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

Andrea Miele, Leila Aissa, Marine Moutarlier
EPFL
andrea.miele@epfl.ch, leila.aissa@epfl.ch, marine.moutarlier@epfl.ch

Abstract

This project aims to develop a vaccination strategy using reinforcement learning to mitigate the spread of a disease. In our environment, an agent moves across a grid and decides whether to vaccinate individuals. The grid represents people who can be in one of four states: susceptible, infected, vaccinated, or recovered (SIRV). We employed various RL algorithms PPO, DQN, and REINFORCE, to observe the emergence of vaccination strategies. While we successfully solved the environment using PPO with a fixed initialization, we achieved a functional yet suboptimal strategy with random initialization. Furthermore DQN and REINFORCE can not solve this environment. Codebase is available on GitHub.

1 Introduction

In the face of a global pandemic, developing effective vaccination strategies is crucial to mitigate the spread of infectious diseases. In recent years, reinforcement learning (RL) has demonstrated remarkable success in solving complex decision-making problems. This project focuses on applying RL to develop an effective vaccination strategy to control the spread of a disease in a grid-based environment, see Fig. 1. The grid represents individuals who can be in one of four states: susceptible, infected, recovered or vaccinated (SIRV). Our agent, operating in this environment, must navigate the grid and decide whether to vaccinate individuals based on observed states to minimize the spread of the disease.

In this study, we explore the use of Proximal Policy Optimization (PPO), Deep Q-Network (DQN), and REINFORCE algorithms to observe the emergence of effective vaccination strategies. Our experiments revealed that while PPO efficiently addressed the challenge, resulting in an optimal strategy with fixed initialization, it produced only a functional but suboptimal strategy when random initialization was employed. On the other hand, both DQN and REINFORCE struggled to develop effective strategies in this complex environment. These results underscore the critical role of algorithm choice in reinforcement learning applications, especially in scenarios involving dynamic and multifaceted environments like disease spread simulations. This report discusses the methodologies, challenges, and outcomes of our approach, highlighting the potential and limitations of different RL techniques in public health strategy development.

The main contributions of this work are:

1. Creating a grid-based reinforcement learning environment to simulate disease propagation and vaccination strategies.
2. Implementing and testing the performance of different reinforcement learning algorithms in this environment.
3. Implementing and testing the performance of different reward-shaping strategies in PPO.

Preprint. Under review.

4. Open-sourcing our code and run histories, providing a comprehensive and reproducible codebase for studying this environment.

2 Background

2.1 Reinforcement Learning

Reinforcement Learning (Sutton & Barto, 2018; Cevher) formalizes the interaction between an agent and an environment using the finite-horizon undiscounted Markov decision process (MDP), defined by finite sets of states S and actions A, and a reward function $r : S \times A \times S \to \mathbb{R}$. The agent's goal is to maximize its expected return over the induced random trajectories.

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{t_{\max}-1} R_{t+1}\right] \tag{1}$$

2.2 Deep Q-Network (DQN)

Deep Q-Network (DQN) (Mnih et al., 2015) combines Q-Learning with deep neural networks, using a neural network to approximate the Q-function. DQN employs experience replay and a target network to stabilize learning. The loss minimized by DQN is:

$$\min_{w} L(w) = \mathbb{E}_{(s,a,r,s')\sim D}\left[\left(r + \gamma \max_{a'} Q(s', a'; w') - Q(s, a; w)\right)^2\right] \tag{2}$$

Here, D represents the experience replay buffer, w are the parameters of the Q-network, r is the reward, γ is the discount factor, and $Q(s, a; w)$ is the Q-value.

2.3 REINFORCE

REINFORCE (Williams, 1992) is a fundamental policy gradient method that optimizes the policy by estimating the gradient of the expected return. The gradient estimator is:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[R(\tau)\sum_{t=0}^{\infty}\nabla_\theta \log \pi_\theta(a_t|s_t)\right] \tag{3}$$

2.4 Actor-Critic Agent

We use on-policy deep actor-critic agents, consisting of a policy network $\pi(\cdot;\theta)$ (the actor) and a value network $\hat{v}(\cdot; w)$ (the critic). The critic minimizes the Euclidean distance to an estimator of the returns, such as G_t. We use λ-returns computed with the Generalized Advantage Estimator (GAE) (Schulman et al., 2015). The actor is trained using Proximal Policy Optimization (PPO) (Schulman et al., 2017).

