2022 7th International Conference on Data Science and Machine Learning Applications (CDMA) | 978-1-6654-1014-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/CDMA54072.2022.00007

Towards Using Deep Reinforcement Learning for Better COVID-19 Vaccine Distribution Strategies

Fouad TRAD
Computer and Communication Engineering
Lebanese University
Ras Maska, Al-Koura, Lebanon
fouad.trad@ul.edu.lb

Salah EL FALOU
Computer and Communication Engineering
Lebanese University
Ras Maska, Al-Koura, Lebanon
salah.elfalou@ul.edu.lb

***Abstract*— Vaccination has been the most promising hope to get back to normal ever since the COVID-19 outbreak started. But as promising as this sounds, vaccinating all of the population at the same time is practically infeasible because of the limited supply of vaccines from one side and the high demand from the other side. So, the process cannot happen overnight, and this is why governments kept thinking about how they can distribute vaccines in a way that helps their citizens get back to normal with the least possible damages (infections and deaths). In this study, we investigate how Reinforcement Learning (RL) can be used to distribute vaccines more efficiently among the citizens of a country, given their age and profession. For this reason, we created an RL agent that learns vaccine distribution strategies through its interaction with a Monte Carlo (MC) simulation environment that we built. This environment runs an Agent-Based Model (ABM) where we have agents interacting with each other and with the environment where they live and based on their behavior, the virus will spread. The goal of the RL agent was to find vaccine distribution strategies that would minimize the number of infections and deaths in the environment where our agents live. After training our RL agent for 100 episodes, we compared the best strategy that RL gave us with some of the well-known strategies that countries adopt, and we found that the RL strategy outperformed them.**

Keywords—Reinforcement Learning, Agent-Based Modeling, COVID-19, Vaccination, Monte Carlo Simulation

I. Introduction

Ever since the vaccines started rolling out, all governments have been trying to find the best way to distribute them to their citizens. Some countries like the United States and Berlin wanted to vaccinate elderly people (aged above 60 years old) before the younger ones, because old people, when infected, are more exposed to hospital care and death, while others like Indonesia decided to start vaccinating workers and adults (aged between 18 and 59 years old) because these categories are more exposed to catching the virus in the first place. Some countries like Germany and Spain altered their strategies after the "return to schools" announcement for the academic year 2021-2022, and they wanted to start including children (aged between 12 and 17 years old) in the vaccination process to stay on the safe side and avoid a new outbreak. So, the strategies are varied, and all of them would work well because vaccination – regardless of the adopted strategy – is supposed to help reach herd immunity faster and accelerate the "return to normal" process. But how can we be sure that these adopted strategies are the best ones? What if there is another strategy that might lead to better outcomes and help us get back to normal with less human damages? Wouldn't it be better to adopt it? Definitely yes, and this is our goal from this study. We need to show how RL can be applied to solve such decision-making problems while leading to better outcomes. Two Previous studies [1], [2] have used RL to optimize COVID-19 vaccine distribution, but their main focus was to find which country or region should receive more resources depending on the pandemic situation in each of them. What we are proposing in our study is to find, using RL, how we can distribute the limited supply of vaccines across the citizens of a country depending on their age and profession in an optimal way that minimizes deaths and infections. But in order to do this, an RL agent has to interact with an environment. This is why we used the simulator that we built and validated in [3] as a simulation environment. This system consists of an ABM where we have agents (representing people) interacting with each other and with their environment (the country where they live). The agents can be infected with COVID-19 and based on their behavior, they might infect other agents as well, and infections might lead to death, just like what happens in real life. It is true that agents can get infected, but agents can also get vaccinated, and vaccination will reduce the virus spread in the country where the agents live. To vaccinate these agents, our simulator allows us to use defined strategies that we can choose, but it can also leverage the magic behind RL and train an RL agent to find better vaccination strategies. RL has been used before to solve many complex problems that involve decision-making like robotics and healthcare, and it shined in the world of gaming. For this reason, we framed the problem that we have as a game where the RL agent has to learn which group to vaccinate first. Just like any other game, the player seeks to get a higher score, and in our system, the lower the number of infections and deaths is, the higher the score the player would get. This is why the RL agent would learn to vaccinate agents in a way that keeps infections and deaths as low as possible to get the highest possible reward. After training our RL agent on this task, we compared the best strategy we obtained with two other defined strategies: The "younger first" strategy (where we start vaccinating younger ages and then move to older ones), and the "older first" strategy (where we start vaccinating older people and then move to the younger ones). We saw that the RL agent has found a strategy that outperformed these two.

