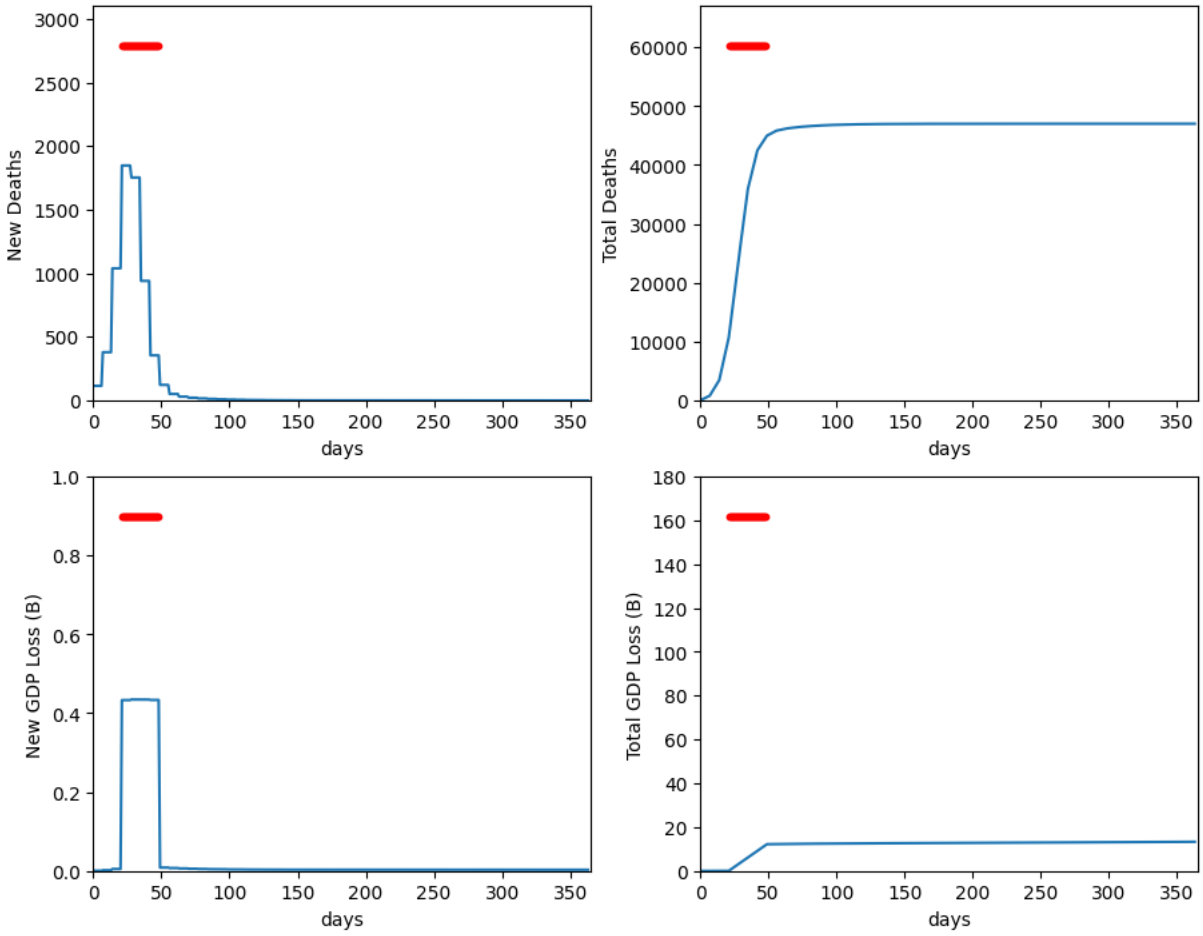


Figure 13: plot of actions selected to minimize the health and economy cost for a year (365 days) simulation using NSGA-II. The red dots means lockdown is on and the white space indicates no lockdown (i.e Lockdown enforcement are marked by red dots). In this case, the agent discovered an approach that effectively decreases economic costs by implementing a lockdown lasting a a couple of weeks to suppress the initial surge. However, this approach performed poorly in terms of reducing health costs, resulting in a range of 40,000 to 50,000 deaths within the first 50 days of its implementation.

	Hypervolume	I_ϵ	$I_{\epsilon-mean}$
PCN_{ARH} (0DE)	0.1207	0.040	0.006
PCN_{ARH} (Binomial)	0.1194	0.040	0.009
PCN_{ARI} (0DE)	0.1197	0.037	0.007
PCN_{ARI} (Binomial)	0.1269	0.043	0.011

Table 4: Evaluation metrics for the coverage sets comparing hospitalizations with social burden showing the hypervolume, ϵ-indicator(I_ϵ) and ϵ-mean indicator($I_{\epsilon-mean}$).

