ECE 238 Project: Stage-Wise School Reopening using Reinforcement Learning

Mohan Srinidhi Acharya
UID: 905627884
mohsriach@g.ucla.edu

Terri Tsai
UID: 805624292
tctsai@g.ucla.edu

Abstract

COVID-19 has caused immense suffering and damage throughout the world, overwhelming healthcare systems and hurting economies.The education sector has also taken a hit, with uncertainties regarding re-opening of schools for in-person classes. In this work, we aim to solve the problem of school re-opening in a stage-wise approach using Machine Learning. We introduce Reinforcement Learning strategies to optimize lockdowns and subsequent re-opening of the education sector. We do this to ensure that while the healthcare system remains capable of handling COVID cases, students' educational experiences are intact and does not suffer significantly. Using our defined metrics, we analyze RL-learned policies under modified models with varying action space sizes and agents' action frequencies, and conclude that our RL model is able to schedule stage-wise re-openings that bring more utility than the baseline method.

1 Introduction

Since December of 2019, COVID-19 has created dangers and disruption to the world that we have not seen before. In early 2020, we saw the number of deaths rise unexpectedly high across the country, and we experienced government lockdown regulations at varying degrees in attempts to slow down the spread of the virus. While it is clear that more regulations lead to lesser spread of the virus, enforcing restrictions comes with costs like the decline of the economy from business closures, decline of mental health from isolation, decline of employee and student productivity from working from home, and much more. With the unpredictability of COVID-19, we have seen how difficult and controversial of a process it is to define a set of regulations that can optimally balance the community's health with other aspects of the community's well-being.

With this complex problem disrupting our everyday lives, many have looked into ways to use Machine Learning to look for solutions to mitigate the damage COVID-19 causes. Examples include Youyang Gu's *Covid19 Projections* model that uses Supervised Learning and information about past deaths to learn the parameters of his SEIR model, then using those learned parameters to make future death projections [1]. Other works have been explored in the Reinforcement Learning (RL) paradigm, such as [2] that trains an RL agent to take suitable cyclic actions of locking down and reopening a community in order to prevent catastrophic increases in the number of ICU beds in use, to avoid placing a huge burden on the healthcare system. For our work, we wish to learn optimal strategies to aid the reopening of schools, because while remote learning is convenient and delivers content that is similar to what is discussed in classrooms, it certainly does not fulfil the same expectations as in-person learning.

In this work, we consider the Reinforcement Learning problem of formulating effective lockdowns and subsequent re-openings within a school setting. A RL setting is fitting because in a RL problem, we can model actions that have lasting effects contributing to future scenarios. Since COVID-19 is a novel disease that has a delayed transmission time and a marked lag in terms of onset of symptoms, it is crucial to make informed decisions that not only affects the present but also the future. We introduce

an agent that takes actions of fully opening, partially opening, and fully closing a school campus based on observations such as the number of infected people and the number of new cases each day. To encourage the agent to take steps in the right direction, rewards are assigned to smart actions, and the agent works to maximise the rewards. We perform analysis on how an agent's learned policy is affected with the change in action frequency, i.e. how often the school can change the regulation level, and with the change in action space size, i.e. how many available levels of regulations the agent has to choose from. At the end, we also introduce vaccination as a parameter to show that the agent can still learn a high-performing policy when anticipating an upcoming vaccination date. An elaborate analysis of the policies chosen by the agent will be discussed to evaluate its effectiveness and practicality.

2 Related Works

A prominent inclusion to solving COVID-19 problems is adapting compartmental models like SEIR models that mathematically summarize the progression of infection diseases within a population.

In *COVID-19 Pandemic Cyclic Lockdown Optimization Using Reinforcement Learning*, Mauricio Arango and Lyudmil Pelov used reinforcement learning to optimize cyclic lockdowns to avoid ICU bed shortages and minimize economic impact [2]. The paper uses an extended SEIR model that includes other detailed compartments such as "Hospitalized", which is used to determine the number of ICU beds in use. The actions of the RL agent is to either stay open or go into lockdown, and this action is fed into the SEIR model in the form of toggling the input parameter R (the effective reproduction number), between a low and high number. The reward after each day is determined by if there is a lockdown and the whether the ICU beds surpassed the threshold. The epidemic state after each day is fed back to the RL agent as the next state, where the RL agent observes the number Infected individuals. This work showed that the inclusion of RL methods better avoided ICU usage overflows compared to baseline none-RL methods.

