Reversed Plague Inc: Simulation of Spreading And Containment of a Virus by Means of A.I. Techniques

Cognolato Samuel 1237767

Fusaro Daniel 1233437

Trombin Edoardo 1242262

Abstract

The Covid-19 pandemic led to a boom in the epidemiologic modelling field, hoping to predict the trend of the virus in each nation worldwide, in order to help governments make decisions. In this paper we propose a framework[1] that simulates the spreading of a virus in a city. Such a city contains five types of buildings: houses, groceries stores, schools, workplaces and leisure places. Walkers (the people of the city) spend each day roaming around the city with a schedule, possibly infecting other walkers. An intelligent agent, implemented through a DQN, has the role of the governor of the city: it takes choices such as making mandatory social distancing, the use of masks, the closure of places, etc. It is trained in order to maximize the reward obtained from the simulation, involving the health status, the wealth and the discontent of the population.

1. Introduction

Since Covid-19 pandemic started, the entire world has acknowledged the fact that these specific situations require swift and continuous application of special measures to contain the spread of the virus and restore people's everyday life. The quickness, and the number of people that apply the measures is fundamental to reduce the multiple losses induced by the presence of the virus, and the world itself has shown how different approaches can bring **very different** outcomes. The idealization of a virtual authority that tries to rationally apply the above considerations sparked the idea of Reversed Plague Inc. The project consists of an agent (that we can idealize as "the government") that applies actions w.r.t. a population, in order to eradicate the virus and its effects on the aforementioned community. The representation of the world implemented by the project is obviously relaxed in many ways, considering the uncertainty related to human (and pathogenic) behaviour, but has the exact same goal to reach. Also, a lot of assumptions are made in order to keep the framework simple, but not so as to prevent a possible extension.

2. Related work

Q-learning [1] is a dynamic programming based algorithm for delayed reinforcement learning. Assume that the world is in a certain state, and the learning system then chooses an action. The immediate result is that a reward is received by the learner from the world, which then undergoes a transition to the next state. The objective of the learner is to choose actions maximizing discounted cumulative rewards over time. In Q-learning, policies and the value function are represented by a two-dimensional lookup table indexed by state-action pairs. The Q-learning

[1] Github project at: https://github.com/Bender97/RPlagueInc

algorithm works by maintaining an estimate of the Q* quality function, and adjusting Q* values (often just called Q-values) based on actions taken and rewards received. To find the optimal Q function eventually, however, the agent must try out each action in every state many times.

DQN (Deep Q-Network) [2] is able to combine reinforcement learning with a class of artificial neural networks known as deep neural networks, in which several layers of nodes are used to build up progressively more abstract representations of the data. A deep neural network is used to approximate the optimal action-value function Q, which is the maximum sum of rewards. A possible implementation is through a 1D CNN which enables convolving states in time (different time steps).

3. Environment

3.1 Definitions

Our agent ("the government") resides on a **discrete-time** (minimum discrete time step of 1 minute), **partially observable** state space, which represents the activity of a city. The population (we will call them "Walkers") act independently from the agent, except for the **choices** applied by the government, that they have to follow in order to stop the virus. The agent's percepts are saved as a single set of **observation parameters** (instead of a complete sequence). As we did it, the agent and the environment are two separate entities and the framework provides the tools to interact with it, independently from the agent implementation (which role may be performed also by a human operator). A full simulation run is called "episode" and it terminates when there are no more infected walkers (specifically no more asymptomatic, symptomatic or in incubation). In the episode beginning, the city and the population is created. The simulation runs every episode's step as a day, divided into 24 hours, divided into 60 minutes where each walker performs a random walk in the location where they reside in that moment. Every day, the agent makes a choice which will impact on the system and return a new state and a reward for the agent. Such a reward will be used to train the agent to know which actions lead to a better state. The simulation may be run for a multitude of episodes in order to train an agent with a machine learning implementation (as we did).