The rest of the paper will be organized as follows: In section 2, we will introduce Agent-Based Modeling which is the base of our simulator. Then, in section 3, we will talk about our simulation environment in detail to understand how it functions. Section 4 will introduce the concept behind Reinforcement Learning and its main algorithms. Sections 5 and 6 will talk about how we used RL to solve our problem. Section 7 contains the results that we obtained, and finally, section 8 contains a conclusion and some future works.

II. Agent-based modeling

Agent-Based Modeling is an Artificial Intelligence technique that relies on Multi-Agent Systems. In this modeling technique, a system is represented by its micro-components (which are the agents), and based on the behavior and the interaction of these agents with each other and with their environment, a system will evolve. So, the higher system properties are just a result of the agents' interactions.

This technique is a go-to when we need to model complex systems that are hard to model using pure mathematical equations. Note that even the simplest ABM can exhibit complex behavior patterns and provide valuable information about the dynamics of the real-world system that it emulates. In our study, we used agent-based modeling to simulate the spread of COVID-19 in a virtual country where the local interactions between the agents will influence the pandemic spread. The parameters of an ABM are defined by the characteristics, dynamic states, and behaviors of the agents, but how can we determine them? In our study, we will be dealing with statistical distributions that help us determine the properties, actions, and evolving states of our agents. For this purpose, we use the MC algorithm to sample from those distributions. So, our ABM runs on top of the MC algorithm.

III. Simulation Environment

The goal of our simulator is to forecast the evolution of COVID-19 inside of a country with the presence of different control measures and different vaccine distribution strategies. The good thing about the simulator is that it can give an idea about how the pandemic situation can become after a decision is made. This way governments can benefit from it while seeing the effect of their actions before doing them, and then adapt the best measures and strategies to limit the virus spread, and save a lot of trouble for their citizens. The four key components in our simulator are the virus, the vaccine, the country, and the agents. We will discuss each of them and how they are connected in the following parts.

A. The Virus

The virus that we want to simulate in our study is the SARS-CoV-2, but other viruses can be simulated as well simply by changing the parameters. Those parameters are the latent period, the incubation period, the infectious period, and the different kinds of symptoms. Once an agent becomes infected, his states will evolve according to the kind of symptoms that he will develop [4], [5], [6], [7] (Fig. 1). An agent will develop a particular symptom according to his age. To determine which kind of symptoms an agent will develop, we use the MC algorithm to sample from the distribution of symptoms vs age range that we derived from [8] and is illustrated in Table I.

TABLE I Symptoms distribution across age ranges [8]

Age Group	Asymptomatic	Mild/Moderate	Severe	Critical
0 – 17	43%	57%	0%	0%
18 – 39	14%	79%	5%	2%
40 – 59	7%	77%	10%	6%
60 – 79	3%	50%	33%	14%
80 +	0%	33%	17%	50%

B. The Vaccine

In our simulator, we can supply different kinds of vaccines at different times for different clusters of the population. The characteristics of a vaccine can be summarized by the number of doses, the time between the doses, the transmission risk (how probably a vaccinated person transmits the virus), the infection risk (how probably a vaccinated person catches the virus) after each dose, and the time that each dose takes to become fully effective. In our study, we only used The Pfizer vaccine. This vaccine is characterized by 2 doses separated by 21 days, with respective infection risks of 48% and 5%. The time until the first dose becomes effective is 12 days, and the time until the second dose becomes effective is 7 days [9], [10]. In the Pfizer case, the risk of transmission after one dose is between 45-50% according to England Public Health reports [11], and after 2 doses it is still not determined. In our study, we assumed the risk of transmission to be 50% after taking the first dose, and 20% after the second dose. These findings are represented in Fig.2.