With the increase of vaccine development, some works have been incorporating vaccinations into their solutions. In *Control strategies for COVID-19 epidemic with vaccination, shield immunity and quarantine: A metric temporal logic approach*, Zhe Xu *et al.* proposed synthesized control methods to control the epidemic, and showed that better vaccination control efforts can achieve lesser deaths [3]. Our reinforcement learning model will also be modeled with vaccinations included.

[2] strictly uses an action space of size 2, and explores the effects of using different values of R to represent the reopening action by individually testing their model under different R-values. In our work, instead of a strictly open and reopen policy, we want to explore a stage-wise reopening, where the actions of the agent is allowed to take actions representing lockdown, fully reopening, and intermediate partial reopening. Additionally, the agent in [2] strictly operates on a daily time step; our work will also model a daily time step but will as well explore the effects of longer action lengths. Lastly, we will be including the vaccination scenario in our SEIR model like [3] to examine how our agent responds under the vaccination cases.

3 Problem Formation: A Reinforcement Learning Problem

Our problem is set up to be Markov and stationary throughout time. This means that the information observed by the agent in the current state entirely summarizes the past, and that the transition model stays the same overtime.

3.1 States

Each state is represented by a 12-dimensional vector, in the continuous space. The first five dimensions contain values of the number of people in the population $N = 50,000$ who are: *susceptible, exposed, infected, recovered, dead*. The last seven dimensions contain the number of new cases that arose in the last seven days, i.e. the number of cases on day $t-7$, on day $t-6$, up till day $t-1$. In the environment, all twelve values are represented in their normalized form by dividing by N.

3.2 Actions

The action space is discrete and 3-large, where actions $0, 1$, and 2 represent the act of a full shut-down of in person schooling, a partial shut-down, and a full-reopening. These actions when fed into the SEIR model for simulation, will be represented by the effective reproduction number R with values of $0.7, 1.2$, and 1.5. Here we make the assumption that a full lockdown can bring the effective reproduction down to $R = 0.7$, a full reopening would bring it up to $R = 1.5$, and a partial-reopening would have a value somewhere in between that we decided to be $R = 1.2$ [2]. At each time step, a continuous 12-dimensional state as described previously, and a discrete action as described here, are passed to the SEIR model to undergo state transition.

3.3 State Transitions

The first five dimensions of the state at day t transitions to the state that the agent sees at day $t + 1$ through the set of differential equations that defines the SEIR model. The new numbers of the number of people in each compartment after one day are calculated as follows:

$$\begin{aligned} S_{t+1} &\leftarrow S_t - \frac{\beta I_t}{N} S_t - V \\ E_{t+1} &\leftarrow E_t - \epsilon E_t + \frac{\beta I_t}{N} S_t \\ I_{t+1} &\leftarrow I_t + \epsilon E_t - (\gamma + \alpha) I_t \\ R_{t+1} &\leftarrow R_t + \gamma I_t + V \\ D_{t+1} &\leftarrow R_t + \alpha I_t \end{aligned} \tag{1}$$

where $\epsilon = \frac{1}{\text{Incubation Period}} = 0.2$ is the rate of progression from Exposed to Infected, $\gamma = \frac{1}{\text{Infectious Period}} = 0.2$ is the recovery rate, $\alpha = 0.006$ is the death rate, and $\beta = \gamma R$ is the transmission rate [3]. V is the vaccination rate per day, and is set to zero when vaccinations are not considered. This set of equations are made dynamic and a part of the reinforcement learning problem because the parameter $\beta = \gamma R$ varies at each t depending on the agent's actions at time t, as describe in the last section.