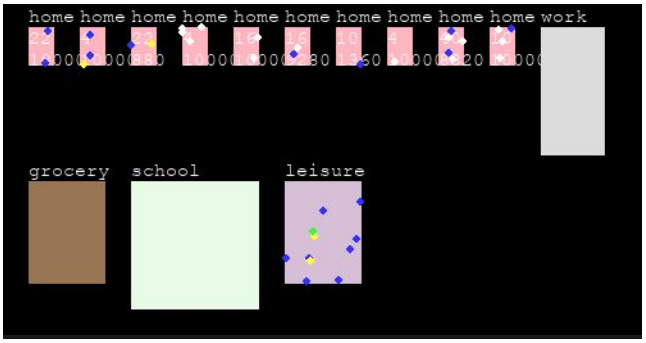

Figure 1: Example of simulation with 10 houses, one workplace, one grocery shop, one school and one leisure. Walkers are points of color representing their health status.

3.2 Walkers

Walkers (we chose this name because they keep on walking during the entire simulation) are the main components of the simulation, and represent the people inside the city; they move through different locations (individually chosen, but strongly dependent on the walker's age) during the 24 hours, following a daily-generated schedule. The schedule is represented by a sequence of locations that the walker has to visit during the day, staying for a chosen amount of hours inside each of them (locations will be discussed in the next section) and assuming that the movement between them doesn't require time. When inside a location, the walker moves in a random walk fashion, which implies the

possibility of getting in contact with other walkers. Specifically, there is a range of space (virus dependent) in which an infected walker has the possibility of infecting others with a predefined probability (more details in the Virus section). Being the active component of the simulation, all of the observation parameters depend on the walkers behaviour, and all of the choices impact on the way they act inside the environment (specifically, impact on the way they move, between and inside the locations). One of the most important features of the walker is the disobedience parameter, that models the tendency to disobey government's rules. This is needed to depict a realistic abstraction of society, where disobedience is not uncommon; it is also important because disobedience produces very different outcomes in terms of virus spreading.

3.3 Locations

We designed 5 types of locations: home, school, workplace, leisure and groceries shop.

Home: where walkers live. Each home hosts a family, and here walkers eat and sleep. A variable amount of money is related with each family. In proportion with the other locations, there are 10 houses for each of the other locations (1 per type).

School: where younger walkers go in the morning. The school defines a number of days for resting, that is how many days of the week children aren't supposed to go to school. Each young walker has a specific school to go.

Workplace: where adults go to work. Each adult has a specific workplace. Each workplace defines a salary, which every walker that works there receives (every day for simplicity). Also workplaces define a number of days dedicated to rest.

Leisure: where walkers go to have fun. A walker can go to any of the available leisure locations. Leisures define a number of closing days of the week. If a walker cannot go to such a place (i.e. because there is a closure to all leisure places) it will contribute to the overall discontent.

Grocery shop: where walkers go to buy food. Since each walker needs to eat, every day a walker for each house goes to buy food to a grocery shop, depending on the schedule. Each grocery shop defines a food price that walkers have to spend using their home's amount of money. If a family does not have enough food units to eat, they will contribute to the overall discontent.

3.4 Virus

The virus we designed is a simplification of a real virus. The infection spreads just by distance, age and health factors, and immunization after the infection survival is guaranteed.

The infection course is the following:

1. A walker starts as a **susceptible**. When it finds itself nearby a contagious walker within a certain **range** (evaluated using the euclidean distance between their x and y positions), it may get infected. The virus has a *probability of infection* parameter, which defines the probability of getting infected whenever a walker is nearby a contagious one.

2. If a walker doesn't get infected, it remains susceptible. Otherwise, it undergoes an incubation period. This period is randomly defined (within *an incubation range of days*).

3. After the incubation period, the walker has two fates: getting the symptomatic **disease** (with a certain *disease probability*) or becoming **asymptomatic** (with a probability of *1 - disease probability*). In our model, this is the *severity parameter* of the virus.

