Framework for solving time-delayed Markov Decision Processes

Yorgo Sawaya, George Issa, and Sarah E. Marzen[*]
W. M. Keck Science Department, Pitzer, Scripps, and Claremont McKenna College, Claremont, California 91711, USA

(Received 22 September 2022; accepted 5 June 2023; published 18 July 2023)

Reinforcement learning has revolutionized our understanding of evolved systems and our ability to engineer systems based on a theoretical framework for understanding how to maximize expected reward. However, time delays between the observation and action are estimated to be roughly ∼150 ms for humans, and this should affect reinforcement learning algorithms. We reformulate the Markov Decision Process framework to include time delays in action, first deriving a new Bellman equation in a way that unifies previous attempts and then implementing the corresponding SARSA-like algorithm. The main ramification—potentially useful for both evolved and engineered systems—is that, when the size of the state space is lower than that of the action space, the modified reinforcement learning algorithms will prefer to operate on sequences of states rather than just the present state with the length of the sequence equal to 1 plus the time delay.

DOI: 10.1103/PhysRevResearch.5.033034

I. INTRODUCTION

Reinforcement learning is, simply put, learning with reinforcement. An agent moves around in some (potentially complicated) world and, by taking various actions, accrues rewards. Its goal is to maximize the sum total of its discounted rewards by adjusting its action policy, or how it reacts to being in a particular world state. This problem is difficult because we are interested in the sum total of discounted rewards *in the future*, and not just the present. Many textbooks [1–3] have been written to elucidate algorithms to solve such problems.

Reinforcement learning is now used to engineer artificial systems, robots included [2]. And it is now nearly canon that different regions of the brain are implementing different kinds of reinforcement learning algorithms [4]. Even so, there has been a surprising lack of attention on a key constraint that affects both biological and engineered systems: time delays. There are delays associated with reception of reward signals, delays associated with taking actions, and delays associated with observation. We focus on the second of these, the action delays: agents take some time Δt to take an action after perceiving the world's state, and this time delay can greatly affect their ability to collect rewards.

Deterministic action delays (hereafter simply referred to as "time delays") and their effect on the reinforcement learning framework have been previously considered [5–8]. Much work has focused on Markov Decision Processes (MDPs), in which the agent fully observes the world state at all times. Previous authors have argued that an MDP with action delays is still an MDP, once transformed, and propose a new set of corresponding variables that the agent should now use to decide upon a time-delayed action given an environmental state. We find a new set of variables for solving MDPs with action delays based on state sequences rather than action sequences. Furthermore, we find that the optimality results for MDPs without time delays (that there is a deterministic optimal policy) do not carry over to the case of MDPs with action delays.

We therefore formulate a new framework for solving MDPs with constant action delays, relying heavily on an early paper by Blackwell [9]. Using this framework, we slightly modify some existing algorithms for solving MDPs so that they can tackle MDPs with action delays. Our main prediction is that brain regions that try to estimate the value of taking an action in a particular state need information about sequences of states or sequences of actions, with the length of the sequences being equivalent (roughly) to the time delay. Which sequences one uses are a balance of at least two factors. First, sequences of states will be preferred biologically if efference copies—signals about actions taken that propagate back to earlier sensory layers, useful for computing using sequences of actions—are costly compared to the machinery required to store sequences of states. But algorithmically, sequences of states may also be preferred if the effective number of world states is less than the effective number of possible actions.

Finally, it seems intuitively as though a time delay between action and perception would need to be taken into consideration in order to find the most effective action policy and would lead to less collected reward, as the information one effectively receives about the state of the world becomes less and less relevant to its actual state as the time delay increases. We show these two things using the framework developed here in a simple example. This has two biological implications. First, the organism must know its own time delays in order to formulate the most effective action policy. And second, populations of organisms would endeavor over evolutionary timescales to

[*]smarzen@cmc.edu

minimize these time delays, similar to the predictions of the minimal cortical wiring hypothesis [10].

The paper proceeds as follows. In Sec. II, we describe the MDP's mathematical formulation and the formulation of an action delay. In Sec. III, we describe our new framework, derive optimality results and comment upon when they cannot be derived, and propose slight modifications of existing algorithms for MDPs. In Sec. III C, we use an example to illustrate that time delay negatively affects collected reward and must be considered in order to collect as much reward as possible. Section IV concludes.

II. SETUP

We imagine a world with states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$. It will turn out that we are interested in sequences of states, $(s_t, \dots, s_{t+\Delta t}) \in \mathcal{S}^{\Delta t+1}$, and sequences of actions, $(a_t, \dots, a_{t+\Delta t}) \in \mathcal{A}^{\Delta t+1}$. When in state s, upon taking action a, an agent receives a reward $r(s, a)$. [The framework can be easily generalized to the important case that $r(s, a)$ is itself probabilistic.] The agent is said to take action a in state s with the probability $\pi(a|s)$, and following Blackwell [9], we instead allow for a nonstationary policy and write $\pi = (\pi_t, \pi_{t+1}, \dots)$ to describe the policy in which $\pi_t(a|s)$ governs the action choice at time t. We simply write $T\pi$ to denote $(\pi_{t+1}, \pi_{t+2}, \dots)$, i.e., T is an operator incrementing the time index by 1. The environment is governed by a transition probability $p(s_{t+1}|s_t, a_t)$, the probability of the environment transitioning to state s_{t+1} given that it was in s_t and that the agent has taken action a_t. The goal is to maximize $r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \cdots$ where $\gamma \in [0, 1)$ is the commonly used discount factor.