The last seven dimensions of the state are the number of new cases that arose on the past seven days. We make the assumption that the number of new cases that arises on day t is the number of people who go from being exposed to being infected that day, i.e. ϵE_t. Let us define the seven dimensions as $(n_t^{[7]}, n_t^{[6]}, n_t^{[5]}, n_t^{[4]}, n_t^{[3]}, n_t^{[2]}, n_t^{[1]})$, where $n_t^{[1]}$ is the number of new cases yesterday, and $n_t^{[7]}$ is the number of cases seven days ago, then transition of the last seven dimensions is like so:

$$(n_{t+1}^{[7]}, n_{t+1}^{[6]}, n_{t+1}^{[5]}, n_{t+1}^{[4]}, n_{t+1}^{3]}, n_{t+1}^{[2]}, n_{t+1}^{[1]}) \leftarrow (n_t^{[6]}, n_t^{[5]}, n_t^{[4]}, n_t^{[3]}, n_t^{[2]}, n_t^{[1]}, \epsilon E_t) \tag{2}$$

Basically this acts as a running tracker of the number new cases in the past seven days.

Because the states transition via the SEIR model that has no stochastic component to it, the state transitions are deterministic.

3.4 Rewards

In order to optimize the agent to balance keeping the school open as much as possible, while ensuring the health safety of the students, our reward function is two-fold to account for both goals. We assign positive reward points in $reward_1$ to encourage opening up, and assign negative reward points in $reward_2$ to penalize the observation of number of new cases being too high. Let m_t denote the number of new cases in the last seven days, which can be extracted from the state space, i.e. $m_t = n_t^{[7]} + n_t^{[6]} + n_t^{[5]} + n_t^{4]} + n_t^{[3]} + n_t^{[2]} + n_t^{[1]}$. The threshold of what is too high of a m_t is chosen to be $m_{threshold} = \frac{N}{100{,}000} \times 100$. This threshold was chosen because the CDC website states that if

the number of new cases in the last seven days exceeds 100 for every 100,000 people in a community, then it is said that the an K-12 school is at high transmission risk [4].

$$\text{reward}_1(t) = \begin{cases} 0, & \text{if } action = 0 \\ \alpha_1, & \text{if } action = 1 \\ \alpha_2, & \text{if } action = 2 \end{cases} \tag{3}$$

$$\text{reward}_2(t) = \begin{cases} 0, & \text{if } m_t < m_{threshold} \\ \frac{-\alpha_1}{m_{threshold}} m_t, & \text{if } m_t \geq m_{threshold} \end{cases} \tag{4}$$

$$\text{reward}_{total}(t) = \text{reward}_1(t) + \alpha_{linear}\text{reward}_2(t) \tag{5}$$

Note that the penalty assigned in $reward_2$ is scaled to be around the same magnitude as $reward_1$. For example, if $m_t = m_{threshold}$, then the penalty will equal $-\alpha_1$, essentially canceling out the benefits of choosing $action = 1$ of half-opening the school. By defining $reward_1$ and $reward_2$ this way of the same magnitude, we can view setting $\alpha_{linear} = 1$ in $reward_{total}$ as representing weighing the importance of both goals equally on that day t. In all our models, we set $\alpha_1 = 1$ and $\alpha_2 = 2$, and tune α_{linear} as a hyperparameter when tuning for optimal score using metrics described in section 5.2.

4 Proposed Solution/Implementation

4.1 Algorithm

We use Deep Q-Learning [5] as the algorithm to solve the problem. Since it is an off-policy algorithm, it learns the optimal policy independently of the agent's actions. Deep Q-Learning is an upgrade from Q-Learning in that it uses a function approximator like a Neural Network instead of the Q-table that would grow exponentially in size and violate memory constraints.

DQN incorporates two important features : a) Experience Replay and b) Target Networks. There are issues associated with using a function approximator instead of Q-Table in Q-Learning. The sequential dependency of a state can potentially lead to difficulties in training a DQN. This happens because reward obtained at time t is dependent on the state and action taken at time $t-1$, which in turn is dependent on time $t-2$ and so on. Using Experience Replay, a memory bank of different states, actions and rewards is created, out of which we randomly sample values. The sampling breaks any kind of sequential dependency in the data during learning.