C. The Country

A country is defined by different locations where an agent might go (houses, hospitals, markets, malls, restaurants, nightclubs, companies, praying locations, universities, and schools, and even an airport). In addition to that, a country is characterized by specific age distribution and a house population distribution. The country we will be simulating in our study will have a Lebanon-like structure, and this is why we chose to build it using real Lebanese distributions that are represented in tables II and III. Note that other countries can be simulated as well, simply by applying their corresponding distributions.

TABLE II House Population Distribution

Household Size	1	2	3	4	5	6	7	8
Percentage (%)	10	18	18	20	16	10	4	4

TABLE III Age Distribution

Age Range	*Min*	0	5	10	20	30	40	50	60	70	80
	Max	4	9	19	29	39	49	59	69	79	89
Percentage (%)		8	8	16	17	13	11	11	8	5	3

D. The Agent

The agent is defined by several characteristics when created at the beginning of the simulation. These characteristics include an age, a personal house, a profession and some locations where he can possibly go. The profession's

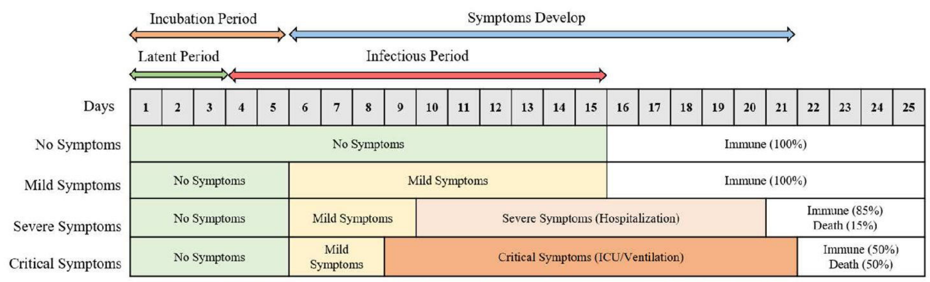
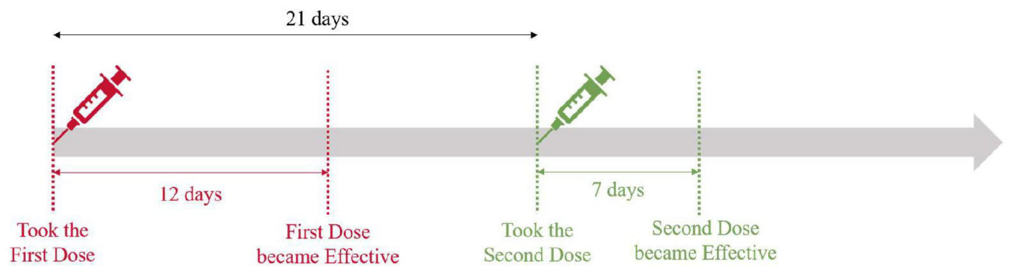


Fig. 1 COVID-19 Symptoms with their evolution through days [4], [5], [6], [7]



Infection Risk	100%	48%	5%
Transmission Risk	100%	50%	20%

Fig. 2 Pfizer Vaccine Characteristics [9], [10], [11]

distribution is also based on Lebanese society and is represented in Table IV. Besides his work and study locations, an agent can go to random locations like restaurants, markets, companies, hospitals, etc. One thing to note is that the study location and the work location for an agent are fixed, which means he has to go there every day (or at least when they are open). But for the other locations, he can visit them during random times where he has nothing to do. For example, a Hospital worker does not have to go to the market every single day but he should go to the hospital each day during the week.

TABLE IV Profession Distribution

Age Range	***Min***	0	5	10	20	30	40	50	60	70
	Max	4	9	19	29	39	49	59	69	79
School Students (%)		26	92	71	2	0	0	0	0	0
University Students (%)		0	0	9	15	0	0	0	0	0
Workers (%)		0	0	7	52	62	57	47	31	16

E. Building the Environment

The process of building the environment happens according to a specific flow. First thing, the Locations are built. Among these locations, Houses are the origins of the population because this is where an agent is born. Inside each house, we use MC Algorithm to decide the number of people that should live inside of it according to the House Population Distribution. After knowing how many people live in the house, we need to know their ages. Based on the ages we determine other characteristics (profession, locations the agent visits, etc.). After we have set up our agents, let's see how they behave during the simulation.

