Efficiency of an RL-based Algorithm for Models with an Absorbing Set of States*

Christian Carballo Lozano [a,b], Josu Doncel [b]

[a] Plain Concepts. Bilbao 48001, Spain
[b] UPV/EHU, University of the Basque Country. Leioa 48940, Spain

1. INTRODUCTION

Reinforcement Learning (RL) has emerged as a fundamental approach for solving sequential decision-making problems in environments characterized by uncertainty, complexity, and dynamic evolution. Unlike classical optimization techniques, RL does not require any knowledge of the model under investigation but instead learns optimal policies through interaction and feedback. This makes RL particularly suitable for real-world applications where modeling the entire system accurately is infeasible. Over the past decades, RL has achieved significant success in diverse fields such as robotics [1], healthcare [6], cybersecurity [3], and communication networks [2].

Our work presents an RL-based algorithm that applies to a wide family of models. Specifically, our model only assumes the existence of a set of states $\mathcal{S}^a$ which satisfies the following property: when the agent is in a state of $\mathcal{S}^a$ and takes any of the possible actions, the state that the environment returns also belongs to $\mathcal{S}^a$. This occurs, for instance, in any epidemic model without external infections.

We present an algorithm that leverages the structure of these models. Our numerical results show that our algorithm learns the optimal policy in a more efficient manner than the Q-learning algorithm.

2. MODEL DESCRIPTION

We consider a system that evolves randomly in discrete time. We denote by X_n the state of the system at time slot n and by $\mathcal{S}$ the set of states. In state $s \in \mathcal{S}$, an action $a \in A_s$ is chosen. We assume that $\mathcal{S}$ is finite and that A_s is a countable set for all $s \in \mathcal{S}$. We consider that being in state $s \in \mathcal{S}$ incurs a cost of $c(s)$. Moreover, when in state $s \in \mathcal{S}$ an action $a \in A_s$ is chosen, the state in the next time step is $s' \in \mathcal{S}$ with probability $p_{s,a}(s')$. The decision maker decides on a deterministic policy π, which determines the action to be taken in each state, i.e., $\pi : \mathcal{S} \to A_s$.

An optimal policy, which is denoted as π^*, is defined as the policy that minimizes the value function for all $s \in \mathcal{S}$. The standard way to compute π^* is by solving the Bellman equations [4]. However, in this work, we consider that we do not have any knowledge about the transition probabilities and the costs of the model. Thus, we tackle the problem of finding π^* using an alternative approach based on RL.

The Q-learning algorithm [5] is an RL-based algorithm that provides $\hat{Q}$, which is an estimator of the action-value function. Let s_t and a_t be, respectively, the state and the action taken at time t. The Q-learning algorithm follows the next update rule:

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \eta_t \left(c(s_t) + \delta \min_a \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t) \right), \quad (1)$$

where $\eta_t \in (0, 1]$ and $\delta \in [0, 1)$.

In this work, we make the following assumption.

ASSUMPTION 1. *We assume that there exists a set of absorbing states $\mathcal{S}^a$, i.e., if $s \in \mathcal{S}^a$, then for all $a \in A_s$, $\sum_{s' \in \mathcal{S}^a} p_{s,a}(s') = 1$.*

Under this assumption, it is easy to see that the Q-learning algorithm does not converge to the global optimum in the following cases: (i) when there is a single episode whose length is infinite, and (ii) when in each episode the initial state belongs to $\mathcal{S}^a$. As a consequence, we present an alternative approach in the following section.

3. OUR APPROACH

We notice that, under Assumption 1, the optimal action-value function of the states in $\mathcal{S}^a$ can be learned with any knowledge of the action-value function of the states that are not in $\mathcal{S}^a$. Therefore, we propose an RL-based algorithm that learns the optimal policy in a more efficient manner than the Q-learning algorithm. The pseudocode of our algorithm is shown in Algorithm 1. This algorithm initializes the Q-values randomly and, then, it works in two phases:

- In the first stage, we apply the update rule (1), where the starting state of each episode is a state of $\mathcal{S}^a$ drawn at random. This stage is executed for a fixed number of steps K^a. We consider that, in this stage, there are several episodes to make sure that all the states are visited a minimum number of times.
- In the second stage, we apply the update rule (1) and we consider that an episode ends when one of the following conditions is satisfied: (i) s_{t+1}, i.e., the state of the system at time $t+1$, belongs to $\mathcal{S}^a$ or (ii) the maximum number of steps in that episode is reached. Then, a new episode starts in a randomly selected state in $\mathcal{S} \setminus \mathcal{S}^a$. The second stage ends when the number of iterations equals K^{sim}.

*This work has been partially funded by the Department of Education of the Basque Government through the Consolidated Research Group MATHMODE (IT1456-22).