Estimation of the next state is another issue in DQN. In tabular Q-Learning, the next state estimate is given by :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\big(R_t + \gamma max_a Q(s_{t+1}) - Q(s_t, a_t)\big) \tag{6}$$

The tabular method updates only $Q(s_t, a_t)$ whereas DQN updates entire network in each step. When this happens, the estimate of the best action in the next state also changes. This makes it difficult for the network to learn because the target is always moving and hence the back-propagation error is not consistent from one update to the next. Therefore, a target network is introduced, which is a copy of the neural network that is being trained, but it is only copied every **N** time steps. This ensures stability in the error allowing the network to learn better and provides the target network a better approximation to learn again after the update.

At each update, the function approximator neural network is updated with respect to the gradient of the loss function. The loss function is defined as follows :

$$\mathcal{L}(\theta) = \left(R_t + \gamma max_a\big(Q(s_{t+1}; \theta_t)\big) - Q(s_t, a_t; \theta)\right)^2 \tag{7}$$

where θ and θ_t are parameters of our prediction network and target network respectively. The loss function is squared in order to penalize the larger errors more than the smaller errors. Figure 1 shows the DQN architecture consisting of a target network, an experience replay memory with the defined loss function.

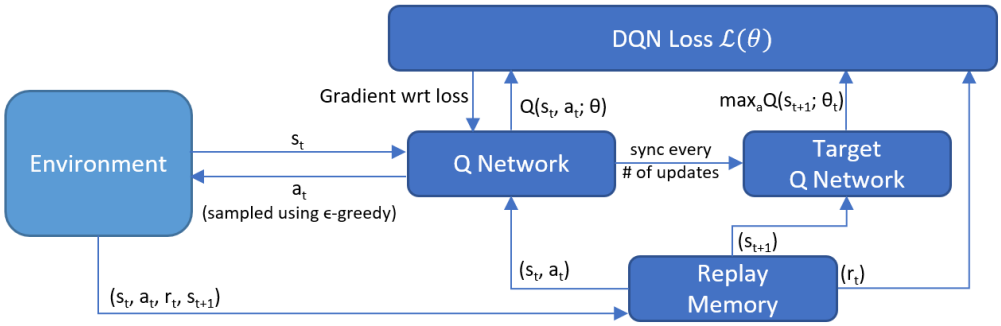


Figure 1: DQN Architecture

5 Results

5.1 Pre-RL Setup and Initial Conditions

Before the reinforcement learning takes place, the first 30 days of the simulation is modeled with the SEIR model to behave similarly to March 2020 in California, where COVID started becoming prevalent. A high effective reproduction number of $R = 3$ is used during this 30-day period to model a surge of cases. After 30 days, a lockdown is simulated with $R = 0.7$ to reflect similarity to the enforced government lockdown regulations around April 2020. Only after the surge and lockdown do we start to consider feeding our initial state into the reinforcement learning problem. A brief investigation was conducted to examine how the the initial state can affect the agent's performance where we tested with starting the RL training 30 days and 60 days after the lockdown. The agent was able to learn reasonable policies from both initial states, showing that our model is robust to initializing in a state above the high-risk threshold and under. All our experiment results are initiated at 30 days after lock down (day 60).

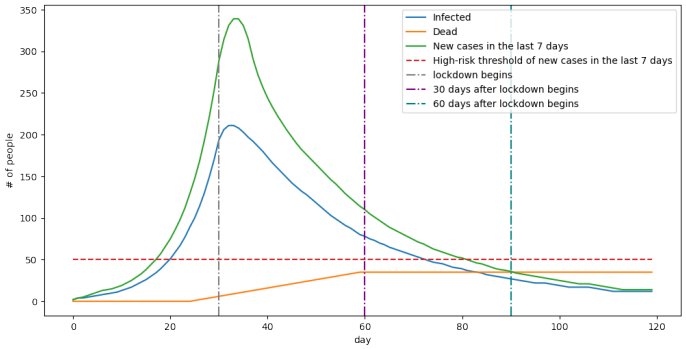


Figure 2: Figure shows that new cases in the last 7 days is above threshold 30 days post-lockdown (day 60), and below threshold 60 days post-lockdown (day 90). These two states were used to investigate robustness against initial state, and both of them resulted in successful training.