4. If it gets **asymptomatic**, it remains in that state for a random period (within *an asymptomatic range of number of days*) and then becomes **recovered**.

5. If it gets **diseased**, it remains in that state for a random period (within *a diseased range of number of days*) and then it may **die** (with a *death probability*) or may not, becoming **recovered**. In our model, this is the *lethality parameter* of the virus.

3.5 Agent

We propose the agent implementation as a DQN, which makes a choice from the observation given from the world. The agent's goal is to eradicate the virus in the shortest possible time, by trying out different choices and remembering the outcome of them. The observations are normalized to their maximum value and grouped in a temporal window forming a matrix, which will be the input to the DQN. The observations are:

Susceptible walkers: are the walkers $s(n)$ perceived by the agent as healthy walkers but susceptible to the virus. This category also contains asymptomatics, and in incubation as these 2 categories' goal is to model the fact that the virus may expand unknowingly.

Infected walkers: are the symptomatic walkers $i(n)$. Their movements may be restricted by the quarantine choice.

Recovered walkers: are the walkers $r(n)$ which were once infected symptomatic.

Dead walkers: are the walkers $d(n)$ which died from the virus before recovering.

Average wealth: is the mean value of the wealth (money) of every house.

$$M(n) = E[m(house,\ n)] \qquad (1)$$

Discontent: is the unhappiness of the population, caused by several factors.

$$d_f f(n) + d_l l(n) \qquad (2.a)$$

$$\sum_{i=1}^{C} d_i c_i(n) \qquad (2.b)$$

$$\sum_{i=1}^{C} b_i \delta(c_i(n) - c_i(n-1) - 1) \qquad (2.c)$$

where $f(n)$ and $l(n)$ are the number of walkers without food and without leisure activity during the whole step n. $c_i(n)$ is an integer value between 0 and 1 which signifies whether the choice i is enabled or not. The Dirac delta in 4.c is used to extract the fact that the choice was enabled in step n. The other values are weights. All these terms are summed to obtain $D(n)$. The term 4.a depends on the walker's happiness. The term 4.b depends on the unhappiness generated every step by an enabled choice (like the mandatory mask). The term 4.c is the unhappiness generated in the first step when a choice is enabled (like the unhappiness generated by the walkers knowing the government closed leisure places).

Some other useful functions to consider for the reward are:

Healthy walkers: is the sum of the susceptible and recovered walkers at step n.

$$h(n) = s(n) + r(n) \qquad (3)$$

Infection ratio: is the gradient of the infected from step n-1 to step n.

$$\Delta i(n) = \frac{\partial i}{\partial n}(n) = i(n) - i(n-1) \qquad (4)$$

Dead ratio: is the gradient of the dead from step n-1 to step n.

$$\Delta d(n) = \frac{\partial d}{\partial n}(n) = d(n) - d(n-1) \qquad (5)$$

Some of these functions are first of all normalized based on their maximum values, and will be weighted to compute the reward function:

$$R_n(n) = \alpha h_n(n) + \beta \Delta i_n(n) + \gamma M_n(n) + \delta D_n(n) + \\ + \varepsilon d_n(n) + \zeta \Delta d_n(n)$$

where each weight $\alpha, \beta, \gamma, \delta, \varepsilon, \zeta$ is an hyperparameter to weight its corresponding component and decide if the term is positive or negative with respect to the final reward (e.g. the dead number $\Delta d_n(n)$ is penalized heavily, hence it will have a negative weight assigned).

The agent's action space is a set of containment measures that can be enacted or abolished, and are represented by a set of boolean variables. Every enacting choice creates discontent in the walkers, because it impacts on their daily schedule, but at the same time speeds up the eradication process in a unique way, which can be combined with the benefits of other choices. The choices include both the enactment and abolishment version. When a choice comes into effect, it cannot be undone for at least a *measure duration period* to avoid an "action repetition" effect. These choices are:

1. **No Operation:** Idle choice, does nothing.
2. **Mandatory Mask:** Reduces the probability to infect people in range.
3. **Mandatory Safe Distance:** Imposes a safe distance between the walkers. It's worth noting that walkers may disobey with a probability defined by the *disobedience* parameter, and get closer regardless.
4. **Quarantine infected:** Lock infected Walkers in their houses, to keep them away from the other walkers. Asymptomatic and "in incubation" walkers are not considered, and can still roam and infect other walkers.
5. **Closing Locations:** Set a specific type of location (Schools, Leisures or Workplace) as "unavailable", unabling the walkers to enter them. This choice is subdivided into 3 separate choices.