In this article, we place a constraint on the relationship between states and actions. When experiencing a state s_t, we choose an action a. This action is, perhaps confusingly, not the same as the action taken at time t, a_t. Instead, this action a is the action taken at time $t + \Delta t$, $a_{t+\Delta t} = a$. This simulates a time delay of Δt between perception and action, and forces us to reformulate the MDP framework. We call this a time-delayed MDP. See Fig. 1 for an explanation.

Altogether, the reward function, the transition probability, the policy, the discount factor, and the time delay describe a time-delayed MDP. Achieving our goal—finding the optimal action policy, perhaps given initial conditions—is called "solving the time-delayed MDP."

To solve the time-delayed MDP, it is not enough to choose an action a_t that maximizes the current reward, as this may adversely affect which states you end up in later. One must consider the long-term behavior of the system. This is commonly done using dynamic programming, which we describe later.

Throughout this article, when doing numerical examples, $|\mathcal{S}|, |\mathcal{A}| < \infty$ such that there are a finite number of actions and states, i.e., the "tabular" case.

III. RESULTS

The results follow in three parts. In Sec. III A, we adapt well-known techniques in the solution of MDPs (dynamic programming) to define and find a value function that describes the value of a *sequence* of successive states. In Sec. III B, we show that there is no optimal policy for when we have a time-delayed MDP. And in Sec. III C, we use existing model-free algorithms for solving MDPs to solve time-delayed MDPs.

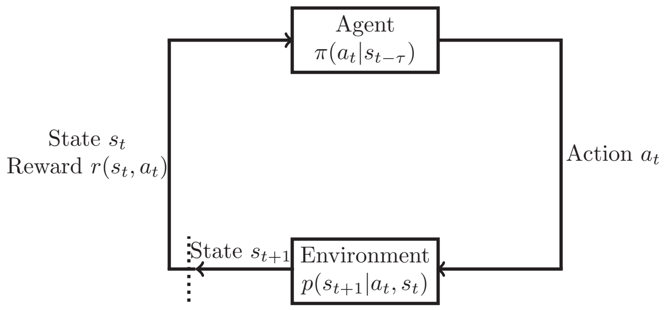

FIG. 1. The setup of a Markov Decision Process (MDP) with action delay τ, or a time-delayed MDP. The reinforcement learning agent has an action policy $\pi(a_t|s_{t-\tau})$, indicating that it uses the sensory perception from a time τ ago to choose its action. The action is taken, and the environment responds by transitioning to a new state s_{t+1} according to the transition probability $p(s_{t+1}|a_t, s_t)$. Meanwhile, the reward accrued is $r(s_t, a_t)$, based on the current environmental state and *current* action.

A. Reformulating Bellman's equations

As stated in Sec. II, solving both the MDP and the time-delayed MDP is not as easy as finding the action that maximizes $r(s, a)$. Such actions may benefit the agent in the short term, but in the long term, these actions may send the agent into undesirable states s for which $\max_a r(s, a)$ is far lower. Hence, planning over long time horizons is required.

An obvious experience-based approach to this issue is the Monte Carlo approach: choose a policy and experience the rewards, and from that calculate $\mathbb{E}_\pi[r(s_1, a_1) + \gamma r(s_2, a_2) + \cdots]$. Then, improve the policy (in a way that we have yet to describe) so that one continually improves this running weighted average. We pursue a typical and different approach that allows us to get beyond the large sample complexities and high variance estimates in Monte Carlo approaches.

We first define a "value function" that indicates the value of being in a particular sequence of states. The idea of this is that some states or sequences of states may be particularly valuable, in that they are likely to lead to higher expected reward. But in defining this value function, we must take some care to also consider another factor. The eventual goal of defining such a value function is to make it easier for us to solve time-delayed MDPs. We would, in the spirit of the Bellman equation for non-time-delayed MDPs, like to formulate our revised Bellman equation so that the value function is easy to compute from the reward function, the transition probability, and the action policy. We therefore choose to define the value of a sequence of states rather than that of a single present state:

$$\begin{aligned} &V_\pi(s_t, s_{t+1}, \dots, s_{t+\Delta t}) \\ &\quad := \mathbb{E}[r(s_{t+\Delta t}, a_{t+\Delta t}) + \gamma r(s_{t+\Delta t+1}, a_{t+\Delta t+1}) \\ &\qquad + \cdots | S_t = s_t, \dots, S_{t+\Delta t} = s_{t+\Delta t}]. \end{aligned} \tag{1}$$