5.2 Performance Comparison

To compare learned policies, the following metric was employed to measure performance and used in hyper-parameter tuning:

$$\text{score}_1(t) = \begin{cases} 0, & \text{if } action = 0 \\ 1, & \text{if } action = 1 \\ 2, & \text{if } action = 2 \end{cases} \tag{8}$$

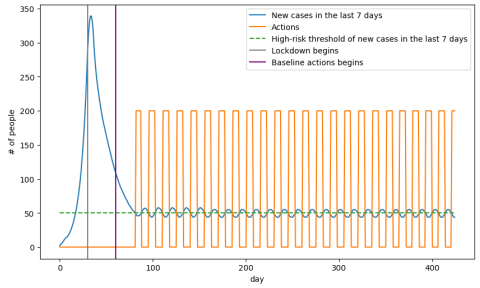
$$\text{score}_2(t) = \begin{cases} 0, & \text{if } m_t < m_{threshold} \\ \frac{-m_t}{m_{threshold}}, & \text{if } m_t \geq m_{threshold} \end{cases} \tag{9}$$

$$\text{score} = round\left(\sum_{t=1}^{365} \text{score}_1(t) + \text{score}_2(t)\right) \tag{10}$$

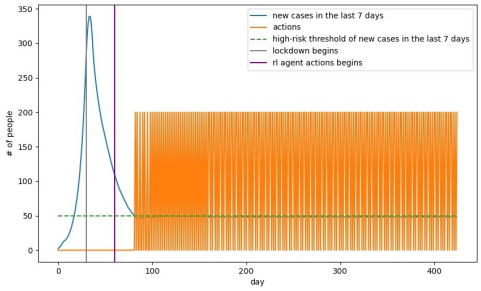
At the high level, the score function assigns points each day for open-ness, and subtracts points each day if the number of cases the last seven days exceeds the threshold. The score function is designed so that points on a given day is canceled out to be zero if the school is half-opened, but the number of cases the last seven days is at threshold, because we want to model that there is no net utility if the school is allowed to be somewhat opened at the cost of cases exceeding threshold. Note that the metric is very similar to the reward function, but the score function explicitly weighs the two objectives equally, whereas the weight factor in the reward function is tuned as a hyperparameter during training investigations. For comparison, a baseline action was created as the following: the school full opens the moment cases are observed to be under threshold, and fully shuts down he moment cases exceed the threshold, on a day-by-day basis. This baseline model scores 104 in our metric.

5.2.1 Action Frequency

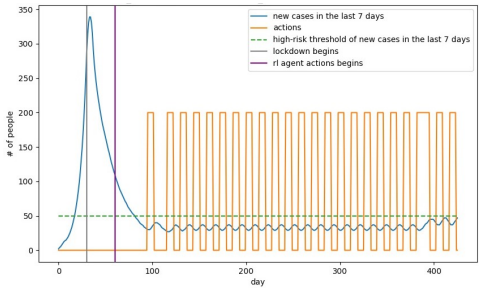
We first model the case where the reinforcement learning agent is, like the base case, making its action decisions on the daily basis. However, a practical consideration is the feasibility of such policy. Although high in metric score, locking down and reopening a school on a high frequency basis is convenience-wise sub-optimal. Additional cases were modeled where the agent is required to stick to their chosen action for 7 days, and for 14 days. These decision-making frequencies were chosen to be examined because course schedules could possibly be designed around a weekly or biweekly pattern. From figure 3 we observe policies created under reinforcement learning consistently outperforms the baseline model. We also observe that the models' performances match our intuitive understanding, where agents trained under models that allow for more frequent action switches score higher in metric. When it comes to optimizing stage-wise lockdown and reopening schedule, there is as trade-off between convenience and score.