F. Simulation process

The simulation starts with some agents being infected, and the others are susceptible (can become infected). The infected agents will go into the world, interact with other agents, and eventually infect them. This is how the number of infections grows. Once the simulation starts, there is a specific number of agents that register for the vaccine, so that when it becomes available they can have it. At any time during the simulation, we can supply a specific quantity of vaccines for a specified group of the population (according to the age range or the profession). These vaccines will be given only to people who have registered at the beginning. The simulation process runs for several days. At the beginning of each day, an agent is assigned a list of locations where he will be during each hour of the day (like a schedule). This list is given based on the profession of the agent (it should include his fixed locations), and on the locations where he can go (e.g. markets, malls, etc). This is how agents meet at different locations. When an agent is present in a specific location, he is considered to be in a room inside of this location. This is why different locations are modeled as different rooms, each of them having specific characteristics, and based on that, each location has a defined infection rate. This rate indicates the chance that an infected agent infects another agent within the same location (or room). In our study, we used a COVID-19 risk calculator developed by the Harvard School of Public Health that is based on the peer-reviewed paper of Azimi *et al.* [12]. This calculator helped us approximate the infection rate for different kinds of rooms where an agent might be. This risk percentage takes into consideration a lot of factors related to the room: its space, the activity done inside it, the quality of the HVAC system, the time spent in the room, the fact that people in the room are wearing masks or not, the amount of social distancing, etc. So, each of our locations was modeled as a room that has specific characteristics, and based on that we got the infection rate of each of them. But how do we decide if an agent should become infected once he is in a specific location with another infectious agent?

When Agent A (who is infectious) and Agent B (who is susceptible) are present in the same room, we decide if A will infect B according to the following process. First, we use the MC Algorithm to see if Agent A should transmit the virus according to the transmission risk associated with the vaccine he has taken (if he has taken one). If A is supposed to transmit, we use the MC algorithm again to determine if according to the infection rate of the room, Agent B will become infected or not. If yes, we use the MC algorithm a third time to determine if Agent B will truly become infected according to the infection risk associated with the vaccine that he took (if he has taken one). This is how the vaccine can protect an agent by giving him a second chance if he was supposed to become infected. The process of applying the triple MC can be well understood from Fig. 3.

In this study, we aim to use RL to distribute the vaccines in a way that minimizes cumulative infections and deaths. This way, RL is supposed to give us better strategies that lead to better outcomes. This is why, in the next section we will explain the foundations of RL and after it, we will see how we trained an RL agent to distribute a limited number of vaccine shots in a way that reduces the number of infections and deaths in the simulation environment.

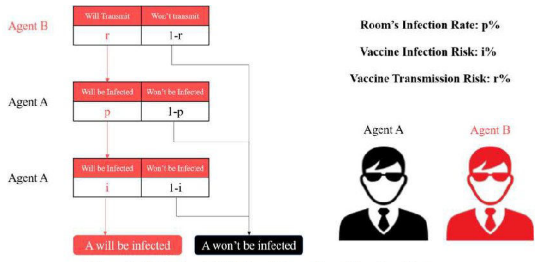


Fig. 3 Triple MC for Vaccine Protection

IV. Reinforcement Learning

RL is one of the oldest and most challenging kinds of Machine Learning where we have an agent learning to achieve a particular task by interacting with an environment. The RL agent starts by being in a particular state that the environment determines. The agent chooses an action to take based on the state he is in, and after that, the environment will respond with a reward (or a penalty) depending on whether the action was good or bad and provides the agent with a new state, and the process continues like this. The goal of the agent is to maximize the rewards it is getting from the environment through the set of interactions. So, to formulate an RL problem, we need an agent, an environment, states, actions, and rewards. The RL process is illustrated in Fig.4.

In RL, we have the concept of what we call an "Episode". An episode mainly characterizes tasks that can finish across time. In episodic tasks, the interaction between the agent and the environment does not go forever, there will be an end state at some point. In games, for example, an episode is between the time you start playing, and the time you finish (by winning or losing). The goal of an agent is to maximize the total reward he is receiving during his interaction with the environment. The total reward at a time step t (1) consists of an immediate reward (after performing an action) and a discount for future rewards (that correspond to future actions). The discount factor γ indicates how much attention the agent is paying attention to future rewards.