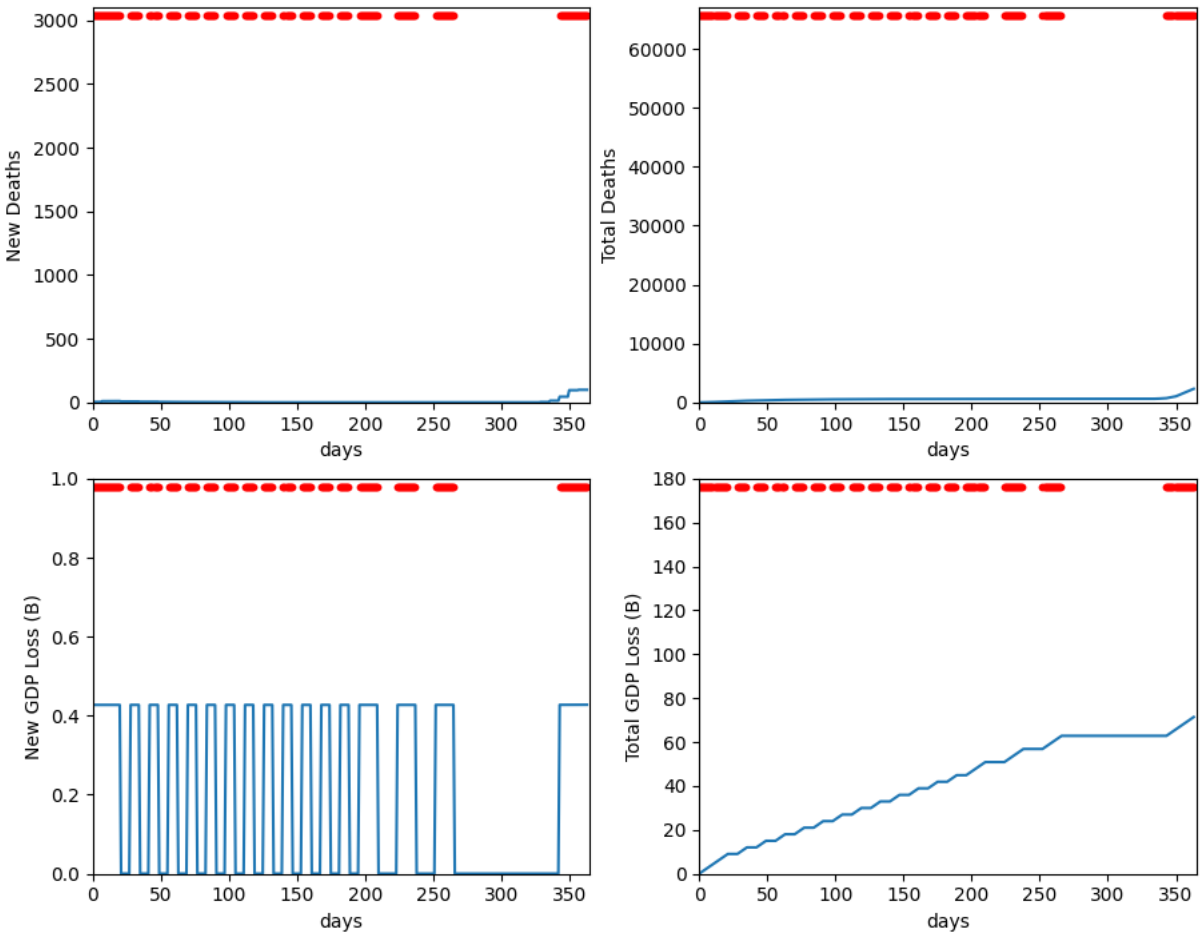


Figure 14: Plot of actions selected to minimize the health and economic cost for a year simulation using DQN. In this approach, the agent begins with an initial sustained lockdown and subsequently implements periodic lockdowns. As a result of this strategy, there is a reduction in health costs but an increase in economic costs. Importantly, the economic costs remain stable on days when there is no lockdown.

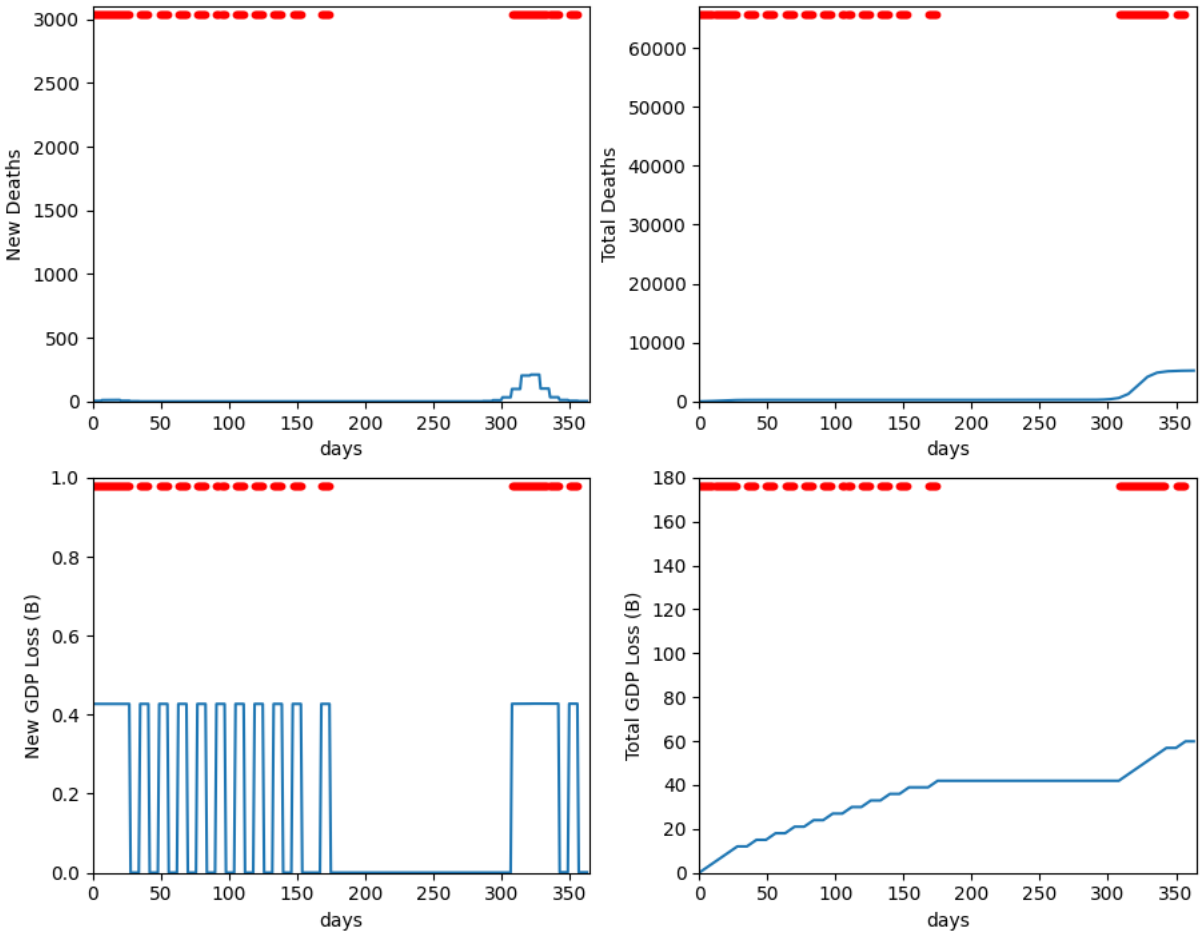


Figure 15: Plot of actions selected to minimize the health and economy cost for a year simulation using GOAL-DQN. In this approach, the agent initiates a sustained lockdown initially and then follows with periodic lockdowns. This strategy guarantees the prevention of a second wave, resulting in lower health costs but higher economic costs.