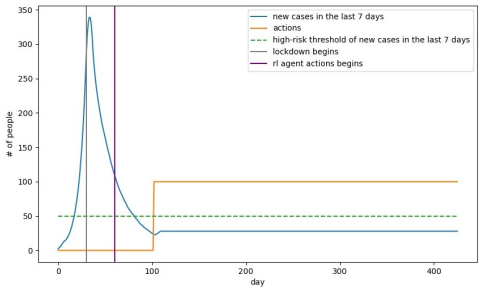
(a) Baseline model with daily actions; Score:104



(b) Daily-action model; Score:351



(c) Weekly-action model; Score:303



(d) Bi-weekly-action model; Score:291

Figure 3: Performance results of the Baseline, Daily-action RL, Weekly-action RL, and Bi-weekly-action RL model.

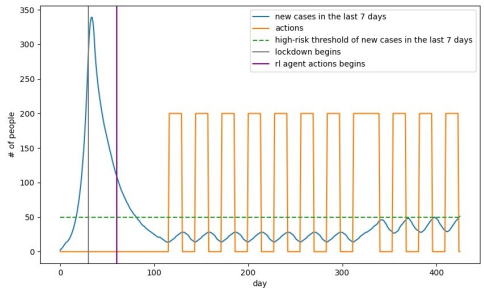
* Agent's actions are represented in the graph by scaling actions $0, 1, 2$ to $0, 100, 200$ for convenient viewing.

5.2.2 Action Space

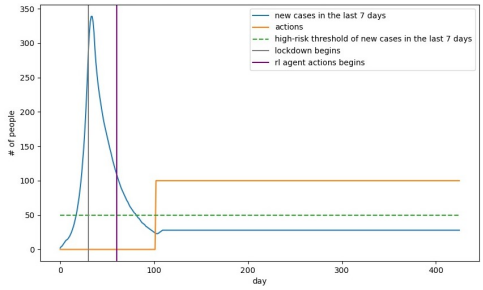
A common occurrence observed in the previous section is that the agents tend to pick the same level of reopening throughout, i.e. the agents are not acting on all three available actions throughout the episode. This observation motivated further examination into differently sized action spaces, to see whether an agent would pick a more varied set of actions when given larger action spaces.

In this modified setting, the effective reproduction numbers R corresponding to the new actions are linearly distributed between the minimum and maximum R values in the 3-action model, the function $reward_1$ assigns values linearly distributed between the minimum 0 and maximum α_2, and the function $score_1$ assigns values linearly distributed between 0 and 2. For example, if our model now has a 5-large action space, actions $0, 1, 2, 3, 4$ will represent R values of $0.7, 0.9, 1.1, 1.3$, and 1.5, with $reward_1$ assignments of 0, $\frac{\alpha_2}{4}$, $\frac{2\alpha_2}{4}$, $\frac{3\alpha_2}{4}$, and α_2, and $score_1$ values of 0, $\frac{1}{2}$, 1, $\frac{3}{2}$, and 2.

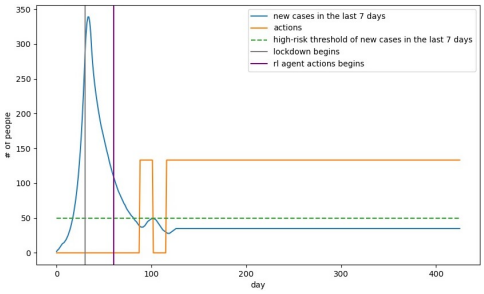
Experiments on action spaces of size 2, 4, and 5 are performed in the bi-weekly action frequency setting to be compared to the standard 3. From figure 4, we notice that changing the number of actions that the agent could possibly take by inserting more intermediate levels of open-ness in between fully closed and fully open does not necessarily influence the agent to pick all the varied actions. Rather, the agents tends to rotate between the two actions of fully closed, and one particular level of open-ness, i.e. fully open action in the 2-dimension action space case, partially open with $R = 1.2$ in the 3-dimensional action space case, partially open with $R = 1.23$ in the 4-dimension action space case, and partially open with $R = 1.3$ in the 5-dimensional action space case. Although increasing the size of the action space does not greatly effect the diversity of chosen actions, results reflect that scores do generally trend upwards with larger action spaces.