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots = R_t + \gamma G_{t+1} \qquad (1)$$

This is why the agent should know beforehand what would be the expected return that results from doing a particular action "*a*" while being in a state "*s*" so that he chooses the right action accordingly. The Q-value (or action-value function) calculates the expected return given that the agent is in state "*s"* and performs action "*a"* (2). This is why, we can represent the Q values of a policy via a table where the rows represent states, the columns represent actions, and the intersection of a row and a column gives us the Q-value for a specific state-action pair. This is what we call a Q-table.

$$Q(s, a) = E[G_t | s_t = s, a_t = a] \qquad (2)$$

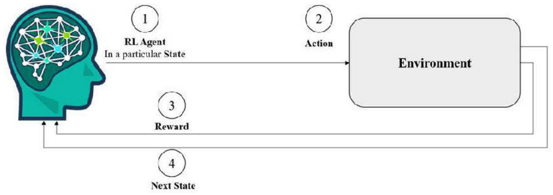


Fig. 4 Reinforcement Learning illustration

A. The Bellman Equation

The Bellman equation (3) gives us an approximation of the Q-value. When an agent is in state s_t, and performs an action a_t, the environment responds with a reward R_{t+1}, and puts the agent in a new state s_{t+1}. Since the agent does not know the rewards of future time steps when he is interacting with the environment (R_{t+2}, R_{t+3}, etc), the Bellman equation approximates the Q-value by assuming that the agent will choose the action a' at state s_{t+1} that gives the highest return when following his policy.

$$Q(s_t, a_t) = R_{t+1} + \gamma \max Q(s_{t+1}, a') \qquad (3)$$

B. Q-Learning

Q-Learning which stands for Quality Learning is an RL algorithm that teaches an agent to take the suitable action given a state. The goal of Q-Learning is to learn a policy that maximizes the total reward. In other words, the Q-Learning algorithm learns the Q-values in a Q-table. Once the Q-table reaches a convergence, we say that we reached an optimal policy, where the agent knows the consequences of doing a specific action in a specific state. The agent starts with a Q-table initialized with zeros, and then through interaction with the environment he updates this table based on the Bellman Equation, and later (after convergence), this table becomes his reference to select the best possible action given a state.

C. Deep-Q-Learning

Deep-Q-Learning is a version of Q-Learning where we have a Neural Network (NN) that learns the Q-value function for different states and actions instead of a table. We call this NN a Deep-Q-Network (DQN). This DQN receives a state as input and computes the corresponding Q-values for each action as output. The DQN algorithm can be explained simply as a regression problem, where the NN predicts the Q-values for all possible actions given a state. The difference with the classical supervised learning approach is that in supervised learning, we have the true target values, but in RL we do not. This is why we use the Bellman Equation to approximate the target Q-values (4) and train the network in a supervised online fashion. For detailed information about how this algorithm works refer to [13].

$$Q_{target}(s_t, a_t) = R_{t+1} + \gamma \max Q_{pred}(s_{t+1}, a') \qquad (4)$$

D. Double-Q-Learning

Although DQN is one of the most powerful algorithms, its problem is that not only the predicted Q-values rely on the NN, but also the target Q-values that we estimate using the Bellman equation, and this is why, when updating the weights of the network to give us a prediction close to the target value, the target value will change also. This is why, the double DQN was introduced in [14] to break this correlation, where instead of one NN, we use 2 NNs. One of them (the learning network) gets updated at each timestep just like before, and the other one (the target network) is used to calculate the target values. This way, when updating the learning network, the target variables won't change. Every once in a while, the target network gets updated with the weights of the learning network. For detailed information about how the algorithm functions refer to [14].

In our study, we will be working with a Double DQN, and in the next sections, we will discuss how we formulated the problem to be suitable for RL, and how we trained our RL agent to distribute vaccines efficiently.