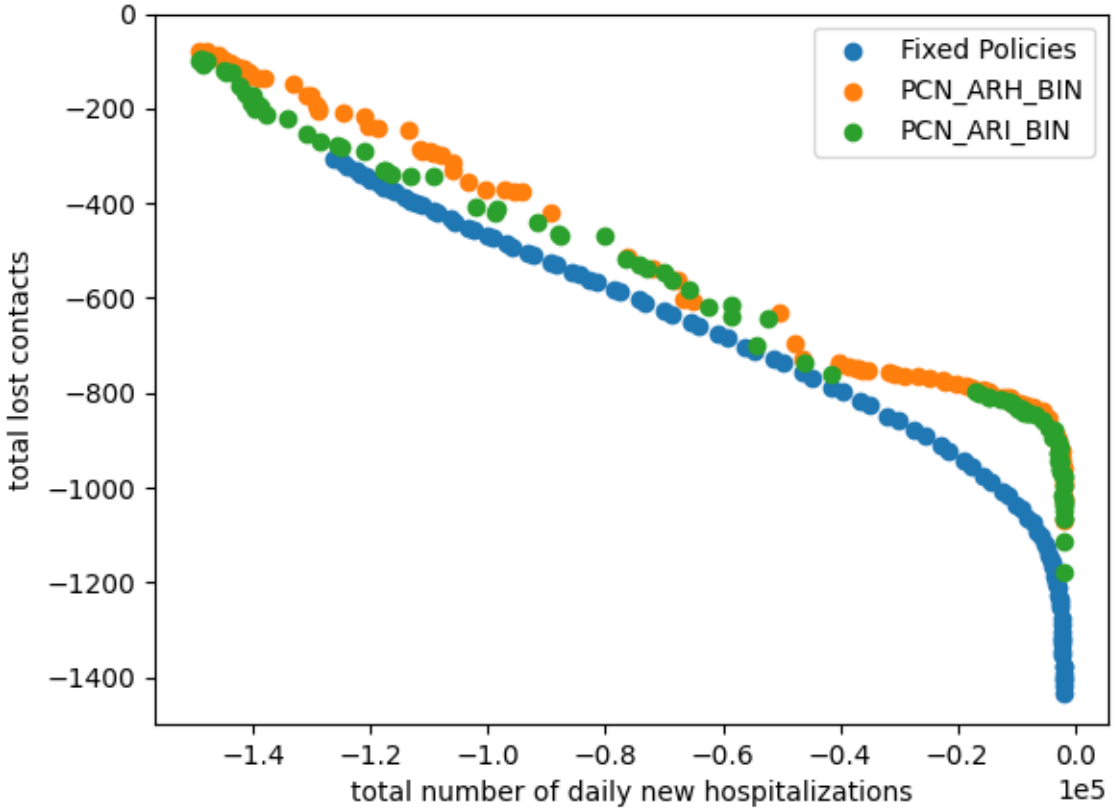


Figure 16: The Pareto front of policies discovered by PCN using the Binomial model. The plot compares the Pareto fronts of the policies by PCN and the fixed policies to find the best possible way to optimize the cumulative lost contacts (which measures social burden) and daily new hospitalizations. $PCN_{ARH}(BIN)$ and $PCN_{ARI}(BIN)$ represent the Pareto fronts of policies discovered when trained on the number of hospitalizations and infections, respectively, using the Belgian Binomial model.

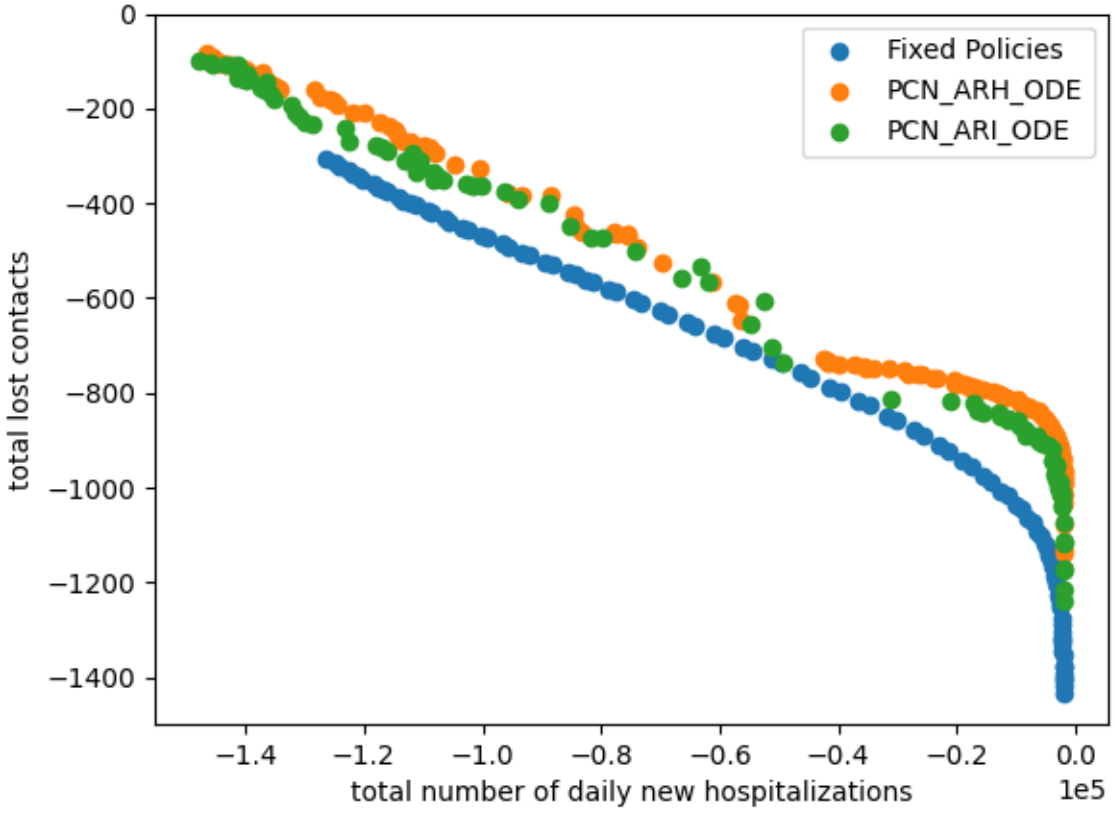


Figure 17: The Pareto front of policies discovered by PCN using the Belgian ODE model. $PCN_{ARH}(ODE)$ and $PCN_{ARI}(ODE)$ represent the Pareto fronts of policies discovered when trained on the number of hospitalizations and the number of infections, respectively, using the Belgian ODE model.

