EPIDEMIC: Effective Planning and Interactive Disease Environment for Managing Infectious Containment

Andrea Miele, Leila Aissa, Marine Moutarlier

EPFL

EPIDEMIC Environment

Our simulation framework is designed to model the spread of a disease within a simple grid-based environment, facilitating the development and evaluation of intelligent vaccination strategies. The simulator includes a representation of the environment and a probabilistic approach to propagate the disease. At each time step, the simulator updates the environmental state based on agent actions and the inherent rules of disease progression. We run all the experiments with 40 available vaccines at the start and 3 propagation days. The following probabilities are used during the simulation: $p_{\text{infection}} = 0.5$, $p_{\text{recovery}} = 0.1$. We decay the number of vaccines available during the training using a step function. We initialize the agent position randomly.

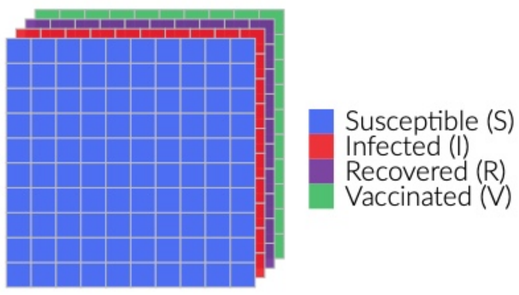


Figure 1. Illustration of the grid environment with dimensions 4x10x10. Each layer (channel) represents a state of the cells within the grid: (S), (I), (R), and (V).

Agent Actions
Move Up
Move Down
Move Left
Move Right
Vaccinate

Figure 2. Agent Action Table

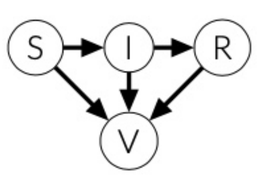


Figure 3. State Transitions Diagram

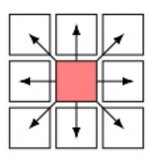


Figure 4. Infection Spread to Neighboring Cells

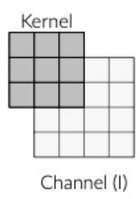


Figure 5. Infection Spread through Convolution

Our observation space contains the whole grid and the agent position and is represented as a 1D array of size $N^2 + 2$.

A critical component for effective learning is the design of the reward function. We use the state of the environment to compute rewards. The simple reward function, R_{SIMPLE}, is defined as:

$$R_{\text{SIMPLE}}(s) = -c \cdot \frac{|I(s)|}{n},$$

where c is a scaling constant, $|I(s)|$ denotes the number of infected cells, and n is the total number of cells in the environment. This negative reward proportional to the number of infected cells incentivizes the agent to reduce infection rates.

A more sophisticated reward function is formulated to better capture the trade-offs between different states. This reward function thus promotes a higher number of susceptible individuals (not infected) while minimizing the number of infections. The function also introduces a penalty for vaccinating reflecting logistical challenges and a limited number of vaccines.

$$R(s) = \frac{50|S(s)|}{n} - \frac{30|I(s)|}{n} - \left(\frac{10|V(s)|}{n}\right)^2$$

where $S(s)$, $I(s)$, and $V(s)$ represent the number of susceptible, infected, and vaccinated cells. This is computed at the end of each episode.

We also introduce a continuous reward, computed at every step. These intermediate rewards are calculated based on various actions and states as follows:

$$r_{\text{intermediate}} = r_{\text{exploration}} + r_{\text{agent near I}} + r_{\text{vaccines near I}} + r_{\text{vaccines near V}}$$

Other tricks tried

Simulation Look Ahead and backtracking: This involves simulating future states to make more informed decisions. You can for instance compute an intermediate reward based on how the disease spreads.

Centering the Grid on the Disease: We center the observation (egocentric view) on the disease to focus on the most critical areas.

Unfortunately, we didn't see any improvement using these two tricks in our experiments.