2.5 Proximal Policy Optimization (PPO)

PPO-Clip, the most popular variant of PPO Schulman et al. (2017), optimizes the actor by maximizing the surrogate objective:

$$L^{\pi_{\text{old}}}_{\text{CLIP}}(\theta) = \mathbb{E}_{\pi_{\text{old}}}\left[\sum_{t=0}^{t_{\max}-1} \min\left(\frac{\pi_\theta(A_t|S_t)}{\pi_{\text{old}}(A_t|S_t)}\Psi_t, \text{clip}\left(\frac{\pi_\theta(A_t|S_t)}{\pi_{\text{old}}(A_t|S_t)}, 1+\epsilon, 1-\epsilon\right)\Psi_t\right)\right] \tag{4}$$

Here, ϵ is a small hyperparameter, π_{old} is the policy used to collect the training batch, and Ψ_t is an estimator of the advantage of π_{old}. PPO ensures a trust region that keeps policies close to each other, resulting in stable policy improvement.

3 EPIDEMIC

Our simulation framework is designed to model the spread of a disease within a simple grid-based environment, facilitating the development of optimal vaccination strategies. The environment includes a representation as a grid and a probabilistic approach to propagate the disease. At each episode, the simulator updates the environmental state based on agent actions and the inherent rules of disease progression.

Environment The environment is represented as a three-dimensional grid with dimensions corresponding to channels (C), height (H), and width (W). For our experiments we use a 4x10x10 grid-based environment, See Fig. 1. Each cell within this grid can occupy one of four possible states: susceptible (S), infected (I), recovered (R), or vaccinated (V). We use four channels to encode the state of each cell. Initially, all cells are set to susceptible, one infected cell to initialize disease outbreak. Depending on the experiments whether it is fixed position or random position, the starting points of infection, termed "ignition points", may or may not vary between episodes, ensuring diverse scenarios for the agent to navigate. The agent has 200 steps to figure a way to stop the disease from spreading. Those steps are described in the 3.1 Action space section. Then the simulation is followed by a 3 days of disease propagation where the agent is stable. The total number of global steps taken in our implementation is 5 million.

Disease Propagation For disease transmission, each infected cell has a probability $p_{\text{infection}}$ of infecting its adjacent cells. We employ convolution operations to efficiently simulate the spread of infection across the grid, optimizing computational performance. Additionally, each cell has a recovery probability p_{recovery}, whereupon recovery, transitions to a recovered state. We assume that vaccinated cells are immune, hence unable to be infected later on. The following probabilities are used during the experiments: $p_{\text{infection}} = 0.5$, $p_{\text{recovery}} = 0.1$.

Motivation for Probability Choices: The choice of $p_{\text{infection}} = 0.5$ reflects a moderately high transmission rate, which is common in highly contagious diseases such as measles or the initial stages of COVID-19 without interventions or in child to child transmission.(Michiel van Boven & Bruijning-Verhagena (2023)). This probability ensures that the disease can spread significantly across the grid, posing a substantial challenge to the agent.

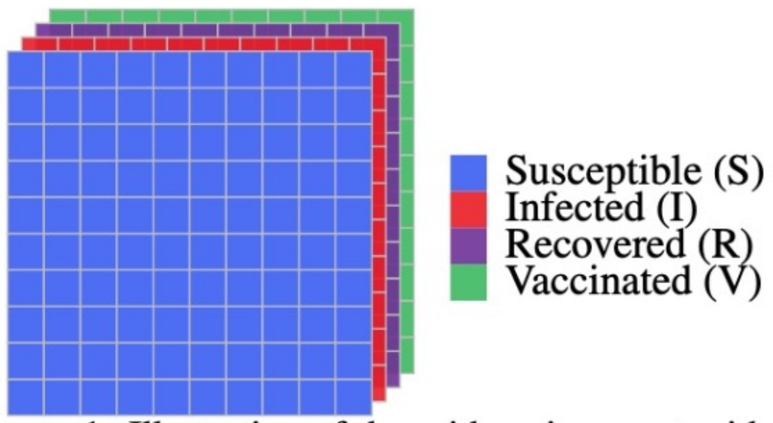


Figure 1: Illustration of the grid environment with dimensions 4x10x10

The recovery probability $p_{\text{recovery}} = 0.1$ is chosen to reflect the natural course of recovery over time without medical intervention. This value balances the dynamics of the model, ensuring that the disease does not persist indefinitely while still requiring active intervention to control the spread.

3.1 RL Environment Interface

EPIDEMIC conforms to the Gymnasium (Towers et al., 2023) wrapper for easy integration with existing frameworks.

Action space: The agent has a discrete set of five actions: move right, move left, move up, move down, or vaccinate. These actions allow the agent to navigate through the grid and implement vaccination strategies to control the disease's spread. The agent is initialized randomly on the grid. Formally, the action space A can be defined as:

$$A = \{\text{move right, move left, move up, move down, vaccinate}\} \tag{5}$$

Observation space: Our observation space contains the whole grid and the agent position and is represented as a 1D array of size $N^2 + 2$. The first N^2 elements represent the state of the grid, where each cell can be susceptible, infected, recovered or vaccinated. The last two elements represent the agent's current position (x, y) on the grid. Thus, the observation space O can be described as:

$$O = \text{grid\_state} \cup \{(x, y)\} \tag{6}$$

where grid_state $\in \{0, 1, 2, 3\}^{N^2}$, with 0 for susceptible, 1 for infected, 2 for recovered and 3 for vaccinated cells, and (x, y) denotes the agent's coordinates.