The subscript π indicates that the value function is a function of the policy π. With work shown in the appendices, the linearity of the expectation value and the law of iterated expectation reveal that

$$\begin{aligned} &V_\pi(s_t, s_{t+1}, \dots, s_{t+\Delta t}) \\ &\quad = \sum_{a_{t+\Delta t}} \pi_t(a_{t+\Delta t}|s_t) r(s_{t+\Delta t}, a_{t+\Delta t}) \\ &\qquad + \gamma \sum_{a_{t+\Delta t}, s_{t+\Delta t+1}} \pi_t(a_{t+\Delta t}|s_t) p(s_{t+\Delta t+1}|s_{t+\Delta t}, a_{t+\Delta t}) \\ &\qquad \times V_{T\pi}(s_{t+1}, \dots, s_{t+\Delta t+1}), \end{aligned} \tag{2}$$

where $T\pi$ is the time-shifted action policy sequence (see Sec. II). In the case of a stationary policy $\pi = (\pi, \pi, \dots)$, we have

$$\begin{aligned} &V_\pi(s_t, s_{t+1}, \dots, s_{t+\Delta t}) \\ &\quad = \sum_{a_{t+\Delta t}} \pi(a_{t+\Delta t}|s_t) r(s_{t+\Delta t}, a_{t+\Delta t}) \\ &\qquad + \gamma \sum_{a_{t+\Delta t}, s_{t+\Delta t}} \pi(a_{t+\Delta t}|s_t) p(s_{t+\Delta t+1}|s_{t+\Delta t}, a_{t+\Delta t}) \\ &\qquad \times V_\pi(s_{t+1}, \dots, s_{t+\Delta t+1}). \end{aligned} \tag{3}$$

If we consider $\vec{V}_\pi$ to be a vector of dimension $|\mathcal{S}|^{\Delta t+1}$ of the values of all sequences of states, then one can rewrite the above set of equations as a linear equation, $\vec{V}_\pi = b_\pi + \gamma A_\pi \vec{V}_\pi$, for $(b_\pi)(s_t, \dots, s_{t+\Delta t}) = \sum_{a_{t+\Delta t}} \pi(a_{t+\Delta t}|s_t) r(s_{t+\Delta t}, a_{t+\Delta t})$, and

$$\begin{aligned} &A_\pi(s_{t+1}, \dots, s_{t+\Delta t+1}; s'_{t+1}, \dots, s'_{t+\Delta t+1}) \\ &\quad = \delta_{s_{t+1}, s'_t} \dots \delta_{s_{t+\Delta t}, s'_{t+\Delta t-1}} \\ &\qquad \times \sum_{a_{t+\Delta t}, s_{t+\Delta t+1}} \pi(a_{t+\Delta t}|s_t) p(s_{t+\Delta t+1}|s_{t+\Delta t}, a_{t+\Delta t}). \end{aligned}$$

We can therefore simply solve for $\vec{V}_\pi$ as

$$\vec{V}_\pi = (I - \gamma A_\pi)^{-1} b_\pi. \tag{4}$$

But this calculation can be difficult for many reasons. First, when there are large time delays or when there are many states, the matrix inversion becomes more computationally intensive. Second, we may not know the transition probabilities or reward function, and so we may not know A_π and b_π. Hence, we introduce the temporal difference (TD) learning algorithm for time-delayed MDPs, following the usual formulation for MDPs. One guesses a value function $\vec{\tilde{V}}_\pi$ and takes actions governed by π, updating $\vec{\tilde{V}}_\pi$ at all times to minimize the TD error:

$$\begin{aligned} &\delta \tilde{V}_\pi(s_t, \dots, s_{t+\Delta t}) \\ &\quad = r(s_{t+\Delta t}, a_{t+\Delta t}) + \gamma \tilde{V}_\pi(s_{t+1}, \dots, s_{t+\Delta t+1}) \\ &\qquad - \tilde{V}_\pi(s_t, \dots, s_{t+\Delta t}). \end{aligned} \tag{5}$$

One can view this as an error that encapsulates the gradient descent on a mean-squared error between the left- and right-hand sides of the revised Bellman equation. When used in an algorithm to evaluate a policy, we multiply the TD error by the learning rate α. This governs how quickly we approach the correct value function, although a value of α that is too large can cause us to overshoot the answer or even move away from the answer. In TD, our value function stops changing when the right-hand side is 0, or when the Bellman equation is (on average) solved. As such, if α is small enough, TD updates will cause $\vec{\tilde{V}}_\pi$ to converge. Note that updates to the value function are "model-free," meaning that no model of the environment is needed to update said value function—only experience.

So far, the framework of MDPs has carried over easily to the framework for time-delayed MDPs. But the difference is that one must operate on sequences of states in order to understand time-delayed MDPs. This has a memory cost and a time cost. When using the matrix inversion method (which requires a model), one has a time complexity of roughly $O(|\mathcal{S}|^{3\Delta t})$, and therefore, increases in the time delay Δt exponentially increase the computational power required. While TD does not suffer as much from increases in Δt, we do have to store $\vec{\tilde{V}}_\pi$, and therefore, we find a memory cost of $O(|\mathcal{S}|^{\Delta t})$. We also need to experience all the sequences of states and therefore suffer a sample complexity of at least $O(|\mathcal{S}|^{\Delta t})$.