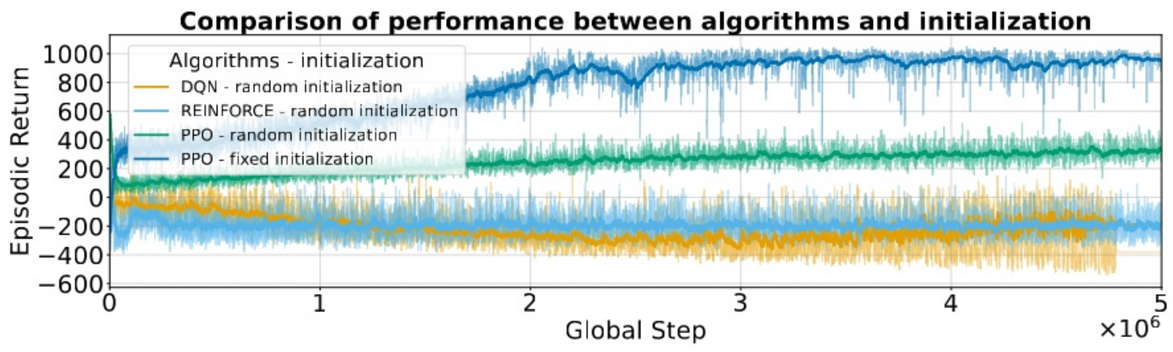


Figure 6. Comparison of the different rewards for the algorithms tested

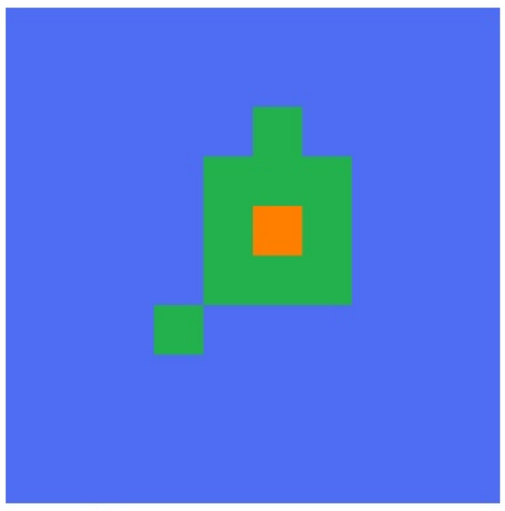
Limitations and future work

The implementation of PPO manages to solve our infection problem for a fixed position in the grid. However, for a random one, it does not due to the added stochasticity. Our implementation of DQN (Deep Q-Networks) and REINFORCE are also not performing well. We could more finely shape the reward and tune the hyperparameters to achieve successful and stable learning. Once achieved, we could train an agent on larger grids, enhance the simulation's realism, and increase initial infected cells. A multi-agent simulation could also be considered.