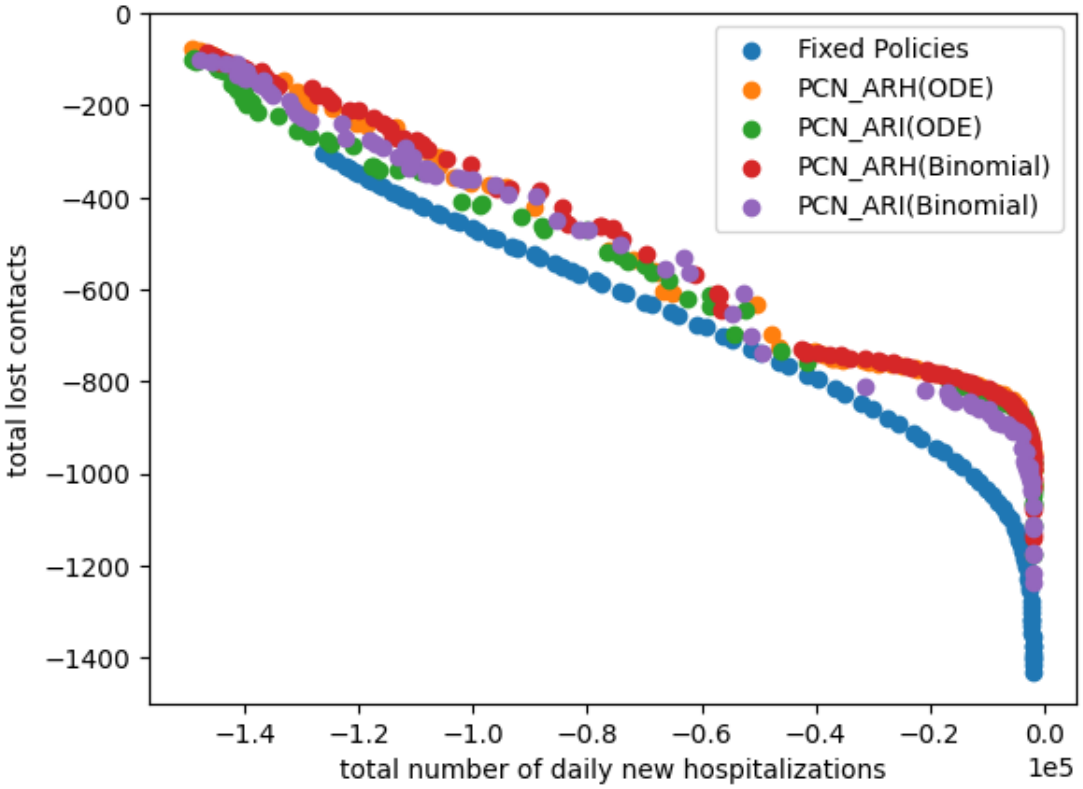


Figure 18: The Pareto front of policies discovered by PCN demonstrates the trade-offs between the number of hospitalizations and lost contacts. It presents different policy options that balance the reduction in hospitalizations with the potential decrease in social interactions.

5 Discussions

In this chapter, we will examine and interpret the results achieved in this work.

5.1 Discussion

The main objective of this study is to utilize algorithms for learning non-pharmaceutical interventions within a given COVID-19 environment. Figure 12 illustrates the Pareto plots generated by the DQN, Goal-DQN, and NSGA-II algorithms employed in this research. The focus of these algorithms is to train and select actions that minimize the cumulative effects of two costs: health cost and economy cost.

Although DQN is not traditionally used for discovering Pareto optimal solutions, it was adapted for this study by constructing a population of policies. This involved training individual DQN policies independently and subsequently merging them. The utilization of Pareto fronts, consisting of multiple DQN policies, proves particularly effective when the health cost is relatively low, specifically when the number of deaths remains under 1000. This approach leads to savings of approximately 15 billion euros compared to the utilization of Goal-DQN and NSGA-II algorithms that is free of all restrictions. The NSGA-II algorithm demonstrates strong performance in scenarios where the economic cost is low, but it struggles to effectively address situations with a low health cost as seen in Figure 13. These findings emphasize the potential benefit of employing multi-objective algorithms to discover non-pharmaceutical interventions that can effectively mitigate both the health and economic costs associated with COVID-19.

Table 4 provides an overview of the metrics utilized to assess the performance of the PCN algorithm in generating coverage sets for both the Binomial and ODE models. The results indicate that the learned policy achieves a return that is comparable to the desired return, regardless of the experimental setting. The I_ϵ metric reveals that across all policies, the decision maker is guaranteed to lose a maximum of 0.04 and 0.04 normalized returns in any of the objectives for the Binomial and ODE models, respectively. Additionally, on average, the decision maker is expected to experience a loss of 0.006 and 0.009 normalized returns, respectively.

The results depicted in Figure 14 showcase the actions chosen by the DQN algorithms during a year-long training phase in the COVID-19 environment. The figure displays the evolution of two costs over the course of the year, with red dots indicating periods when lockdown measures were enforced.

The findings suggest that the DQN algorithm initially proposes significant enforcement of

lockdown measures during the first 200 days of the year, aiming to mitigate both health and economic costs. This trend is evident in the first and second plots of Figure 14. Particularly, the second plot demonstrates a reduction in GDP loss during periods without lockdowns, while the costs on the economy increase during periods with lockdowns. These results highlight the advantages of timely and strategic implementation of lockdown measures in striking an optimal balance of the trade-off between health and economic impacts.