This is not the only approach to solving time-delayed MDPs. Previous authors [5] have considered state-action value functions of the form

$$\begin{aligned} &Q_\pi(a_{t-\Delta t}, \dots, a_{t-1}, s_t) \\ &\quad := \mathbb{E}[r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) \\ &\qquad + \cdots | A_{t-\Delta t} = a_{t-\Delta t}, \dots, S_t = s_t]. \end{aligned} \tag{6}$$

This formulation seems ill-suited for computing optimal policies, as one would like a *past* state to guide a *future* action in the time-delayed setup—we do not have access to s_t at time t. Otherwise, causality is broken. Hence, one cannot turn this state-action value function into a policy improvement theorem [3], a change to the action policy that allows one to accrue more reward. However, with some modifications [8], one can reformulate the information for optimal action selection to be $a_t, \dots, a_{t+\Delta t}$ and s_t, and define

$$\begin{aligned} Q_\pi(a_t, \dots, a_{t+\Delta t}, s_t) &:= \mathbb{E}[r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) \\ &\qquad + \cdots | A_t = a_t, \dots, S_t = s_t]. \end{aligned} \tag{7}$$

The Bellman equation follows as

$$\begin{aligned} &Q_\pi(a_t, \dots, a_{t+\Delta t}, s_t) \\ &\quad = r(s_t, a_t) + \gamma \sum_{s_{t+1}} p(s_{t+1}|s_t, a_t) \pi(a_{t+\Delta t+1}|s_{t+1}) \\ &\qquad \times Q_\pi(a_{t+1}, \dots, a_{t+\Delta t+1}, s_{t+1}). \end{aligned} \tag{8}$$

One can certainly solve this and convert the resulting solution into the averaged state-action value function, described in a later subsection. And so, the main difference between the formulation with sequences of states and the formulation with sequences of actions is one of computational power. When using temporal difference learning, the memory scales as $|\mathcal{S}|^{\Delta t+1}$ for the Bellman equation of the main text and as $|\mathcal{S}||\mathcal{A}|^{\Delta t}$. Which one is preferred will largely depend on the size of the state space versus the size of the action space.

B. Is there an optimal policy?

The key question in reinforcement learning is not usually, "What is the value of a particular state or sequence of states?", but instead, "What is the policy that I should follow?" In fact, we only care about the value of sequences of states so that we can find an "optimal policy," a point that we flesh out in this subsection.

Let us return to the MDP case. There, our value function only depends on our current state. Our goal is to maximize this value function with respect to policy π. Naively, one might think that this leads to a set of optimal policies $\pi^*(s) \in \Pi^*(s)$ that depends on the current state s. However, one can show (and we will revisit this) that there is at least one π^* that does *not* depend on s. So, one can think of this optimal policy as being in the intersection of optimal policies for each state, $\pi^* \in \cap_{s\in\mathcal{S}}\Pi^*(s)$.

Alternatively—and this is the usual definition—one can define an ordering on policies π as follows: $\pi \geqslant \pi'$ if $V_\pi(s) \geqslant V_{\pi'}(s)$ for all $s \in \mathcal{S}$. The optimal policy π^* is a policy such that $\pi^* \geqslant \pi$ for all policies π. Again, with MDPs, the two definitions are one and the same. At least one optimal deterministic policy exists, and your plan of what to do has nothing to do with what state you start in, i.e., the initial conditions.

We find that a policy is optimal if and only if it satisfies the Bellman optimality equation, and there exists a deterministic optimal policy (see the appendices). However, we have used the Bellman equation of Sec. III A to find value functions of deterministic policies in simple time-delayed MDPs. We find generically that there is no deterministic policy that maximizes the value function of all sequences of states. Hence, there is no optimal policy, generically. An example of this is shown in Sec. III C.

Why is this? Imagine that we find $\Pi^*(s_t, \ldots, s_{t+\Delta t})$. In order for there to be an optimal policy that respects the constraints of the problem, we need for $\pi^* \in \Pi^*(s_t, \ldots)$ to depend only on s_t and not on any of the other states. This is not a mathematical condition—it is a physical one. One cannot choose one's action at time t based on future state information; this is simply not allowed in our setup, as it would be acausal. And this physical condition leads to a stringent mathematical condition that is, it seems, unlikely to be satisfied. But note that in the case of no time delay, $\Delta t = 0$, there is no issue. We can choose one π_t for each state s without restriction.

The same logic holds for the alternative Bellman equation formulation discussed earlier. There, the issue is that the optimal policy for a particular value function depends on actions taken at other times, which depend on the environmental states presented that you have no advance knowledge of.

So what should one do when faced with a time-delayed MDP? It seems to us that, although we needed to define value functions on sequences of states in order to ease computation of said value functions, policy improvement theorems are better defined on averages of said value functions so that only current and not future information can be used to choose the next action. In other words, for policy improvement purposes, we really want

$$\hat{V}_\pi(s) = \mathbb{E}[r(s_{t+\Delta t}, a_{t+\Delta t}) + \cdots | S_t = s], \tag{9}$$

$$\hat{Q}_\pi(s, a) = \mathbb{E}[r(s_{t+\Delta t}, a_{t+\Delta t}) + \cdots | S_t = s, A_{t+\Delta t} = a]. \tag{10}$$

Once we have calculated $V_\pi(s_t, \ldots, s_{t+\Delta t})$ and $Q_\pi(s_t, \ldots, s_{t+\Delta t}, a_{t+\Delta t})$, we merely need to average both over $p_\pi(s_{t+1}, \ldots, s_{t+\Delta t}|s_t = s)$, where the subscript indicates that this conditional probability depends on the action policy:

$$\hat{V}_\pi(s_t) = \sum_{s_{t+1},\ldots,s_{t+\Delta t}} p_\pi(s_{t+1}, \ldots, s_{t+\Delta t}|s_t) \times V_\pi(s_t, \ldots, s_{t+\Delta t}), \tag{11}$$

$$\hat{Q}_\pi(s_t, a_{t+\Delta t}) = \sum_{s_{t+1},\ldots,s_{t+\Delta t}} p_\pi(s_{t+1}, \ldots, s_{t+\Delta t}|s_t, a_{t+\Delta t}) \times Q_\pi(s_t, \ldots, s_{t+\Delta t}, a_{t+\Delta t}). \tag{12}$$