V. Problem Setup

We need to solve the vaccine distribution problem using RL. This is why we framed it as a game that an RL agent should learn to play well and get high scores while playing it. The game rules and specifications can be summarized as follows:

- We have 300 houses in the environment which gives us 1120 agents. We chose this number just to be able to train the RL agent faster. In our previous studies, we used to take around 60000 agents in the simulation.
- Out of the 1120 agents, 90% will register to take the vaccine.
- Each day we have a limited amount of 20 vaccine shots that we can deliver.
- These 20 shots can be delivered to one of the 5 possible groups:
  - G1: agents having an age between 12 and 17.
  - G2: agents having an age between 18 and 39.
  - G3: agents having an age between 40 and 60.
  - G4: agents having an age higher than 60.
  - G5: agents that are workers, regardless of their age.
- An agent that has taken the first shot will automatically take the second shot when the time comes.
- The episode starts with 1% of the population infected.
- The episode ends when the number of active infections drops below 5, or when the RL agent loses (this idea will be explained later in the section).
- During the episodes, we keep schools and universities closed.

We start by simulating in our environment a random strategy for vaccine distribution where the vaccines are distributed to a random group each day. For the simulated random strategy, we obtain 2 curves: one that shows the cumulative infections over time, and one that shows the Deaths over time. We consider this strategy as a baseline that the RL agent should be bypassing during the learning process. For this reason, each day, the RL agent will receive a reward (5) that takes into account the difference between the random strategy and the strategy that the agent is following in terms of the resulting cumulative cases and deaths.

$$\boldsymbol{R}[i] = \alpha \frac{|\boldsymbol{I_r}[i] - \boldsymbol{I_s}[i]| * |\boldsymbol{D_r}[i] - \boldsymbol{D_s}[i]|}{\boldsymbol{N_{agents}}} \qquad (5)$$
$$\alpha = \begin{cases} 1 & if\ (\boldsymbol{I_s}[i] < \boldsymbol{I_r}[i])\ \boldsymbol{and}\ (\boldsymbol{D_s}[i] < \boldsymbol{D_r}[i]) \\ -1 & Otherwise \end{cases}$$

$\boldsymbol{I_r}$: Number of infections reached in the random strategy.
$\boldsymbol{I_s}$: Number of infections reached using the learned strategy.
$\boldsymbol{D_r}$: Number of deaths reached in the random Strategy.
$\boldsymbol{D_s}$: Number of deaths reached in the learned strategy.
$\boldsymbol{N_{agents}}$**:** Number of Agents.
$[i]$: Index representing the day during an episode.

From the reward formulation, we can see that the fewer infections and deaths our strategy achieves compared to the random strategy, the higher the reward will be. Another thing to consider, the reward is going to be positive if and only if the numbers of infections and deaths achieved by the RL strategy are lower than the ones achieved by the random strategy. This is the goal of weighing the reward by α. When the RL agent receives 5 consecutive negative rewards (meaning: the strategy he is following is not better than the random one) he loses and the episode ends.

In our study, we trained a Double DQN where the network has 2 inputs, 3 hidden layers (with ReLU activation function), and one output layer (with a linear activation function) of 5 Q-values that correspond to the 5 possible actions. The architecture of the NN that we used is visualized in Fig.5.

VI. Training Process

We trained our Double DQN with the mentioned conditions for 100 episodes (equivalent to 5000 training steps) using an ε greedy policy where at a particular time step, an agent selects the action that gives the highest expected return with a probability of $1-\varepsilon$ but selects a random action with a probability of ε. This helps the RL agent explore the environment. The actions that an agent takes have long-term consequences and this is why we used a discount factor γ of 0.95 to pay attention to future rewards while approximating the Q-values. The training parameters are summarized in Table V.

TABLE V Training Parameters of the Double DQN

Learning Rate	Optimizer	Loss Function	Batch Size	Discount Factor γ	ε
0.001	Adam	MSE	64	0.95	0.1

We plot the rewards that an agent is getting throughout the 100 training episodes (Fig. 6). We can see that the rewards are noisy, and it might seem that the agent is not progressing, however, if we look at the exponential trendline of these values, we can see that it is going up, which means that the agent is learning and tends to even have higher rewards in the future.