The GOAL-DQN algorithm was evaluated in the COVID-19 environment throughout a year-long training and evaluation period to determine the actions it selected. Figure 15 presents the results, demonstrating the evolution of two costs over the course of the year, with red dots indicating periods of lockdown enforcement. The findings suggest that the GOAL-DQN algorithm supports the implementation of year-long total lockdown measures as an effective strategy for mitigating the spread of the disease. This approach successfully minimizes the health costs while grappling with minimizing the economic impact. Furthermore, the GOAL-DQN algorithm proves effective in mitigating health costs through the utilization of targeted interventions while minimizing economic costs. Specifically, the algorithm suggests a combination of targeted lockdown measures in the early stages of the pandemic, followed by a selective relaxation of lockdowns in later stages. The first and second plots in Figure 15 provide evidence supporting this recommendation, as they demonstrate a decrease in the number of deaths during the targeted lockdown period, while the economic costs experience a minimal increase. These findings underscore the advantages of using GOAL-DQN and other algorithms to discover non-pharmaceutical interventions that can effectively address both the health and economic costs associated with COVID-19.

The results presented in Figure 16 showcase the Pareto fronts of policies discovered by the PCN algorithm using the binomial model. The primary objective is to minimize the social burden experienced by the population, quantified by the total lost contacts, and to reduce the number of daily new hospitalizations, reflecting the epidemiological goal of minimizing the attack rate. To assess the efficiency and quality of the policies identified by PCN, we compared them with a set of fixed policies consisting of 100 predetermined policies that systematically vary through every possible level of social restrictions, ranging from 0 to 1. The PCN-ARI algorithm is trained on the number of infections, and Figure 16 demonstrates that the PCN algorithm, trained on both the number of hospitalizations and infections, outperforms the fixed policies. Moreover, the PCN-ARH algorithm, trained on the number of hospitalizations, performs almost as well, exhibiting a competitive coverage set compared to the one trained on the number of infections in the binomial model. These findings emphasize the effectiveness

of the PCN algorithm in discovering policies that effectively mitigate both the social and epidemiological consequences of COVID-19.

Figure 17 presents the Pareto fronts of policies discovered by PCN and compares them to a set of fixed policies, as described earlier. The PCN-ARI algorithm was trained on the number of infections, and our findings indicate that the PCN algorithm, trained on both the number of hospitalizations and infections, outperformed the fixed policies. These results suggest that there exist more optimal policies that can be employed instead of the fixed policies (100 predetermined policies that systematically vary through every possible level of social restrictions, ranging from 0 to 1) to effectively strike a balance between the number of daily new hospitalizations and the total number of lost contacts, as shown in figure 18. Overall, our study showcases the potential of utilizing the PCN algorithm to discover non-pharmaceutical interventions in COVID-19 models and offers a promising opportunity for future research to further explore the utilization of machine learning in mitigating the adverse consequences of the pandemic.

6 Conclusion

The utilization of algorithms for learning non-pharmaceutical interventions in the context of COVID-19 presents a promising approach to mitigating the negative impacts of the pandemic. Our study successfully demonstrated the effectiveness of employing DQN, Goal-DQN, and NSGA-II algorithms in discovering Pareto optimal solutions for minimizing both health and economic costs. Notably, we discovered that the use of a population of DQN policies yielded significant cost savings compared to employing NSGA-II and Goal-DQN without any constraints. Furthermore, our observations revealed that NSGA-II performed well in scenarios with low economic costs but exhibited limitations in scenarios with low health costs. Conversely, the adoption of multiple DQN policies proved more effective in situations characterized by low health costs. These findings underscore the potential of utilizing a combination of algorithms to identify interventions that can effectively address the social and epidemiological consequences of COVID-19.

In this study, the Pareto Conditioned Network (PCN) algorithm is a promising approach to solving multi-objective reinforcement learning (MORL) problems. The algorithm is designed to produce coverage sets for the Binomial and Ordinary Differential Equation (ODE) models while maximizing desired returns and minimizing losses across multiple objectives. Our evaluation of the PCN algorithm showed that it performs well across a range of settings, with the decision maker losing at worst only 0.04 normalized returns in any of the objectives for both ODE and Binomial versions. On average, the decision maker lost only 0.006 and 0.009 normalized returns for the ODE and Binomial versions, respectively. Overall, the PCN algorithm shows great potential for addressing complex MORL problems in real-world settings, particularly in cases where the objectives are difficult to quantify or trade-offs need to be made between different objectives.

There are several potential areas for future research to further advance the findings of this study. Firstly, exploring the utilization of alternative machine learning algorithms, including various reinforcement learning and evolutionary algorithms, or investigating hybrid approaches that combine multiple algorithms, would be valuable. Secondly, investigating the incorporation of additional variables, such as vaccination rates, age demographics, or geographical location, into more sophisticated models could enhance the understanding of non-pharmaceutical interventions.

Thirdly, evaluating the transferability and effectiveness of the discovered policies across different COVID-19 environments would provide insights into their applicability in diverse settings. Fourthly, analyzing the efficiency of DQN, NSGA-II, and Goal-DQN in Case 2, while assessing the performance of PCN in Case 1, and subsequently comparing the outcomes of these

algorithms would yield valuable insights on their respective effectiveness.

Lastly, examining the impact of these interventions on other societal factors, such as mental health, education, or the environment, would provide a comprehensive understanding of their broader consequences. Overall, the findings of this study open up promising avenues for future research, enabling the utilization of machine learning algorithms to effectively combat the challenges posed by the COVID-19 pandemic.

References

World Health Organization et al. Non-pharmaceutical public health measures for mitigating the risk and impact of epidemic and pandemic influenza: annex: report of systematic literature reviews. Technical report, World Health Organization, 2019.

World Health Organization. Who coronavirus (covid-19) dashboard, 2023. URL https://covid19.who.int/.

World Health Organization. Coronavirus disease (covid-19) pandemic, 2020. URL http://www.euro.who.int/en/health-topics/health-emergencies/international-health-regulations/news/news/2020/2/2019-ncov-outbreak-is-an-emergency-of-international-concern.

Quentin Richard, Samuel Alizon, Marc Choisy, Mircea T Sofonea, and Ramsès Djidjou-Demasse. Age-structured non-pharmaceutical interventions for optimal control of covid-19 epidemic. *PLoS computational biology*, 17(3):e1008776, 2021.