Reward 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, proposed by (William Shen), R_{SIMPLE}, is defined as:

$$R_{\text{SIMPLE}}(s) = -c \cdot \frac{|I(s)|}{n} \tag{7}$$

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. It is given as:

$$R_{\text{FINAL}}(s) = \frac{50|S(s)|}{n} - \frac{30|I(s)|}{n} - \left(\frac{10|V(s)|}{n}\right)^2 \tag{8}$$

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 of the agent (see Eq. 5 These intermediate rewards (details in Appendix B.2) are calculated based on various actions and states as follows:

$$r_{\text{intermediate}} = r_{\text{exploration}} + r_{\text{agent near I}} + r_{\text{vaccines near I}} + r_{\text{vaccines near V}} \tag{9}$$

We thus introduce a new reward, $R_{\text{CONTINUOUS}}$ defined as:

$$R_{\text{CONTINUOUS}}(s) = R_{\text{FINAL}}(s) + \sum_{i=1}^{200} r_{\text{intermediate}}(i) \tag{10}$$

4 Experiments

Our experiments focus on training reinforcement learning agents in a grid-based environment designed to simulate disease propagation. The environment is structured as a 4x10x10 grid. See Fig.1. The agent's goal is to navigate this grid and implement vaccination strategies to minimize the spread of the disease.

We employed three main algorithms: Proximal Policy Optimization (PPO), Deep Q-Network (DQN), and REINFORCE, to observe and compare their performance in this task.

Proximal Policy Optimization (PPO): PPO is a popular policy gradient method known for its stability and efficiency. It uses clipped probability ratios to ensure that the new policy does not deviate excessively from the old policy during updates, thereby maintaining a balance between exploration and exploitation Schulman et al. (2017). For our experiments, we used standard PPO architectures and hyperparameters, including ReLU activations for the policy and value networks. Both actor and critic networks were trained using the same data, consistent with best practices for stable learning Schulman et al. (2017).

Deep Q-Network (DQN): DQN is a value-based method that combines Q-learning with deep neural networks. It uses experience replay and target networks to stabilize training and break the correlation between consecutive updates Mnih et al. (2015).

REINFORCE: REINFORCE is a fundamental policy gradient method that optimizes the policy by directly estimating the gradient of the expected return with respect to the policy parameters Williams (1992). While simple and unbiased, REINFORCE is known to suffer from high variance, which can slow down the learning process.

To evaluate the performance of these algorithms, we tested this scenario: a randomly initialized infected cell. We maintained a constant learning rate and used the same number of total steps (5 million) distributed over multiple epochs for each configuration.

Metrics and Evaluation: The performance metrics included the number of infections, number of vaccinations, and overall reward. Detailed tables of hyperparameters, model architectures, and environment settings are presented in Appendix B . We also used intermediate rewards to provide more frequent feedback to the agents, enhancing their learning efficiency.

We run three seeds per hyperparameter configuration and report mean curves with min/max shaded regions unless specified otherwise. All curves are smoothed using an exponentially weighted moving average with a coefficient of 0.01. We explore the following questions:

Q1. How effective are shaping rewards in improving the agent's ability to learn efficient vaccination strategies in PPO for a fixed initialization? Does this approach lead to better long-term disease control compared to a simple reward structure?
Q2. How do different initialization strategies (fixed vs. random infected cell) affect the performance and stability of the PPO agent? Does a fixed initialization provide a more stable learning trajectory? How does DQN, and REINFORCE algorithms compare in performances for a random infected cell?

4.1 Effectiveness of Shaping Rewards in PPO for Vaccination Strategies with Fixed Initialization

In this section, we explore various reward shaping techniques to enhance the Proximal Policy Optimization (PPO) agent's ability to learn efficient vaccination strategies when the infected cell is fixed at the center of the grid. Reward shaping is crucial in guiding the learning process of reinforcement learning agents, particularly in complex tasks where simple reward structures may fail to provide sufficient feedback.

We begin with a simple reward structure and then progressively introduce more sophisticated reward functions to address the limitations observed in initial experiments. Additionally, we test advanced methods such as simulation look ahead, backtracking, and centering the observation on the disease. These methods aim to provide more granular feedback and help the agent understand the long-term effects of its actions. Despite the theoretical benefits of these advanced techniques, our experiments show varying degrees of success in improving the agent's performance in controlling the spread of the infection.

4.1.1 Simple reward

In our initial experiments, we employed a simple reward function designed to incentivize the reduction of the infected population, $R_{\text{SIMPLE}}(s)$, see Eq. 7.