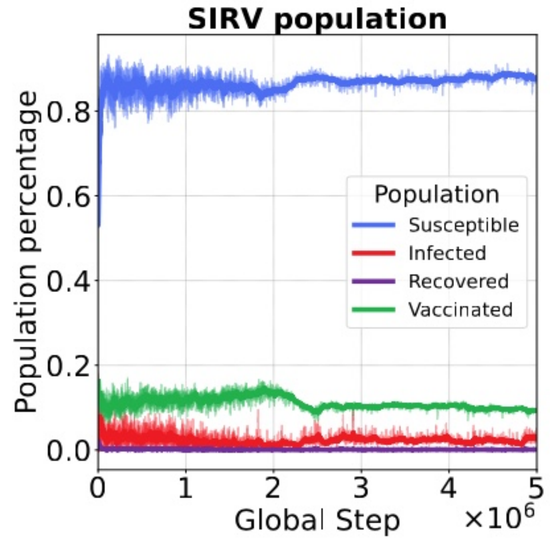
References

Algorithms for reinforcement learning, Szepesvári, Csaba, 2022, Springer nature
Clean-rl https://docs.cleanrl.dev/rl-algorithms/
REINFORCE https://github.com/pytorch/examples/blob/main/reinforcement_learning/reinforce.py
Reinforcement learning class, by Cevher Volkan, EPFL https://moodle.epfl.ch/course/
William Shen, Aidan Curtis, Wildfire Prevention and Management using Deep Reinforcement Learning, 2022 https://williamshen-nz.github.io/firehose/firehose-paper.pdf
Ravi N. Haksar1 and Mac Schwager, Distributed Deep Reinforcement Learning for Fighting Forest Fires with a Network of Aerial Robots, 2018
Farama Foundation. Gymnasium documentation,2024 https://gymnasium.farama.org/
Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. Journal of Machine Learning Research, 23(274):1–18, 2022

Experiments



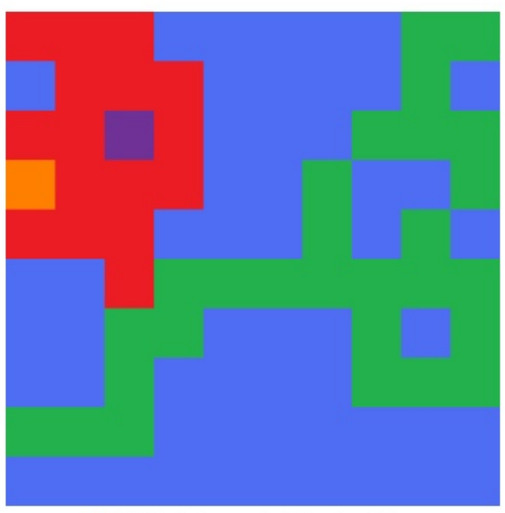
(a) Grid at the end of the training



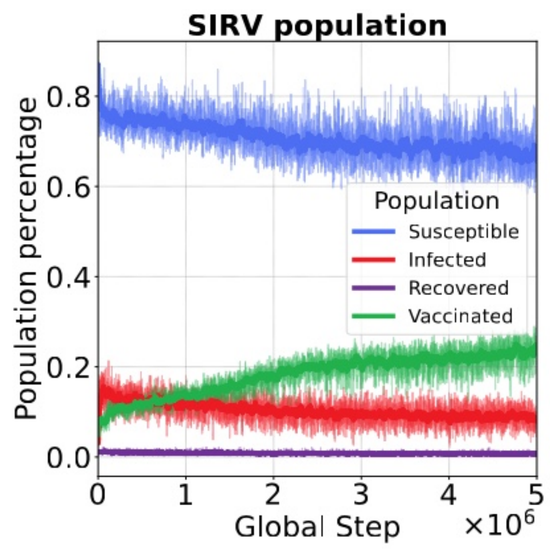
(b) Evolution of the different populations during training

Figure 7. PPO with a centered starting infected cell

In these experimental settings, we manage to solve this environment with PPO (Proximal Policy Optimization). The agent successfully contains the disease propagation by learning a ring strategy around the initial disease cell.