Suppose that we wait a bit before we start estimating the accumulated reward. The average that we use could be one of two types: an average based on initial conditions of the system that we cannot alter or an average over state sequences that depends mostly on our action policy and the environmental transition probability. If our burn-in period is long enough, the initial conditions of the system that are not under our control become unimportant. This is similarly true if γ is high enough, so that initial conditions do not take on undue importance. But the lack of an optimal policy insinuates itself into our attempts to solve the MDP by refusing to let us completely forget the system's initial conditions. Using the Bellman equations of the previous section, we find that

$$\begin{aligned}\hat{V}_\pi(s_t) = {}& \sum_{a_{t+\tau}} \pi(a_{t+\tau}|s_t) p_\pi(s_{t+\tau}|s_t) r(s_{t+\tau}, a_{t+\tau}) \\ &+ \gamma \sum_{a_{t+\tau}, s_{t+\tau+1}} \pi(a_{t+\tau}|s_t) p_\pi(s_{t+\tau}|s_t) \\ &\times p(s_{t+1}, s_{t+\tau+1}|s_{t+\tau}, a_{t+\tau}) \hat{V}_\pi(s_{t+1}),\end{aligned} \tag{13}$$

$$\begin{aligned}\hat{Q}_\pi(s_t, a_{t+\tau}) = {}& \sum_{s_{t+\tau}} p_\pi(s_{t+\tau}|s_t) r(s_{t+\tau}, a_{t+\tau}) \\ &+ \gamma \sum_{s_{t+\tau+1}} p(s_{t+\tau+1}|a_{t+\tau}, s_{t+\tau}) \\ &\times p_\pi(s_{t+1}, s_{t+\tau}|s_t) \hat{V}_\pi(s_{t+1}).\end{aligned} \tag{14}$$

Notice the dependence of the reformulated Bellman equations on $p_\pi(s_{t+1}, s_{t+\tau}|s_t)$—a transition probability that depends in a highly nonlinear way on both of the accessible quantities, $\pi(a_{t+\tau}|s_t)$ and $p(s_{t+1}|s_t, a_t)$, as an eigenvector of a complex transition matrix. Hence, one cannot use the Bellman equations for $\hat{V}_\pi$ and $\hat{Q}_\pi$ if given $\pi(a_{t+\tau}|s_t)$ and $p(s_{t+1}|s_t, a_t)$ to estimate $\hat{V}_\pi$ and $\hat{Q}_\pi$ without a great deal of extra (hidden) legwork. In this way, these equations are deceptive. It appears that the time-delayed MDP is simply a regular MDP, but the Bellman equations obtained from this viewpoint are not useful in a model-based framework. It is still necessary to operate on sequences of states if one wants to efficiently calculate value. But in a model-based framework, Eq. (14) is quite useful, as discussed in the subsection below.

Choosing $a^*(s_t) = \arg\max_{a_{t+\Delta t}} \hat{Q}_\pi(s_t, a_{t+\Delta t})$ is guaranteed to improve our averaged value function, and these improvements will be achievable, since we only need assign one action per state. We discuss algorithms for doing so in Sec. III C.

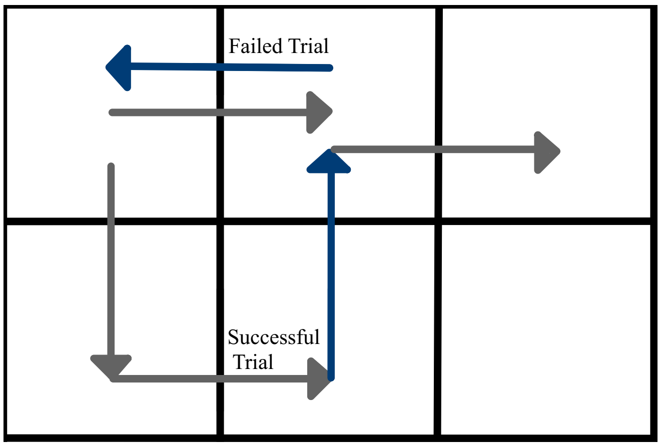

FIG. 2. Basic illustration of a failed and a successful trial (action) in the windy grid world environment. The environment is a 2×3 table of possible positions. Possible actions in this environment are up, down, left, and right. The wind pushes the agent up in the second column, and if the agent passes the upper boundary of the grid, it is sent back to the upper leftmost state.

C. An illustrative example: The effect of time delay on collected reward

We introduce the ramifications of the action time delay through a concrete example. Consider an agent that is permitted to move between spots on a given grid in a given environment. This grid therefore imposes the agent to occupy specific positions (representing the states) at specific instants of time. At each of these steps, the agent can take one of the four possible moves: up, down, left, or right. This move represents the agent's action. Furthermore, the environment contains an outside factor (the wind in the windy grid world environment) which biases the agent's move. In the example shown in Fig. 2, whenever the agent is present in the second column, the wind takes it one position up. Each time the agent takes one step, it receives a reward of -1. That is, the objective of the agent is to take the least possible steps required to reach the terminal state. The world and the example paths are shown in Fig. 2.