Algorithm 1 Our approach

Parameters: learning rate $\eta \in (0, 1]$, $\epsilon > 0$, number of steps in the first stage K^a, number of steps of the simulation K^{sim}:
Initialize $Q(s, a)$ arbitrarily and $k = 0$
while $0 \leq k < K^a$ **do** // beginning of the first stage
  Initialize $s_0 \in \mathcal{S}^a$ randomly
  for each step t of episode **do**
    Apply Q-learning update rule (1) with ϵ-greedy policy
    $k \leftarrow k + 1$
    if episode ends **then**
      Initialize new episode
    end if
  end for
end while // end of the first stage
while $k < K^{\text{sim}}$ **do** // beginning of the second stage
  Initialize $s_0 \in \mathcal{S} \setminus \mathcal{S}^a$
  for each step t of episode **do**
    Apply Q-learning update rule (1) with ϵ-greedy policy
    $k \leftarrow k + 1$
    if $s_{t+1} \in \mathcal{S}^a$ or episode ends **then**
      Initialize new episode
    end if
  end for
end while // end of the second stage

4. APPLICATION

We consider a variant of the SIR model with N homogeneous elements that evolve in continuous time. An element encounters another element at a rate γ. If a susceptible element encounters an infected element, then it becomes infected. An infected element gets recovered at rate ρ. A recovered individual cannot be infected until it gets susceptible again at rate β. Moreover, with a vaccination rate α, a susceptible individual becomes recovered directly without being infected. We uniformize the continuous-time Markov chain with a uniformization constant $\Omega < (N(\gamma + \rho + \beta + \alpha))^{-1}$ and obtain a discrete-time Markov chain. In this model, susceptible elements follow a confinement policy (or strategy) $\pi : \mathbb{N} \to [0, 1]$, where $\pi(t)$ is a probability that indicates the level of exposure to the epidemic of that element at time t. More precisely, when $\pi(t) = 1$, susceptible elements are completely exposed to the epidemic at time zero, whereas when $\pi(t) = 0$ they are protected from taking the infection, i.e., they are confined. We consider that there is a confinement cost that applies to each susceptible element. We assume that the confinement cost of a susceptible element at time t is $c_L - \pi(t)$, where $c_L \geq 1$ (remember that $\pi(t)$ is the strategy of susceptible elements at time t). We also consider that there is a cost associated to each infected element; more precisely, each infected element leads to a cost of $c_I > 0$ per unit of time.

We consider that the action space is $\{0, 1, \}$, i.e., susceptible elements are confined or fully exposed to the epidemic.

In this model, $\mathcal{S}^a$ is the set of states where there are no infected elements.

As a representative instance of the general pattern, let us consider the following configuration: $N = 5$, $\gamma = 1.1$, $\rho = 0.6$, $\alpha = 0.2$, $\beta = 0.3$, $c_I = 2$, and $c_L = 1.001$. The discount factor is set to $\delta = 0.99$ and $T = 2000$. In our experiments, we also consider $K^{sim} = 10^6$. We first study the impact of K^a in the performance of our approach, and then we will show how the presented algorithm outperforms Q-learning.

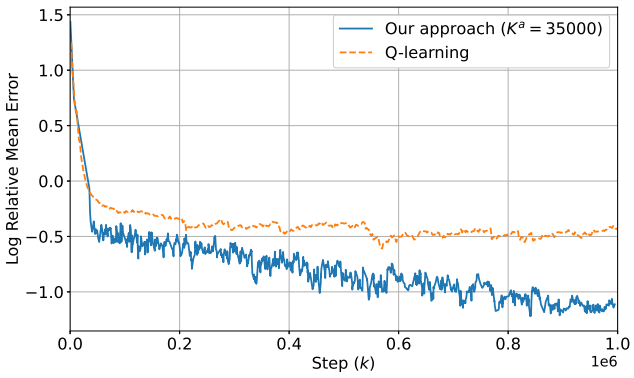

Figure 1: Logarithm of the mean relative error of Q-learning (dashed orange line) and our algorithm (blue line).

We compare the performance of our algorithm with that of Q-learning for this model. Figure 1 shows the evolution of the logarithm of the mean relative error. The dashed orange line corresponds to Q-learning: the error decreases quickly at first, but then slows down. The blue line shows our algorithm with $K^a = 35000$. This illustration shows that both methods perform similarly in the initial phase, but afterwards our approach achieves a faster and more sustained error reduction. The plotted case corresponds to the best performance observed among the tested values of K^a.

5. REFERENCES

[1] P. Abbeel, A. Coates, M. Quigley, and A. Ng. An application of reinforcement learning to aerobatic helicopter flight. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

[2] E. Ghadimi, F. Davide Calabrese, G. Peters, and P. Soldati. A reinforcement learning approach to power control and rate adaptation in cellular networks. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7, 2017.

[3] T. T. Nguyen and V. J. Reddi. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):3779–3795, 2023.

[4] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. J. W. & Sons, 2014.

[5] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[6] M. Tejedor, A. Z. Woldaregay, and F. Godtliebsen. Reinforcement learning application in diabetes blood glucose control: A systematic review. *Artificial Intelligence in Medicine*, 104:101836, 2020.