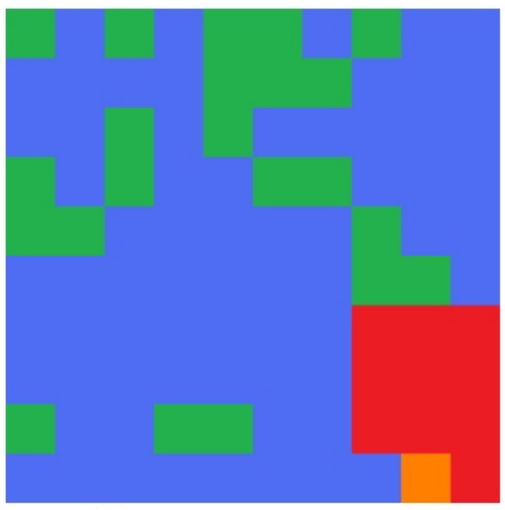
(a) Grid at the end of the training



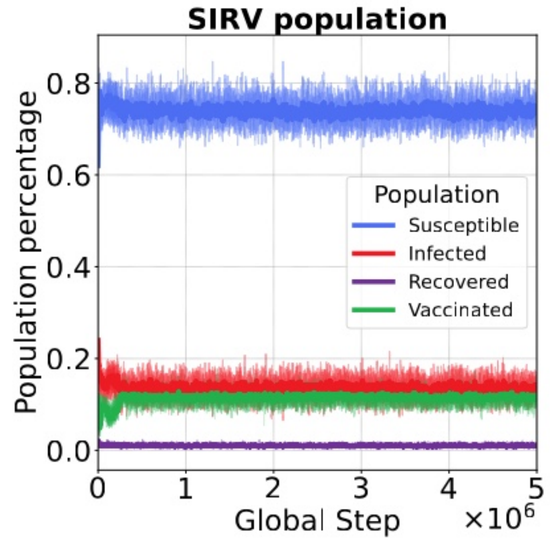
(b) Evolution of the different populations during training

Figure 8. PPO with a random starting infected cell

Here, we manage to see an emerging strategy. The infection is contained with a diagonal wall of vaccinations.



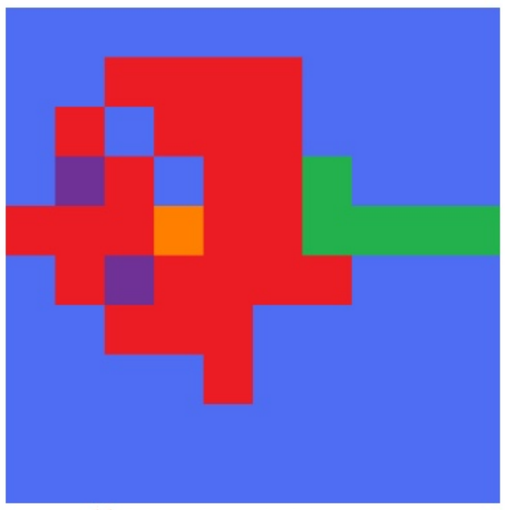
(a) Grid at the end of the training



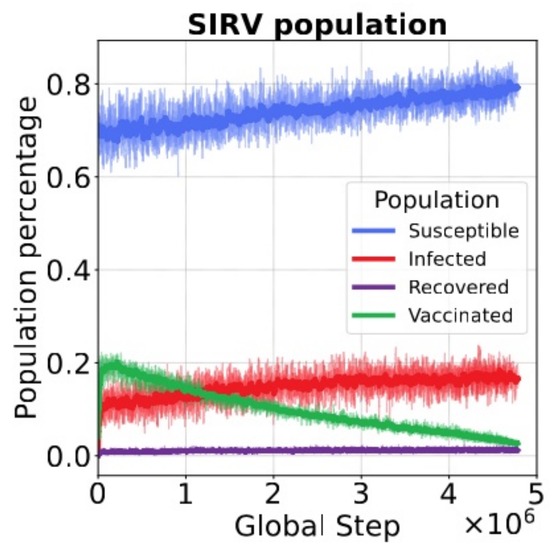
(b) Evolution of the different populations during training

Figure 9. REINFORCE with a random starting infected cell

We compare the PPO algorithm with REINFORCE using the same hyperparameters. Here REINFORCE seems to be learning. The number of infectious people is slightly dropping, and the one of vaccinated is slightly rising. Both are a good sign.



(a) Grid at the end of the training



(b) Evolution of the different populations during training

Figure 10. DQN with a random starting infected cell

We compare the PPO algorithm with the DQN using the same hyperparameters. Here DQN does not seem to find a solution. Moreover, the number of infected people is rising, and the number of vaccinated people is falling.