We tested this simple reward function using the Proximal Policy Optimization (PPO) algorithm with a fixed starting position for the infected cell. Despite extensive training, the results were unsatisfactory. The vaccinated population quickly dropped to zero, while infections surged to the maximum. This suggests the simple reward structure fails to guide the agent in learning effective vaccination strategies. The agent's poor performance indicates the need for more sophisticated reward mechanisms to capture the task's complexity and provide more detailed feedback.

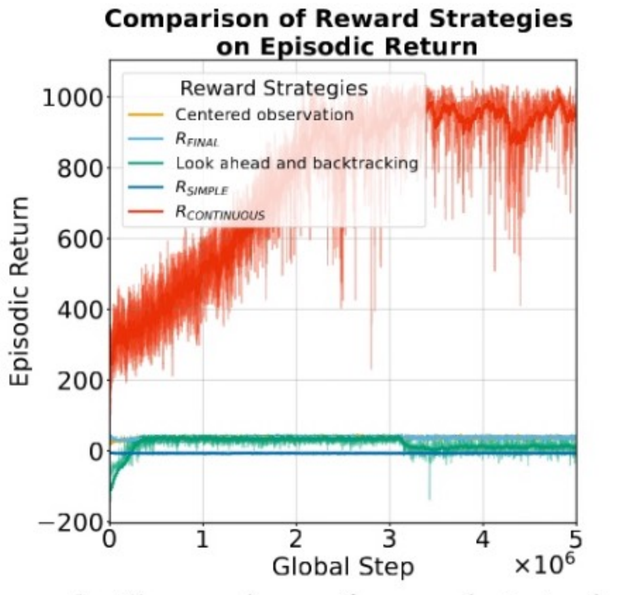


Figure 2: Comparison of reward strategies on episodic return.

4.1.2 Enhancing Reward Structure for Improved Learning

To address the limitations of the simple reward function, we formulated a more sophisticated reward function, R_{FINAL},see Eq. 8., designed to better capture the trade-offs between different states in the grid environment.

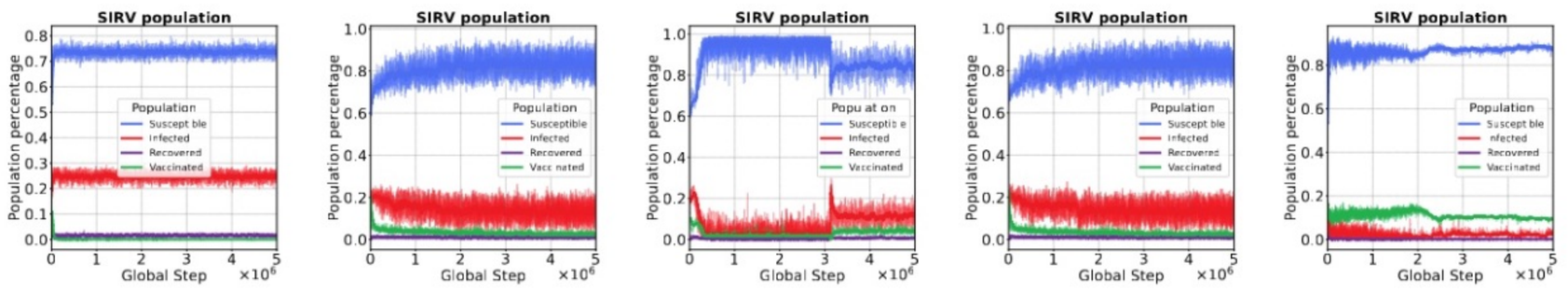
We tested the simple reward function with the Proximal Policy Optimization (PPO) algorithm using a fixed starting position for the infected cell. Despite extensive training, the vaccinated population quickly dropped to zero, while infections surged to the maximum (see Fig. 3(b) and Fig. 3(g)). This outcome suggests that the simple reward structure is inadequate for guiding the agent to learn effective vaccination strategies, highlighting the need for more sophisticated reward mechanisms to capture the task's complexity and provide detailed feedback.