4. Results

We implemented the agent's DQN as a 1D CNN which takes all the mentioned observations in a time window of k = 6 steps. The network includes 2 1D Convolution layers, a 1D Max Pooling layer. The result is flattened and applied to 2 fully connected layers with dropout. The learner uses the MSE loss function and the Adam optimizer with a learning rate of 0.001. The network is trained on samples of the memorized tuples (state, action, reward, terminal) every step. Here the DQN tries to fit the quality function over n-steps [3], updated with the tuples data:

$$Q(s_t, a_t) = R_{t+1} + \gamma R_{t+2} + ... + \gamma^n \max_{a_{t+n}} Q(s_{t+n}, a_{t+n})$$

where $\gamma = 0.9$ is the discount factor which weights the future reward with respect to the current reward, and $n = 10$ is the number of time steps to accumulate the future reward.

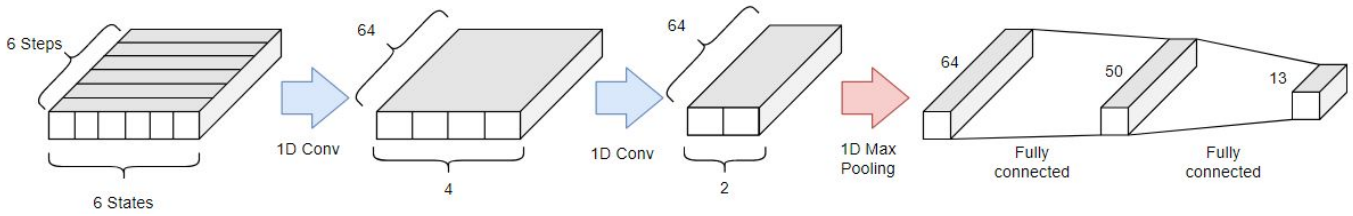

Figure 2: Architecture of the DQN. It extracts the temporal features in the first section, and computes the final choice based on 2 fully connected layers.

For the first iterations, the net tries to explore the possible actions by choosing randomly (decided by an *exploration* factor which decays with time). We implemented the environment using OpenAI Gym [4] and the agent with Keras. For this experiment the city have 100 houses (~330 walkers), and a virus with a 2 meters range of infection, 0.3 infection probability, 0.8 severity (~80% of incubated walkers becomes symptomatic) and 0.4 lethality (~40% of symptomatic walkers die). For the reward function we penalize deaths and death ratio a lot as the main goal of the system is to try and minimize deaths instead of just trying to keep the infection low. From the results we can see that the agent does not perform very well on this task, because after a certain number of episodes, the agent just starts enacting and abolishing its measures on the *measure duration period* (which we set to 14 days). Other techniques such as 1-step (which updates the quality function looking just at one step in the future) proved to be inefficient because this task involves trying to predict future effects from a choice. Due to the complexity of the task, more research is needed to address this problem.