VII. Results

We take the strategy that gave us the highest reward and we plot the corresponding curves for the cumulative cases and the deaths (Fig. 6 and Fig. 7), and we compare them with the random strategy, the "younger first" strategy, and the "older first" strategy. We can see that the learned strategy was able to outperform all of the other strategies in terms of cumulative cases and deaths at the same time.

If we look at the strategy the RL agent derived, it was about starting with workers (G5), then moving to agents

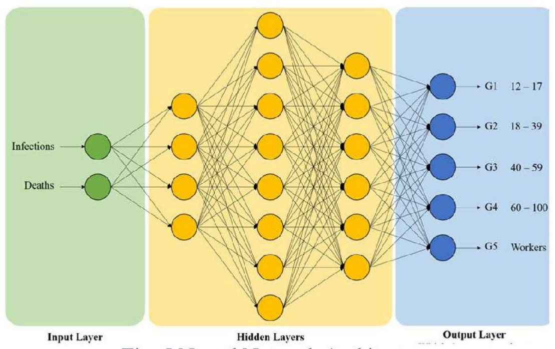


Fig. 5 Neural Network Architecture

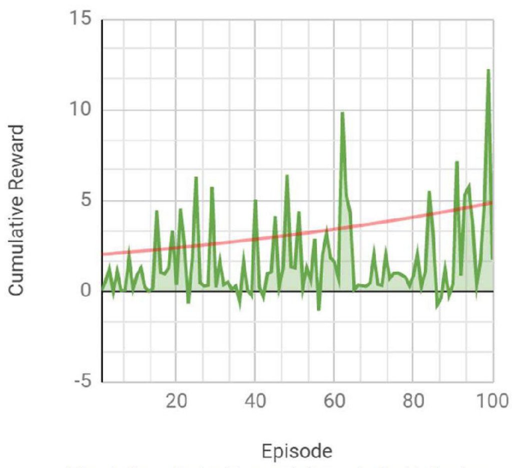


Fig. 6 Cumulative Reward through the Episodes

between 17 and 39 years old (G2), then moving to agents between 40 to 59 years old (G3), then move to agents older than 60 years old (G4), and finally vaccinating agents younger than 17 years old (G1).

Maybe if we train for more episodes, we would be able to come up with a strategy that achieves even better outcomes than this one did, but the results we obtained help us prove the concept that the way we vaccinate agents has an impact on the human damages caused by the pandemic and such decision-making problems can be better solved using RL.

VIII. Conclusion and Future Works

The aim of this study was to prove the concept that Artificial Intelligence techniques and more specifically Reinforcement Learning can be very useful for complex decision-making problems. Applying RL directly in real life especially in applications related to the medical field can be very challenging, and this is why, we need simulation systems where we test our ideas to make sure they are feasible and effective, and this study is the first step in that direction. The main limitation that we had in this study is the lack of computing resources that can maintain training for a large number of episodes with a big number of agents. If we bypass this limitation, we can increase the number of agents, and increase the training time, and therefore achieve better strategies. We can also perform further hyperparameter tuning to improve the learning process. Another thing we can do is to think about how we can apply such a system in real life by getting real data instead of operating on statistical distributions.

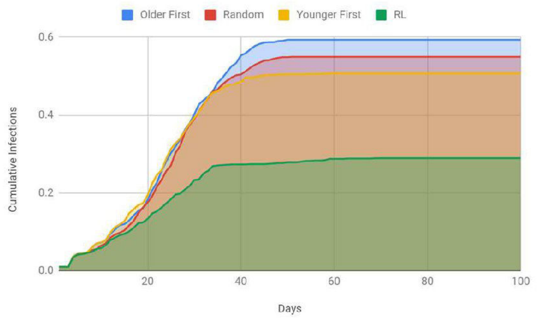


Fig. 7 Normalized Cumulative Infections for different Strategies

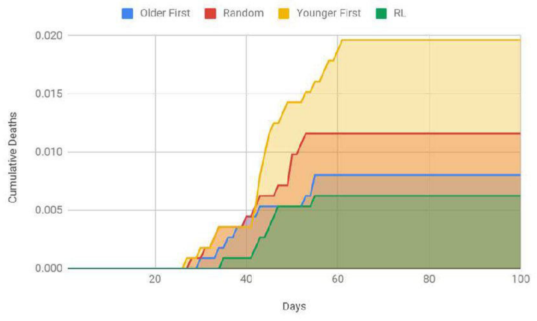


Fig. 8 Normalized Deaths for different Strategies

IX. References

[1] R. Awasthi *et al.*, “VacSIM: Learning Effective Strategies for COVID-19 Vaccine Distribution using Reinforcement Learning,” *ArXiv200906602 Cs*, Jan. 2021, Accessed: May 05, 2021. [Online]. Available: http://arxiv.org/abs/2009.06602