Our goal is to finally discuss algorithms that can compute $\hat{Q}_{\pi^*}(s_t, a_{t+\Delta t})$, and from that, we can calculate $\pi^*(a|s) = \delta_{a,a^*(s)}$ from

$$a^*(s) = \arg\max_a \hat{Q}_{\pi^*}(s, a). \tag{15}$$

We focus on the state-action-reward-state-action (SARSA) algorithm [11]. The SARSA algorithm is based on the successive evaluation of the value function $\hat{Q}_\pi$ using temporal difference learning and policy improvement. When π is improved, we first flip a biased coin such that heads comes up with the probability ϵ. We follow the policy $\pi(a|s) = \delta_{a,a^*(s)}$ when tails and choose an action uniformly at random when heads. In the tabular case, these algorithms converge under well-known conditions on ϵ and the learning rate α [12]. This is referred to as ϵ-greedy.

As this is a model-free algorithm, the fact that we cannot easily obtain $p_\pi(s_{t+\tau}|s_t)$ from the action policy $\pi(a_{t+\tau}|s_t)$ and environmental transition probabilities $p(s_{t+1}|s_t, a_t)$ is of no concern. We simply look at our experience—at what state was seen at time $t + \tau$ and at what state was seen at time t. We use this to calculate two types of error in our estimate of the state-action value function $\hat{Q}_\pi(s_t, a_{t+\tau})$.

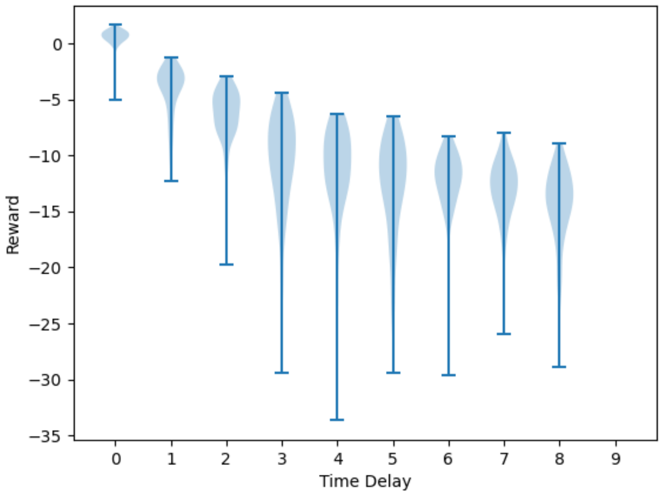

FIG. 3. Violin plots showing reward in the windy grid world for increasing values of time delay. Reward decreases as time delay (and the length of state sequence used in SARSA) increases. The algorithms that lead to reward near the top of the violin plots for various time delays are discussed in the text.

In SARSA, we measure the deviation between the current $\hat{Q}_\pi(s_t, a_{t+\tau})$ and that predicted by the corresponding Bellman equation. From Eq. (14), after some manipulation, we see that this error is

$$\delta\hat{Q}(s_t, a_{t+\tau}) = r(s_{t+\tau}, a_{t+\tau}) + \gamma\hat{Q}(s_{t+1}, a_{t+\tau+1}) - \hat{Q}(s_t, a_{t+\tau}). \tag{16}$$

Every n time steps, we choose actions based on the ϵ-greedy algorithm and update $\hat{Q}$ based on the accumulation of $\delta\hat{Q}$ in that time period. This gives us an improved estimate of the state-action value function for that policy. Then we improve π using our policy improvement theorem. This continues until convergence of the policy. It is customary to decrease ϵ as the number of epochs increases. One can show by numerical brute force that there is no optimal policy, as expected from Sec. III B.

We first show that a non-time-delayed SARSA algorithm collects less reward than a time-delayed SARSA algorithm when the agent has an action time delay. We then show that with time delay, the agent receives less reward, as it is forced to use less information to choose its actions.

While we expect to see that time delay decreases reward, we might—and will—encounter different trajectories. This is because there is stochasticity in the environment through $p(s_{t+1}|a_t, s_t)$ and some initial stochasticity in the action policy π. Hence, we must average over many trials in the figures that follow.

From two runs, we see that a SARSA algorithm that does not account for the time delay produces less reward than an algorithm that correctly accounts for the time delay. Figure 3 portrays a run example for values of the time delay from 0 to 8. The reward accounting for a time delay equal to 1 falls in the interval (1.02, 3.06) and the obtained reward for the nondelay case falls in the interval $(-17.17, -8.45)$. Note that often we do obtain rewards in an interval with positive values, which nearly never occur for the non-time-delayed case.

In Fig. 3, we see that once our algorithm is optimized for time delay, we accrue less reward with a larger time delay. When the time delay is larger, the information used to optimize actions is less relevant to the action. Hence, the reward generally decreases as the time delay increases—as the overall shape of the graph in Fig. 3 shows. In fact, in the limit that the time delay is infinite, the agent moves with no memory of the previous states and a preference for one specific nonchanging direction.

Why? Consider the case where at $t = 0$, the agent observes that it is in the top left corner, and at $t = 1$, it is still in the same state. Based off its observation one time-step earlier, the agent executes at $t = 1$ the action corresponding to its earlier state. The optimal policy for this problem and for these specific initial conditions is that the agent goes downwards, ending up in the bottom left-hand corner. Because it responds only knowing that it was previously in the *top* left-hand corner, it chooses to go down again, going out of the grid and thus ending up in the same state it was in. This step shows clearly why an agent with time delay will perform more poorly than one without delay. Although the agent is in the bottom left corner, its action at this instant is determined by its observation one time step earlier. It is not until one time step later that it takes the correct action and moves right. This means that due to its slower reaction time, the agent has wasted a time step and reduced its reward in the process. One can follow the progress of the agent using this optimal policy to see that, following this step, the agent takes all the right actions to reach the terminal state in two steps.