4.1.3 Simulation Look Ahead and Backtracking

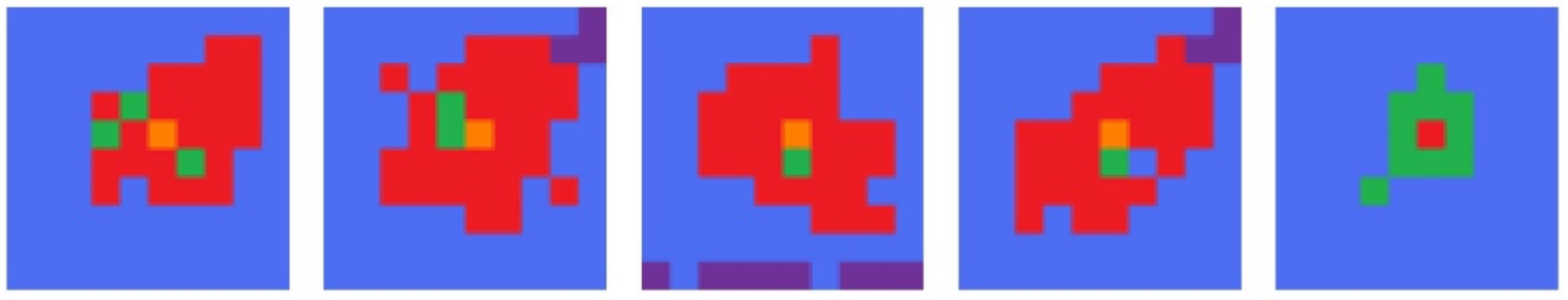
One of the methods we tried to enhance the agent's performance is Simulation Look Ahead and Backtracking. This involves simulating future states to make more informed decisions. We can compute an intermediate reward based on how the disease spreads in these simulated future states. For instance, if the agent simulates k steps ahead, the intermediate reward can be calculated as:

$$r_{\text{sim}}(t) = \frac{1}{k} \sum_{j=1}^{k} (I(t+j) - I(t)) \tag{11}$$

where $I(t)$ is the number of infections at time t. This method helps the agent understand the potential long-term effects of its actions. However, our experiments showed that this approach did not lead to significant improvements in the agent's performance. The final grid states (see Figure 3(h)) and the training progression (see Figure 3(c)) indicate that the agent was still unable to effectively control the spread of the disease using this strategy.



(a) R_{SIMPLE} (b) R_{FINAL} (c) Look ahead strategy (d) Centered observation (e) $R_{\text{CONTINUOUS}}$



(f) Grid R_{SIMPLE} (g) Grid R_{FINAL} (h) Grid Look ahead (i) Grid centered observation (j) Grid $R_{\text{CONTINUOUS}}$

Figure 3: Comparison of PPO performance using different reward functions. The first row shows the evolution of different populations during training with fixed initial infection. The second row displays the corresponding final grid state.

4.1.4 Centering the observation on the disease

Another technique we explored is centering the grid on the disease, providing the agent with an egocentric view of the most critical infection areas. This transformation was intended to help the agent prioritize actions.

However, this approach did not significantly improve performance in our experiments. Since the disease is always fixed at the center, centering the observation might not offer the expected benefits. Figure 2 shows that the reward is identical to the final reward, indicating no additional benefit. The final grid states (Figure 3(i))) and training progression (Figure 3(d)) confirm that centering the observation on the disease did not enhance the agent's ability to control infection spread.

4.1.5 Intermediate Rewards

To mitigate the issues arising from sparse rewards, we introduce intermediate rewards. The reward function $R_{\text{CONTINUOUS}}(t)$ at time step t is given by:

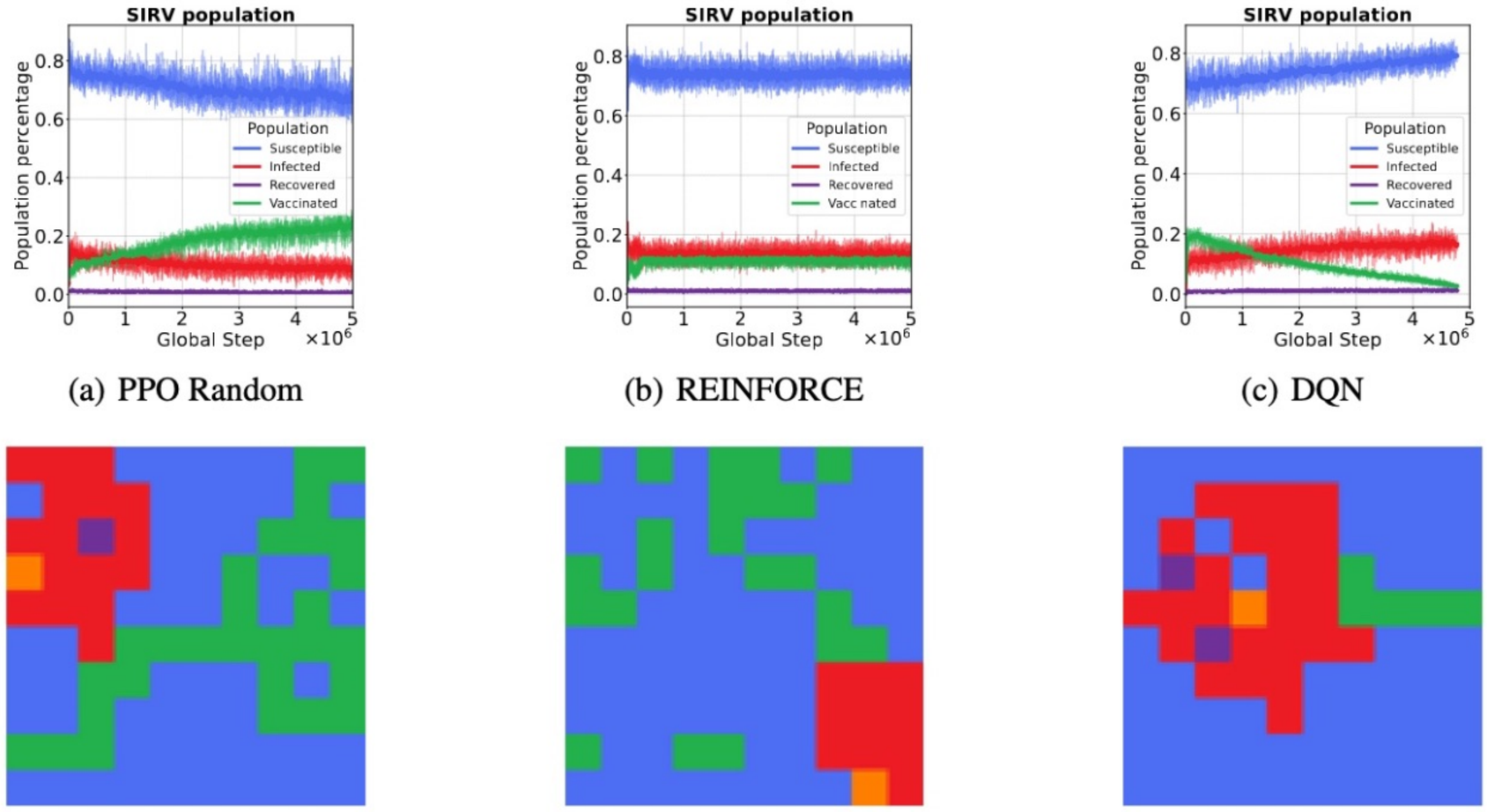
$$R_{\text{CONTINUOUS}}(t) = R_{\text{FINAL}} + \sum_{i=1}^{t-1} r_{\text{intermediate}}(i) \tag{12}$$