Markus Kantner and Thomas Koprucki. Beyond just "flattening the curve": Optimal control of epidemics with purely non-pharmaceutical interventions. *Journal of Mathematics in Industry*, 10(1):1–23, 2020.

M Irfan Uddin, Syed Atif Ali Shah, Mahmoud Ahmad Al-Khasawneh, Ala Abdulsalam Alarood, and Eesa Alsolami. Optimal policy learning for covid-19 prevention using reinforcement learning. *Journal of Information Science*, page 0165551520959798, 2020.

Gloria Hyunjung Kwak, Lowell Ling, and Pan Hui. Deep reinforcement learning approaches for global public health strategies for covid-19 pandemic. *Plos one*, 16(5):e0251550, 2021.

Folashade B Agusto, Nizar Marcus, and Kaseem O Okosun. Application of optimal control to the epidemiology of malaria. 2012.

Olena Kostylenko, Helena Sofia Rodrigues, and Delfim FM Torres. Banking risk as an epidemiological model: An optimal control approach. In *Congress of APDIO, the Portuguese Operational Research Society*, pages 165–176. Springer, 2017.

Mathieu Reymond, Conor F Hayes, Lander Willem, Roxana Rădulescu, Steven Abrams, Diederik M Roijers, Enda Howley, Patrick Mannion, Niel Hens, Ann Nowé, et al. Exploring the pareto front of multi-objective covid-19 mitigation policies using reinforcement learning. *arXiv preprint arXiv:2204.05027*, 2022a.

Cédric Colas, Boris Hejblum, Sébastien Rouillon, Rodolphe Thiébaut, Pierre-Yves Oudeyer, Clément Moulin-Frier, and Mélanie Prague. Epidemioptim: A toolbox for the optimization of control policies in epidemiological models. *Journal of Artificial Intelligence Research*, 71: 479–519, 2021.

Stella Quah. *International encyclopedia of public health*. Academic Press, 2016.

Anderson Luiz Pena da Costa, Orlando Alves Rodrigues Neto, and Antonio Carlos Souza Silva-Júnior. Conditioners of the infectious diseases dynamics. *Estação Científica (UNIFAP)*, 8(3):09–23, 2019.

Jonathan R Davis and Joshua Lederberg. Emerging infectious diseases from the global to the local perspective: a summary of a workshop of the forum on emerging infections. 2001.

Allen GP Ross, Suzanne M Crowe, and Mark W Tyndall. Planning for the next global pandemic, 2015.

Justin Lessler and Derek AT Cummings. Mechanistic models of infectious disease and their impact on public health. *American journal of epidemiology*, 183(5):415–422, 2016.

Kathleen A Alexander, Bryan L Lewis, Madhav Marathe, Stephen Eubank, and Jason K Blackburn. Modeling of wildlife-associated zoonoses: applications and caveats. *Vector-borne and zoonotic diseases*, 12(12):1005–1018, 2012.

Pratha Sah, Michael Otterstatter, Stephan T Leu, Sivan Leviyang, and Shweta Bansal. Revealing mechanisms of infectious disease spread through empirical contact networks. *PLoS computational biology*, 17(12):e1009604, 2021.

Katherine Royce and Feng Fu. Mathematically modeling spillovers of an emerging infectious zoonosis with an intermediate host. *PloS one*, 15(8):e0237780, 2020.

Maria D Van Kerkhove and Neil M Ferguson. Epidemic and intervention modelling: a scientific rationale for policy decisions? lessons from the 2009 influenza pandemic. *Bulletin of the World Health Organization*, 90(4):306–310, 2012.

Bo Xu, Bernardo Gutierrez, Sumiko Mekaru, Kara Sewalk, Lauren Goodwin, Alyssa Loskill, Emily L Cohn, Yulin Hswen, Sarah C Hill, Maria M Cobo, et al. Epidemiological data from the covid-19 outbreak, real-time case information. *Scientific data*, 7(1):1–6, 2020a.

Aaron Aruna, Placide Mbala, Luigi Minikulu, Daniel Mukadi, Dorothée Bulemfu, Franck Edidi, Junior Bulabula, Gaston Tshapenda, Justus Nsio, Richard Kitenge, et al. Ebola virus

disease outbreak—democratic republic of the congo, august 2018–november 2019. *Morbidity and Mortality Weekly Report*, 68(50):1162, 2019.

Marguerite Massinga Loembé, Akhona Tshangela, Stephanie J Salyer, Jay K Varma, Ahmed E Ogwell Ouma, and John N Nkengasong. Covid-19 in africa: the spread and response. *Nature Medicine*, 26(7):999–1003, 2020.

Peipei Wang, Xinqi Zheng, and Haiyan Liu. Simulation and forecasting models of covid-19 taking into account spatio-temporal dynamic characteristics: A review. *Frontiers in Public Health*, 10, 2022.

Jayanta Mondal and Subhas Khajanchi. Mathematical modeling and optimal intervention strategies of the covid-19 outbreak. *Nonlinear dynamics*, pages 1–26, 2022.

George Macdonald et al. The epidemiology and control of malaria. *The Epidemiology and Control of Malaria.*, 1957.

Oliver Bent, Sekou Remy, Stephen Roberts, and Aisha Walcott-Bryant. Novel exploration techniques (nets) for malaria policy interventions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Ingemar Nasell and Warren M Hirsch. Mathematical models of some parasitic diseases involving an intermediate host. Technical report, NEW YORK UNIV NY COURANT INST OF MATHEMATICAL SCIENCES, 1971.

Maurice S Bartlett. The critical community size for measles in the united states. *Journal of the Royal Statistical Society: Series A (General)*, 123(1):37–44, 1960.

H Muench. Catalytic models in epidemiology." harvard univ. press, cambridge, massachusetts. 1959.

Lila Elveback, John P Fox, André Varma, et al. An extension of the reed-frost epidemic model for the study of competition between viral agents in the presence of interference. *American journal of hygiene*, 80(3):356–64, 1964.

HT Waaler and MA Piot. The use of an epidemiological model for estimating the effectiveness of tuberculosis control measures: sensitivity of the effectiveness of tuberculosis control measures to the coverage of the population. *Bulletin of the World Health Organization*, 41 (1):75, 1969.