A larger time delay also implies larger sequences of state and bigger computational complexity. For instance, although it does not affect our results due to guarantees on convergence, the computational complexity associated with choosing the appropriate action policy increases. Moreover, agents may accrue more varying and less reward. As the time delay is increased, it gets more challenging for the agent to "choose" the best action to take at the following far step.

Note that there are model-based approaches to solving time-delayed MDPs that we are simply not considering in this paper. Model-based approaches make a model of the environmental transition probability $p(s'|a, s)$ and the reward structure $r(s, a)$ and then use the Bellman equation (or here, the revised Bellman equation) to evaluate the value of a particular sequence of states or actions. Sometimes, in model-based approaches, the law of iterated expectations is applied more than once, unrolling the expectation value. These approaches are quite flexible in that, if the environment changes in any way, one can learn new transition probabilities and reward structures and easily update the optimal action policy, but these model-based approaches lack the computational efficiency of the model-free approaches that we consider.

IV. DISCUSSION

We showed that solving time-delayed Markov Decision Processes (MDPs) in which action lags sensory perception is different than solving ordinary MDPs. First, one must operate on sequences of states or sequences of actions. Second, there is no optimal policy generically, and your initial conditions govern what the optimal policy is. Finally, time delay decreases the reward that you can get, and ignoring time delay in model-free algorithms leads to diminished reward. The literature on time-delayed MDPs has focused on using sequences of actions rather than sequences of states to solve the MDP [6–8], but we see sequences of states stored in the brain [13,14], and as such, this represents new work on time-delayed MDPs that may be applicable to biological organisms.

We hope that these advances aid those who engineer artificial cognitive systems. Previous work on time delays has discussed time-delayed MDPs in terms of the information required to turn that time-delayed MDP into an MDP; this information is called an "information state." Indeed, Ref. [8] uses an information state similar to one in the appendices while Ref. [5] uses the information state introduced in Sec. III B as an approximation for the model-free algorithms. Hence, this manuscript introduces a new information state and unifies previous descriptions.

Moreover, we hope that these results provide insight into brain function, as reinforcement learning algorithms are thought to be implemented in the brain [15,16]. It is thought that evolved organisms are reinforcement learners, and they necessarily have a time delay between visual perception and action of roughly 150 ms [17] due to the time needed for signals to propagate through the neural architecture. Hence, we would expect organisms to store sequences of states or sequences of actions, depending on which requires less memory and requires less time to process. If the size of the state space is lower than that of the action space, we expect the biological organism to operate on state sequences. Several recent articles suggest that the brain does indeed operate on state sequences [13,14], though the reasons for doing so have not yet been tied to solving time-delayed MDPs. The already observed tendency of brains to minimize cortical wiring length so as to minimize time delays [18–20] is correspondingly explainable as a way to minimize computational costs associated with reinforcement learning.

Unfortunately, we do not have an explanation as to why you would see a typical nonzero time delay in the brain such as those seen in human stick balancing. Even more reproducibility in the reward accrued or a higher probability that it is above some threshold would lead to one choosing a smaller time delay. However, time delays are known to be useful for causing recurrent neural networks with feedback to optimally predict their input [21], and they may be useful for other computations. We might, therefore, expect brain regions that are doing model-free reinforcement learning to minimize time delays, while brain regions that are doing model-based reinforcement learning, which can require explicit predictions, require a finely tuned time delay [22].

In summary, we have introduced a new set of algorithms for solving time-delayed MDPs based on sequences of states rather than sequences of actions. Given that humans have a time delay of ∼150 ms, this algorithm may have ramifications for interpreting minimization of time delays and storing of state sequences in brain regions associated with model-free reinforcement learning as trying to solve time-delayed MDPs. We hope that further experimental work [15,16] helps uncover whether or not neurotransmitter responses inherently involve the ∼150-ms time delay, which would be a signature of the brain solving a time-delayed MDP.

ACKNOWLEDGMENTS

S.M. thanks Mark Goldman for valuable conversations and Konstantinos Katsikopoulos for valuable comments. This study was supported by the Air Force Office for Scientific Research, Grant No. FA9550-19-1-0411.