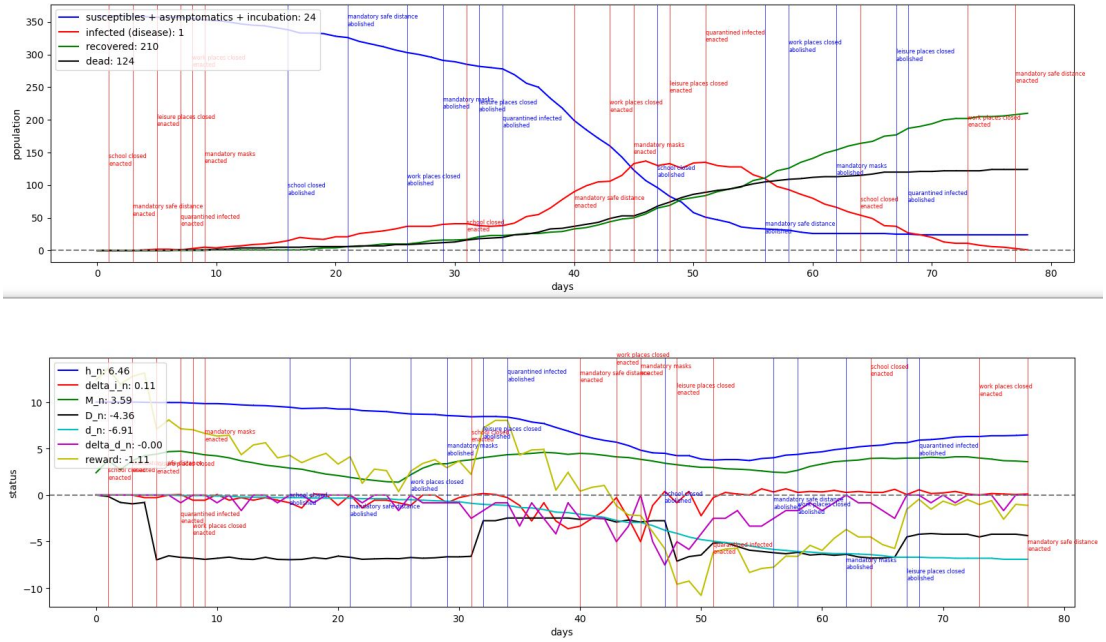

Figure 3: first episode. It is possible to see how the agent explores every choice randomly.