Giulia Giordano, Franco Blanchini, Raffaele Bruno, Patrizio Colaneri, Alessandro Di Filippo, Angela Di Matteo, and Marta Colaneri. Modelling the covid-19 epidemic and implementation of population-wide interventions in italy. *Nature medicine*, 26(6):855–860, 2020.

Julien Arino and Stéphanie Portet. A simple model for covid-19. *Infectious Disease Modelling*, 5:309–315, 2020.

Sarbaz HA Khoshnaw, Muhammad Shahzad, Mehboob Ali, and Faisal Sultan. A quantitative and qualitative analysis of the covid–19 pandemic model. *Chaos, Solitons & Fractals*, 138: 109932, 2020.

Giuseppe C Calafiore, Carlo Novara, and Corrado Possieri. A time-varying sird model for the covid-19 contagion in italy. *Annual reviews in control*, 50:361–372, 2020.

Abdel R Omram. The epidemiologic transition: a theory of the epidemiology of population change. *Bulletin of the World Health Organization*, 79(2):161–170, 2001.

Jeffrey Schank and Charles Twardy. *Mathematical Models*, volume 6 of *The Cambridge History of Science*, page 416–431. Cambridge University Press, 2009. doi: 10.1017/CHOL9780521572019.023.

Fred Brauer. Mathematical epidemiology: Past, present, and future. *Infectious Disease Modelling*, 2(2):113–127, 2017.

William Ogilvy Kermack and Anderson G McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

Md Haider Ali Biswas, Luís Tiago Paiva, and MDR De Pinho. A seir model for control of infectious diseases with constraints. *Mathematical Biosciences & Engineering*, 11(4):761, 2014.

Masaya M Saito, Seiya Imoto, Rui Yamaguchi, Hiroki Sato, Haruka Nakada, Masahiro Kami, Satoru Miyano, and Tomoyuki Higuchi. Extension and verification of the seir model on the 2009 influenza a (h1n1) pandemic in japan. *Mathematical biosciences*, 246(1):47–54, 2013.

Michael Y Li, John R Graef, Liancheng Wang, and János Karsai. Global dynamics of a seir model with varying total population size. *Mathematical biosciences*, 160(2):191–213, 1999.

Phenyo E Lekone and Bärbel F Finkenstädt. Statistical inference in a stochastic epidemic seir model with control intervention: Ebola as a case study. *Biometrics*, 62(4):1170–1177, 2006.

Nita H Shah and Jyoti Gupta. Seir model and simulation for vector borne diseases. 2013.

Anca Radulescu, Cassandra Williams, and Kieran Cavanagh. Management strategies in a seir model of covid 19 community spread. *arXiv preprint arXiv:2003.11150*, 2020.

Samuel Mwalili, Mark Kimathi, Viona Ojiambo, Duncan Gathungu, and Rachel Mbogo. Seir model for covid-19 dynamics incorporating the environment and social distancing. *BMC Research Notes*, 13(1):1–5, 2020.

Leonardo López and Xavier Rodo. A modified seir model to predict the covid-19 outbreak in spain and italy: simulating control scenarios and multi-scale epidemics. *Results in Physics*, 21:103746, 2021.

Sharif Noor Zisad, Mohammad Shahadat Hossain, Mohammed Sazzad Hossain, and Karl Andersson. An integrated neural network and seir model to predict covid-19. *Algorithms*, 14(3):94, 2021.

Michael Y Li and James S Muldowney. Global stability for the seir model in epidemiology. *Mathematical biosciences*, 125(2):155–164, 1995.

Paul Diaz, Paul Constantine, Kelsey Kalmbach, Eric Jones, and Stephen Pankavich. A modified seir model for the spread of ebola in western africa and metrics for resource allocation. *Applied Mathematics and Computation*, 324:141–155, 2018.

Shaobo He, Yuexi Peng, and Kehui Sun. Seir modeling of the covid-19 and its dynamics. *Nonlinear dynamics*, 101(3):1667–1680, 2020.

Side Syafruddin and MSM Noorani. Seir model for transmission of dengue fever in selangor malaysia. In *International Journal of Modern Physics Conference Series*, volume 9, pages 380–389, 2012.

Erivelton Geraldo NEPOMUCENO, Ricardo Hiroshi Caldeira TAKAHASHI, and Luis Antonio AGUIRRE. Individual-based model (ibm): an alternative framework for epidemiological compartment models. *Revista Brasileira de Biometria*, 34(1):133–162, 2016.

Carl Orge Retzlaff, Martina Ziefle, and André Calero Valdez. The history of agent-based modeling in the social sciences. In *International Conference on Human-Computer Interaction*, pages 304–319. Springer, 2021.

Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the national academy of sciences*, 99(suppl 3):7280–7287, 2002.

Joshua M Epstein, Derek AT Cummings, Shubha Chakravarty, Ramesh M Singa, and Donald S Burke. Toward a containment strategy for smallpox bioterror: an individual-based computational approach. *Brookings institution, CSED working paper*, (31), 2002.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Edward L Thorndike. The law of effect. *The American journal of psychology*, 39(1/4):212–222, 1927.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018a.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 2nd edition, 2018b. ISBN 0262193981.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Wenyu Zhang, Jingyao Gai, Zhigang Zhang, Lie Tang, Qingxi Liao, and Youchun Ding. Double-dqn based path smoothing and tracking control method for robotic vehicle navigation. *Computers and Electronics in Agriculture*, 166:104985, 2019.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. a bradford book, cambridge, massachusetts, november 2018. *URL http://www. incompleteideas. net/book/the-book-2nd. html*.

Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021.

Andrea Yanez, Conor Hayes, and Frank Glavin. Towards the control of epidemic spread: Designing reinforcement learning environments. In *AICS*, pages 188–199, 2019.

Mathieu Reymond, Eugenio Bargiacchi, and Ann Nowé. Pareto conditioned networks. *arXiv preprint arXiv:2204.05036*, 2022b.

Harshad Khadilkar, Tanuja Ganu, and Deva P Seetharam. Optimising lockdown policies for epidemic control using reinforcement learning. *Transactions of the Indian National Academy of Engineering*, 5(2):129–132, 2020.