[2] L. Thul and W. Powell, “Stochastic Optimization for Vaccine and Testing Kit Allocation for the COVID-19 Pandemic,” *ArXiv210101204 Cs*, Jan. 2021, Accessed: Jan. 08, 2022. [Online]. Available: http://arxiv.org/abs/2101.01204

[3] S. El Falou and F. Trad, “Forecast Analysis of the COVID-19 Incidence in Lebanon: Prediction of Future Epidemiological Trends to Plan More Effective Control Programs,” in *2021 Sixth International Conference on Advances in Biomedical Engineering (ICABME)*, Oct. 2021, pp. 135–140. doi: 10.1109/ICABME53305.2021.9604861.

[4] Y. M. Bar-On, A. Flamholz, R. Phillips, and R. Milo, “SARS-CoV-2 (COVID-19) by the numbers,” *eLife*, vol. 9, p. e57309, Apr. 2020, doi: 10.7554/eLife.57309.

[5] S. A. Lauer *et al.*, “The Incubation Period of Coronavirus Disease 2019 (COVID-19) From Publicly Reported Confirmed Cases: Estimation and Application,” *Ann. Intern. Med.*, vol. 172, no. 9, pp. 577–582, Mar. 2020, doi: 10.7326/M20-0504.

[6] CDC, “Interim Guidance on Duration of Isolation and Precautions for Adults with COVID-19,” *Centers for Disease Control and Prevention*, Feb. 11, 2020. https://www.cdc.gov/coronavirus/2019-ncov/hcp/duration-isolation.html (accessed Apr. 12, 2021).

[7] Erik Peper, “Coronavirus risk in context: How worried should you be?,” *the peper perspective*, Mar. 08, 2020. https://peperperspective.com/2020/03/07/corona-virus-risk-in-context-how-worried-should-you-be/ (accessed Apr. 10, 2021).

[8] X. Yan *et al.*, “Clinical Characteristics and Prognosis of 218 Patients With COVID-19: A Retrospective Study Based on Clinical Classification,” *Front. Med.*, vol. 7, 2020, doi: 10.3389/fmed.2020.00485.

[9] E. Mahase, “Covid-19: Assess the effects of extending Pfizer vaccine dosing interval, expert urges,” *BMJ*, vol. 372, p. n162, Jan. 2021, doi: 10.1136/bmj.n162.

[10] E. Mahase, “Covid-19: Pfizer vaccine efficacy was 52% after first dose and 95% after second dose, paper shows,” *BMJ*, vol. 371, p. m4826, Dec. 2020, doi: 10.1136/bmj.m4826.

[11] GOV.UK, “COVID-19 vaccine surveillance reports,” *GOV.UK*, May 2021, Accessed: Jun. 14, 2021. [Online]. Available: https://www.gov.uk/government/publications/covid-19-vaccine-surveillance-report

[12] P. Azimi, Z. Keshavarz, J. G. C. Laurent, B. Stephens, and J. G. Allen, “Mechanistic transmission modeling of COVID-19 on the Diamond Princess cruise ship demonstrates the importance of aerosol transmission,” *Proc. Natl. Acad. Sci.*, vol. 118, no. 8, Feb. 2021, doi: 10.1073/pnas.2015482118.

[13] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning,” *ArXiv13125602 Cs*, Dec. 2013, Accessed: Nov. 13, 2021. [Online]. Available: http://arxiv.org/abs/1312.5602

[14] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-learning,” *ArXiv150906461 Cs*, Dec. 2015, Accessed: Nov. 13, 2021. [Online]. Available: http://arxiv.org/abs/1509.06461