where R_{FINAL} is the final reward based on the total number of infections and recoveries, and $r_{\text{intermediate}}(i)$ are the intermediate rewards given at each step i. The intermediate rewards are designed to reflect immediate improvements or deteriorations in the state of the disease spread.

In our experiments, the Proximal Policy Optimization (PPO) algorithm demonstrated a successful vaccination strategy when the infected cell was fixed at the center of the grid. The visualizations in Figure 3(j) depict the population dynamics during training. The infected population (red curve) remained consistently low, indicating effective containment. Initially, the vaccinated population (green curve) rose slightly, then stabilized, showcasing the efficacy of the intervention strategy. These results highlight the robustness of $R_{\text{CONTINUOUS}}$ in managing and controlling disease spread in a grid-based environment. When looking at the grid at the end of the training, we observe the emergence of an optimal vaccination strategy called the **ring strategy**.

Moreover, $R_{\text{CONTINUOUS}}$ outperforms any other reward shaping strategies tried before (cf. Figure 2). The proportion of susceptible, infected, and vaccinated individuals stabilizes as desired (cf. Figure 3(e)), demonstrating the effectiveness of this reward structure in achieving long-term disease control.

4.2 Performance Comparison of PPO, DQN, and REINFORCE on Random Infected Cell Initialization



(a) PPO Random (b) REINFORCE (c) DQN

(d) Grid PPO Random (e) Grid REINFORCE (f) Grid DQN

Figure 4: Comparison of different algorithms and their resulting grid states. The first row shows the evolution of different populations during training for PPO (fixed and random), REINFORCE, and DQN. The second row displays the final grid states for the same algorithms.

In this section, we examine the performance of three reinforcement learning algorithms—PPO, DQN, and REINFORCE—when dealing with the scenario of a randomly initialized infected cell. This setup adds an element of unpredictability and complexity, challenging the agents to adapt their strategies dynamically to effectively manage the spread of infection. We compare their ability to control the disease spread, highlighting the strengths and weaknesses of each approach. The results provide insights into the robustness and flexibility of these algorithms under varying conditions.

PPO with a Random Infected Cell The scenario with a randomly initialized infected cell required significant vaccination efforts. Figure 4(a) illustrates the population changes over time. The infected population (red curve) remained relatively low, reflecting successful containment strategies, while the vaccinated population increased considerably, demonstrating sustained intervention effectiveness. The emergence of a diagonal wall of vaccinations 4(d) highlights an adaptive strategy to contain the infection. Despite the increased complexity, PPO adapted well, confirming its robustness and flexibility Schulman et al. (2017).