Varun Kompella, Roberto Capobianco, Stacy Jong, Jonathan Browne, Spencer Fox, Lauren Meyers, Peter Wurman, and Peter Stone. Reinforcement learning for optimization of covid-19 mitigation policies. *arXiv preprint arXiv:2010.10560*, 2020.

Seyedeh N Khatami and Chaitra Gopalappa. A reinforcement learning model to inform optimal decision paths for hiv elimination. *Mathematical biosciences and engineering: MBE*, 18(6): 7666, 2021.

Nicola Perra. Non-pharmaceutical interventions during the covid-19 pandemic: A review. *Physics Reports*, 913:1–52, 2021.

Yufan Zhao, Michael R Kosorok, and Donglin Zeng. Reinforcement learning design for cancer clinical trials. *Statistics in medicine*, 28(26):3294–3315, 2009.

Pieter JK Libin, Arno Moonens, Timothy Verstraeten, Fabian Perez-Sanjines, Niel Hens, Philippe Lemey, and Ann Nowé. Deep reinforcement learning for large-scale epidemic control. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 155–170. Springer, 2020.

R Lakshmana Kumar, Firoz Khan, Sadia Din, Shahab S Band, Amir Mosavi, and Ebuka Ibeke. Recurrent neural network and reinforcement learning model for covid-19 prediction. *Frontiers in public health*, 9, 2021.

Eelke van der Horst, Patricia Marqués-Gallego, Thea Mulder-Krieger, Jacobus van Veldhoven, Johannes Kruisselbrink, Alexander Aleman, Michael TM Emmerich, Johannes Brussee, Andreas Bender, and Adriaan P IJzerman. Multi-objective evolutionary design of adenosine receptor ligands. *Journal of chemical information and modeling*, 52(7):1713–1721, 2012.

Susanne Rosenthal and Markus Borschbach. Design perspectives of an evolutionary process for multi-objective molecular optimization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 529–544. Springer, 2017.

Abhijit Dey and Amalendu Bikash Choudhury. A comparative study between scalarization approach and pareto approach for multi-objective optimization problem using genetic algorithm (moga) formulated based on superconducting fault current limiter. In *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, pages 1–4. IEEE, 2016.

Gilbert Syswerda. The application of genetic algorithms to resource scheduling. In *Proc. of ICGA-91*, 1991.

Wilfried Jakob, Martina Gorges-Schleuter, and Christian Blume. Application of genetic algorithms to task planning and learning. In *PPSN*, pages 293–302, 1992.

Gareth Jones, Robert D Brown, David E Clark, Peter Willett, and Robert C Glen. Searching databases of two-dimensional and three-dimensional chemical structures using genetic

algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 597–602, 1993.

PB Wilson and MD Macleod. Low implementation cost iir digital filter design using genetic algorithms. In *IEE/IEEE workshop on natural algorithms in signal processing*, volume 1, pages 1–4. Chelmsford, UK, 1993.

Kenneth Chircop and David Zammit-Mangion. On-constraint based methods for the generation of pareto frontiers. *J. Mech. Eng. Autom*, 3(5):279–289, 2013.

Domenico Quagliarella. Coupling genetic algorithms and gradient based optimization techniques. *Genetic algorithms in engineering and computer science*, 14, 1998.

S Ranjithan, JW Eheart, and J Liebman. Incorporating fixed-cost component of pumping into stochastic groundwater management: A genetic algorithm-based optimization approach. *Eos Trans. AGU*, 73:14, 1992.

Abraham Charnes and William Wager Cooper. Management models and industrial applications of linear programming. *Management science*, 4(1):38–91, 1957.

Yuji Ijiri. *Management goals and accounting for control*, volume 3. North Holland Publishing Company, 1965.

Dietrich Wienke, Carlos Lucasius, and Gerrit Kateman. Multicriteria target vector optimization of analytical procedures using a genetic algorithm: Part i. theory, numerical simulations and application to atomic emission spectroscopy. *Analytica Chimica Acta*, 265(2):211–225, 1992.

Hojjat Adeli. *Advances in design optimization*. CRC press, 1994.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2):117–132, 2003.

Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3–34. Springer, 2011.

Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*, 2016.

Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *International Conference on Machine Learning*, pages 10607–10616. PMLR, 2020b.

Scvab Igor. Optimization modelling in python: Multiple objectives, May 2021. URL https://codemonk.in/blog/a-gentle-introduction-to-multi-objective-optimization/. [Online; posted 30-May-2021].

Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning*, 84(1):51–80, 2011.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

C Poloni, D Quagliarella, J Périaux, N Gauger, and K Giannakoglou. An fpga-based approach to multi-objective evolutionary algorithm for multi-disciplinary design optimisation.

Moritz UG Kraemer, Chia-Hung Yang, Bernardo Gutierrez, Chieh-Hsi Wu, Brennan Klein, David M Pigott, Open COVID-19 Data Working Group†, Louis Du Plessis, Nuno R Faria, Ruoran Li, et al. The effect of human mobility and control measures on the covid-19 epidemic in china. *Science*, 368(6490):493–497, 2020.

Antoine Mandel and Vipin Veetil. The economic cost of covid lockdowns: an out-of-equilibrium analysis. *Economics of Disasters and Climate Change*, 4:431–451, 2020.

Mélanie Prague, Linda Wittkop, A Collin, Q Clairon, D Dutartre, P Moireau, R Thiebaut, and BP Hejblum. Population modeling of early covid-19 epidemic dynamics in french regions and estimation of the lockdown impact on infection rate. medrxiv, page 2020.04. 21.20073536. *Google Scholar*, 2020.

Steven Abrams, James Wambua, Eva Santermans, Lander Willem, Elise Kuylen, Pietro Coletti, Pieter Libin, Christel Faes, Oana Petrof, Sereina A Herzog, et al. Modelling the early phase

of the belgian covid-19 epidemic using a stochastic compartmental model and studying its implied future trajectories. *Epidemics*, 35:100449, 2021.

Pieter JK Libin, Arno Moonens, Timothy Verstraeten, Fabian Perez-Sanjines, Niel Hens, Philippe Lemey, and Ann Nowé. Deep reinforcement learning for large-scale epidemic control. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 155–170. Springer, 2021.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pages 507–517. PMLR, 2020.