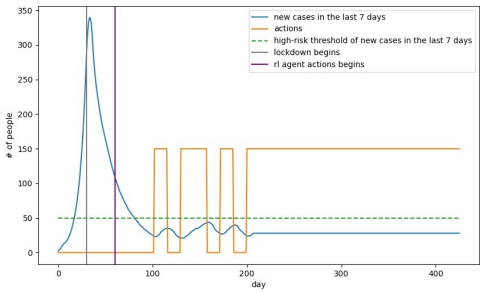
(a) Bi-weekly action, 2-dimensional action space; Score:300



(b) Bi-weekly action, 3-dimensional action space; Score:300



(c) Bi-weekly action, 4-dimensional action space; Score:400



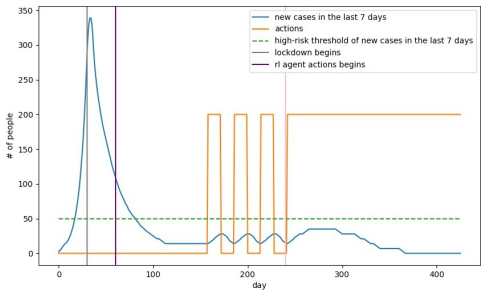
(d) Bi-weekly action, 5-dimensional action space; Score:390

Figure 4: Performance results of the differently-sized action spaces in the bi-weekly action setting.
* Agent's actions are represented in the graph by scaling the discrete actions to between 0 and 200 in increments of $200/(\text{action space size} - 1)$

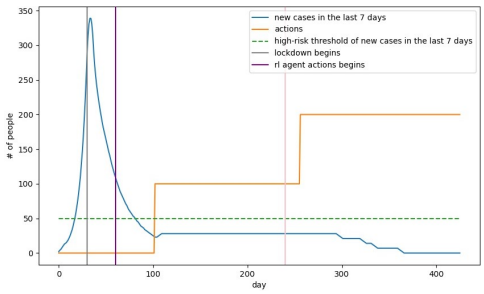
5.2.3 Vaccinations

To include foreseeing the development of vaccinations in the near future, the bi-weekly setting was also modified to include the effects of vaccinations. This change is reflected in the state transition SEIR model equations. We make the assumption that the vaccination rate is 310 vaccinations each day, because between April 1st and May 1st 2021 in California, 18.6% of the healthier population

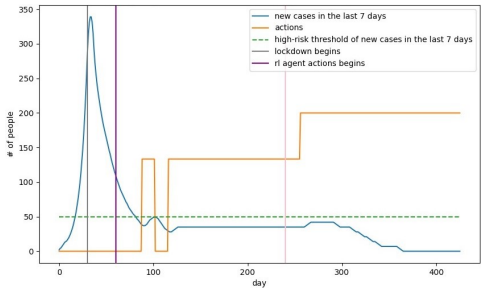
became newly vaccinated [6], so we estimated that a population of $N = 50,000$ would have $50,000 \times \frac{0.186}{30} = 310$ members fully vaccinated each day. Simulated below are the optimal policies for bi-weekly models with different action space sizes, assuming the distribution of vaccines begin 180 days after the agent starts learning. We notice that the agent chooses to fully open shortly after the vaccinations begin, even before it observes the number of new cases declining. We also see that models with more intermediate action choices result in higher reward just like the non-vaccination cases in section 5.2.2.