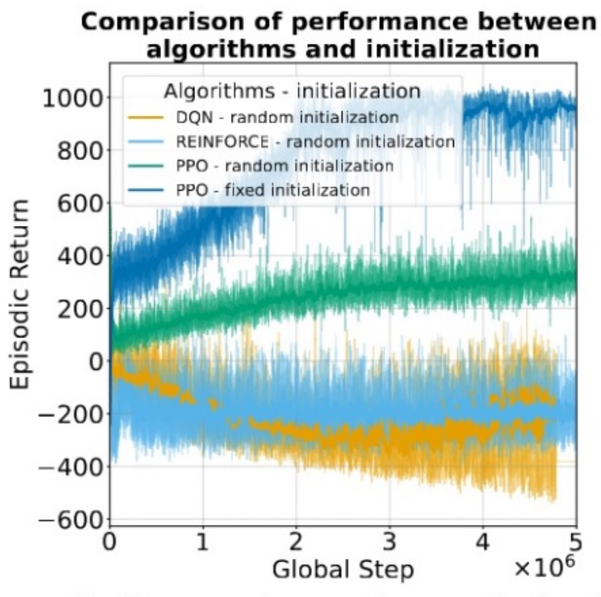


Figure 5: Comparaisons of rewards for the different algorithms used

DQN with a Random Infected Cell When comparing PPO with Deep Q-Network (DQN) using identical hyperparameters, DQN struggled to develop an effective solution. Figure 4(c) shows that the number of infected individuals increased, while the number of vaccinated individuals decreased. This indicates DQN's difficulty in handling the environment's dynamics, including the exploration-exploitation trade-off

and stability issues. The failure of DQN in this context can be attributed to its limitations in learning complex policies in dynamic environments, as discussed in Mnih et al. (2015).

REINFORCE with a Random Infected Cell Comparing PPO with the REINFORCE algorithm using the same hyperparameters, we observed some learning progress with REINFORCE. As shown in Figure 4(b), there was a slight decrease in the number of infected individuals and a modest increase in the number of vaccinated individuals, indicating some level of adaptation. However, PPO still outperformed REINFORCE overall. The variance reduction techniques and more stable policy updates in PPO likely contributed to its superior performance, as supported by Williams (1992); Schulman et al. (2017).

Overall reward for the different algorithms The reward graph as shown in Fig. 5. compares the performance of the various reinforcement learning algorithms tested over training steps. It shows the episodic return, indicating the cumulative reward per episode. The results reveal that all algorithms start with low episodic returns, reflecting their initial learning phase. However, the PPO algorithm, particularly with fixed initialization, demonstrates superior performance, achieving the highest episodic returns consistently. This suggests that fixed initialization offers a more stable and advantageous starting point for learning. In comparison, DQN and REINFORCE show improvement but stabilize at lower episodic returns than PPO. These findings indicate that PPO, especially with fixed initialization, is the most effective for the given task and environment, significantly outperforming DQN and REINFORCE in terms of cumulative reward over the training period.

5 Conclusion and Discussion

Conclusion The complexity of the task and the sparse rewards make it challenging for the agent to learn effective strategies. Introducing intermediate rewards mitigates this by providing more frequent feedback. The Simulation Look Ahead and Backtracking method theoretically aids in understanding long-term effects, while centering the grid on the disease focuses the agent's attention on critical areas. However, our experiments showed that these methods did not significantly improve the agent's learning efficiency.

We found that PPO, particularly with fixed initialization, demonstrated superior performance in managing the vaccination strategy and controlling the spread of infection. PPO's robustness and effective handling of the exploration-exploitation trade-off made it the best-suited algorithm for this task. In contrast, DQN and REINFORCE faced challenges adapting to the environment's dynamics.

Discussion The reward graph comparing different algorithms showed that PPO, especially with fixed initialization, consistently achieved the highest episodic returns. This suggests that a stable starting point is beneficial for learning effective strategies in dynamic environments.

Future work could explore alternative reward structures, advanced simulation techniques, and hybrid models combining various algorithm strengths. Incorporating domain-specific knowledge and constraints could further enhance performance.

Although PPO was the most effective algorithm in our experiments, the challenges faced by DQN and REINFORCE highlight the need for continued research in reinforcement learning for complex real-world problems like disease control.

To improve performance with random initialization, we propose several strategies:

1. **Enhanced Exploration Strategies**: Use advanced exploration techniques such as curiosity-driven exploration to help the agent navigate the state space and discover effective strategies in diverse initial conditions.
2. **Adaptive Reward Shaping**: Dynamically adjust the reward function based on the agent's progress to provide more meaningful feedback and guide the agent towards desirable behaviors.
3. **Curriculum Learning**: Gradually increase task complexity to build foundational skills before tackling more difficult problems, improving the agent's ability to handle random initializations.

Exploring these approaches aims to enhance the agent's ability to manage random initialization and develop robust strategies for disease control.

References

Volkan Cevher. Reinforcement learning class. https://moodle.epfl.ch/course/. Accessed: 2024-5-30.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, and Jeff Braga. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *CoRR*, abs/2111.08819, 2021. URL https://arxiv.org/abs/2111.08819.