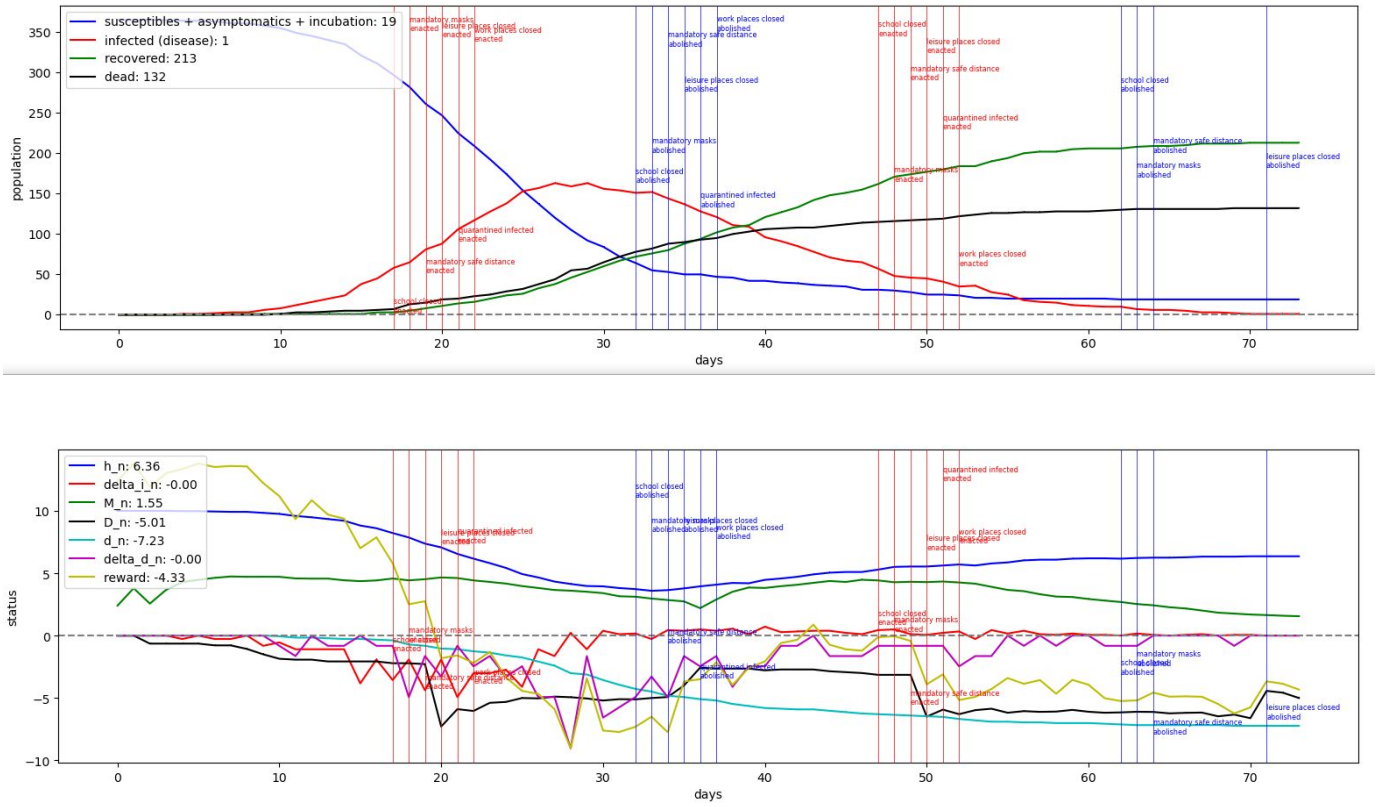

Figure 4: episode 10. We can see that less and more targeted choices are taken

5. Further works

In this paper we have presented a framework for pandemic evolution simulation and the results of our attempt on training an agent to take the right decisions in order to stop the pandemic spreading. As we built it, this framework is highly modular and allows for the easy implementation of new concepts. We have thought about other features that can be added to our framework.

Swab the whole family of a symptomatic infected or a dead person (with a certain percentage of success) in order to find more infected walkers.

Hospitals are not considered in our framework for simplicity. They are an important aspect of pandemics handling and may be added in future releases, leading to problems like critical cases, recovering and quarantined stays.

Circular economy (walkers earn from work, workplaces earn from groceries shops which buy products, and groceries earn from clients) in order to evaluate money related discontent better.

Fake news spreading and disobedience growing.

Temperature measuring at school or work entering, in order to discover infected people (with a certain percentage of success).

Efficiency increases to enable faster simulations. We employed a heavy use of vectorized operations where possible to compute random walks, and other tasks. Increasing the number of houses leads to slower episodes and introducing techniques like parallelization may help.

Agent performances are not high enough to successfully solve the task. One could try other NN architecture models, modify the reward function or also fine-tune the parameters we used in our simulations. Much of the complexity in the reinforcement learning task lies in forging the right reward function. Some suggestion may be to penalize measures abolishing during the infection curve ascent or enactment when the pandemic has finished. Also, a good reward may be given when the curve is kept lower.

Bibliography

[1] Jing Peng, Ronald J. Williams, "Incremental Multi-Step Q-Learning", Editor(s): William W. Cohen, Haym Hirsh, "Machine Learning Proceedings 1994", Morgan Kaufmann, 1994, Pages 226-232, ISBN 9781558603356, https://doi.org/10.1016/B978-1-55860-335-6.50035-0. (http://www.sciencedirect.com/science/article/pii/B9781558603356500350)

[2] Mnih, V., Kavukcuoglu, K., Silver, D. et al. "Human-level control through deep reinforcement learning". Nature 518, 529–533 (2015). https://doi.org/10.1038/nature14236

[3] Hernandez-Garcia, J. & Sutton, Richard. (2019). Understanding Multi-Step Deep Reinforcement Learning: A Systematic Study of the DQN Target

[4] Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie and Zaremba, Wojciech OpenAI Gym. (2016). , cite arxiv:1606.01540.