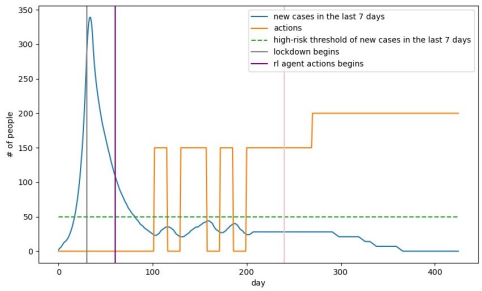
(a) Bi-weekly action, 2-dimensional action space, with vaccination; Score:193



(b) Bi-weekly action, 3-dimensional action space, with vaccination;; Score:461



(c) Bi-weekly action, 4-dimensional action space, with vaccination; Score:785



(d) Bi-weekly action, 5-dimensional action space, with vaccination; Score:969

Figure 5: Performance results of the differently-sized action spaces in the bi-weekly action setting, considering vaccine distribution starting on the agent's 180th day (simulation's 240th day).

* Agent's actions are represented in the graph by scaling the discrete actions to between 0 and 200 in increments of $200/(\text{action space size} - 1)$

6 Implementation Details

We use OpenAI's Gym [7] to build our custom environment. The DQN [8] algorithm is written in PyTorch [9], a Python-based Deep Learning framework built by Facebook that facilitates an automatic differentiation [10](AutoGrad) system. The DQN consists of a series of feed-forward networks (4 layers here) with ReLU activation function. Adam [11] is used as the optimizer of choice. The learning rate is set to 0.001 after varying it between 0.1 and 0.0001 during hyperparemter search. The batch size is set to 32. The network update frequency is set to 4 as per the paper [5]. The network sync frequency is set to 100. In the experience replay buffer, the memory size is set to 50000 while the burn in memory, which initially populates the replay buffer is set to 10000. The loss function used is Mean Squared Error (MSE). The models were trained for 1000 episodes, each episode with a randomized length between 250 and 500 days, on CPU with a training time of around 4 minutes. All final simulations of the trained agent policies were simulated across 365 days.

7 Conclusion

Through investigating models with different action frequencies and action space sizes, we observe that an agent with more choices, either in the form of being able to switch actions more often, or in the form of having more intermediate action options, tends to learn more high-scoring policies. This

found result aligns with the intuitive understanding that better results can be discovered when more options are available to be explored. Qualitatively, we also observe that regardless of the number of actions available for the agent, agents tend to pick only two actions of fully-closing and a form of opening. Thus, when applying this reinforcement learning solution in a real-life scenario, it may be a good idea to include a larger action space for the agent to choose from, because even though the agent will ultimately only choose one level of open-ness in the optimal policy, that level of open-ness will be the best one that produces the highest score, and the regulation matching the chosen level of effective reproduction number can be employed. We can include five levels of regulations for the agent during training, without fearing the inconvenience of actually needing to implement all five levels of regulations in real life practice, because the optimal policies learned tend to only include two actions regardless of the action space size.

It is worth noting that while we explored the effects of different action frequencies, different modified models had to be designed and the scores were manually compared like we did in section 5.2.1. We specifically decided to investigate policies under a weekly and bi-weekly frequency. For future work, redesigning the reward function to include a school's tolerance to frequent regulation changes could be a more automated way of learning an optimal policy. This could take the form of tracking the last action taken and how long it has been since the last action change, and assigning reward for sticking to the same action. We are also interested in exploring an entirely different set of algorithms such as Actor-Critic [12] as used in [13] to understand how the policy would change with a different training mechanism.

References

[1] Youyang Gu. *COVID-19 projections using machine learning.* `https://covid19-projections.com` [Accessed: 05.03.2021].

[2] Mauricio Arango and Lyudmil Pelov. COVID-19 pandemic cyclic lockdown optimization using reinforcement learning. *CoRR*, abs/2009.04647, 2020.

[3] Zhe Xu, Bo Wu, and Ufuk Topcu. Control strategies for covid-19 epidemic with vaccination, shield immunity and quarantine: A metric temporal logic approach. *PLOS ONE*, 16, 03 2021.

[4] CDC.gov. Operational strategy for k-12 schools through phased prevention.

[5] 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.

[6] CA.gov. Vaccination progress data.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. 2016.

[8] DQN. An easy introduction to deep q-learning.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. pages 1861–1870, 2018.

[13] 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. 2020.