Ilse Westerhof Vincent Jaddoe Valerie Heuvelman Liesbeth Duijts d Elandri Fourie Judith Sluiter-Post Marlies A. van Houten Paul Badoux Sjoerd Euser Bjorn Herpers Dirk Eggink Marieke de Hoog Trisja Boom Joanne Wildenbeest Louis Bont Ganna Rozhnov Marc J. Bonten Mirjam E. Kretzschmar Michiel van Boven, Christiaan H. van Dorp and Patricia Bruijning-Verhagena. Estimation of introduction and transmission rates of sars-cov-2 in a prospective household study. *medRxiv*, 2023. doi: 10.1101/2023.06.02.23290879.

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.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

Aidan Curtis William Shen. Firehose : Wildfire prevention and management using deep reinforcement learning. https://williamshen-nz.github.io/firehose/firehose-paper.pdf. [Accessed 30-05-2024].

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

A Additional background

A.1 Exploration vs. Exploitation

The exploration-exploitation trade-off is a fundamental challenge in RL. Exploration involves trying new actions to discover their effects, while exploitation focuses on selecting actions that maximize the expected reward based on current knowledge. Strategies like epsilon-greedy and Upper Confidence Bound (UCB) balance this trade-off by probabilistically choosing between exploration and exploitation.

A.2 Sparse Rewards

Sparse rewards present a significant challenge in RL, as the agent receives feedback only infrequently, making it difficult to learn effective policies. Techniques to handle sparse rewards include reward shaping, where additional intermediate rewards guide the agent towards the desired behavior, and intrinsic motivation, where the agent is rewarded for exploring novel states or achieving subgoals. These methods aim to provide more frequent feedback to the agent, facilitating more efficient learning and policy improvement.

B Experiment details

B.1 Code and run histories

Our codebase is publicly available at `https://github.com/Lei-la1/RL_EPIDEMIC`. It includes the training code, scripts to run all the experiments and the notebook that generated the plots. The codebase uses modified scripts of CleanRL (Huang et al., 2021) to run PPO, REINFORCE and DQN.

The code repository contains links to the Weights&Biases (W&B) project will all of our run histories.

B.2 Details of $\mathbf{r_{intermediate}}$

$$r_{\text{intermediate}} = \begin{cases} -3 & \text{if revisiting a cell} \\ +1 & \text{if visiting a new cell} \\ +5 & \text{if near an infected cell (within distance 1)} \\ +2 & \text{if near an infected cell (within distance 2)} \\ +50 & \text{if vaccinating near an infected cell (within distance 1)} \\ -15 & \text{if vaccinating away from any infected cell} \\ +20 & \text{if vaccinating near another vaccinated cell (within distance 1)} \\ +15 & \text{if vaccinating near an infected cell (within distance 2)} \\ +15 & \text{if vaccinating an infected cell} \end{cases} \tag{13}$$

B.3 Environment and algorithms hyperparameters

Table 1: Disease Spread Simulation Parameters

Parameters	**Value**
$p_{\text{infection}}$	0.5
p_{recovery}	0.1
Number of first infected	1
Max environment steps per episode	200
Number of diseases propagation days	3
Number of available vaccines	40

Table 2: Hyperparameters for PPO.

Environment	
Total timesteps	5,000,000
Number of environments	1
Number of steps	200
Max environment steps per episode	200
Initial infected cell	Fixed at (5,5) or Random
Models (Actor and Critic)	
Activation	ReLU
Critic Linear Layers	[64, 64, 1]
Actor Linear Layers	[64, 64, num_actions]
Optimization	
Optimizer	Adam
Learning rate	0.00025
Gamma	0.99
GAE Lambda	0.95
Number of epochs per update	4
Minibatch size	256
Max gradient norm	0.5
Clip coefficient	0.2
Value loss coefficient	0.5
Entropy coefficient	0.01
Policy	
Clip value loss	True
Normalize advantages	True
Logging	
Training	every 0.1% of total env steps

Table 3: Hyperparameters for REINFORCE.

Environment	
Total timesteps	5,000,000
Log Interval	10
Models	
Input Dimension	Environment specific
Output Dimension	Environment specific
Linear Layers	[128, output_dim]
Dropout Probability	0.6
Activation	ReLU
Optimization	
Gamma	0.99
Optimizer	Adam
Learning Rate	1e-2

Table 4: Hyperparameters for DQN.

Environment	
Total Timesteps	5,000,000
Replay Buffer	
Buffer Size	10,000
Batch Size	128
Exploration	
Start Epsilon	1.0
End Epsilon	0.05
Exploration Fraction	0.5
Models	
Input Dimension	Environment specific
Output Dimension	Environment specific
Linear Layers	[120, 84, output_dim]
Activation	ReLU
Optimization	
Optimizer	Adam
Learning Rate	0.00025
Gamma	0.99
Target Network Update Frequency	500
Tau	1.0
Train Frequency	10