APPENDIX A: DERIVATION OF THE MODIFIED BELLMAN EQUATION

We have

$$V_\pi(s_t, s_{t+1}, \dots, s_{t+\Delta t})$$
$$= \mathbb{E}[r(s_{t+\Delta t}, a_{t+\Delta t})|S_t = s_t, \dots, S_{t+\Delta t} = s_{t+\Delta t}] \quad \text{(A1)}$$
$$+ \gamma \mathbb{E}[r(s_{t+\Delta t+1}, a_{t+\Delta t+1}) + \gamma r(s_{t+\Delta t+2}, a_{t+\Delta t+2}) + \cdots |S_t = s_t, \dots, S_{t+\Delta t} = s_{t+\Delta t}, S_{t+\Delta t+1} = s_{t+\Delta t+1}] \quad \text{(A2)}$$
$$= \sum_{a_{t+\Delta t}} \pi_t(a_{t+\Delta t}|s_t) r(s_{t+\Delta t}, a_{t+\Delta t}) + \gamma \sum_{a_{t+\Delta t}, s_{t+\Delta t+1}} \pi_t(a_{t+\Delta t}|s_t) p(s_{t+\Delta t+1}|s_{t+\Delta t}, a_{t+\Delta t}) \times \mathbb{E}[r(s_{t+\Delta t+1}, a_{t+\Delta t+1}) + \gamma r(s_{t+\Delta t+2}, a_{t+\Delta t+2}) + \cdots \quad \text{(A3)}$$
$$\times |S_t = s_t, \dots, S_{t+\Delta t} = s_{t+\Delta t}, S_{t+\Delta t+1} = s_{t+\Delta t+1}] \quad \text{(A4)}$$
$$= \sum_{a_{t+\Delta t}} \pi_t(a_{t+\Delta t}|s_t) r(s_{t+\Delta t}, a_{t+\Delta t}) + \gamma \sum_{a_{t+\Delta t}, s_{t+\Delta t+1}} \pi_t(a_{t+\Delta t}|s_t) p(s_{t+\Delta t+1}|s_{t+\Delta t}, a_{t+\Delta t}) \times V_{T\pi}(s_{t+1}, \dots, s_{t+\Delta t+1}). \quad \text{(A5)}$$

APPENDIX B: THERE IS NO OPTIMAL POLICY

We now introduce a Bellman optimality equation for sequences of states which, it turns out, are not satisfied by any existing policy. This will illustrate the difficulty of finding an optimal policy in time-delayed MDPs. In deriving these equations, we imagine that we choose our action policy at time t, π_t^*, so as to maximize the value and then, following the policy $T\pi$, finding

$$V_\pi(s_t, \dots, s_{t+\Delta t}) = \max_{\pi_t} \sum_{a_{t+\Delta t}, s_{t+\Delta t}} \pi_t(a_{t+\Delta t}|s_{t+\Delta t}) + \gamma \sum_{s_{t+\Delta t+1}, a_{t+\Delta t}} \pi_t(a_{t+\Delta t}|s_{t+\Delta t}) p(s_{t+\Delta t+1}|s_{t+\Delta t}, a_{t+\Delta t}) \times V_{T\pi}(s_{t+1}, \dots, s_{t+\Delta t+1}). \quad \text{(B1)}$$

Relatedly, we can suppose that we are maximizing

$$Q_\pi(s_t, \dots, s_{t+\Delta t}, a_{t+\Delta t}) = \mathbb{E}[r(s_{t+\Delta t}, a_{t+\Delta t}) + \cdots |S_t = s_t, \dots, S_{t+\Delta t} = s_{t+\Delta t}, A_{t+\Delta t} = a_{t+\Delta t}] \quad \text{(B2)}$$
$$= r(s_{t+\Delta t}, a_{t+\Delta t}) + \gamma \sum_{s_{t+\Delta t+1}} p(s_{t+\Delta t+1}|s_{t+\Delta t}, a_{t+\Delta t}) \times V_{T\pi}(s_{t+1}, \dots, s_{t+\Delta t+1}) \quad \text{(B3)}$$

with respect to $a_{t+\Delta t}$. As maximization of Q_π achieves the maximum of V_π with respect to π_t^*, there exists a deterministic optimal policy, if an optimal policy exists.

Note already that there is a fly in the ointment. It is likely that improvements to the policy will require π_t to know something about $s_{t+1}, \dots, s_{t+\Delta t}$, which it cannot do.

Imagine choosing a policy π_t^* based on this maximization over and over again. One can show that $\pi \leqslant (\pi_t^*, T\pi) \leqslant (\pi_t^*, \pi_{t+1}^*, T^2\pi) \leqslant \cdots$ and, in this way, we have derived a policy improvement theorem: we have a way of continually improving any policy. A policy stops improving when it satisfies the Bellman optimality equation. Hence, if a policy satisfies the Bellman optimality equation, it is optimal. On the flip side, a policy that is optimal will choose π_t^* so as to maximize value and so on, and hence, a policy that is optimal satisfies the Bellman optimality equation. Together, a policy is optimal iff it satisfies the Bellman optimality equation and there exists a deterministic optimal policy.

As such, if we find that there is no deterministic policy that has maximal value on all sequences of states, we have shown that there is no optimal policy.

[1] C. Szepesvári, *Algorithms for Reinforcement Learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. 4 (Springer, Berlin, 2010), p. 1.

[2] J. Kober, J. A. Bagnell, and J. Peters, Reinforcement learning in robotics: A survey, Int. J. Rob. Res. **32**, 1238 (2013).

[3] R. S. Sutton, A. G. Barto *et al.*, Reinforcement learning, J. Cognit. Neurosci. **11**, 126 (1999).

[4] D. Lee, H. Seo, and M. W. Jung, Neural basis of reinforcement learning and decision making, Annu. Rev. Neurosci. **35**, 287 (2012).

[5] K. V. Katsikopoulos and S. E. Engelbrecht, Markov decision processes with delays and asynchronous cost collection, IEEE Trans. Autom. Control **48**, 568 (2003).

[6] D. Bertsekas, *Dynamic Programming and Optimal Control* (Athena Scientific, Nashua, NH, 2012), Vol. I.

[7] D. P. Bertsekas *et al.*, *Dynamic Programming and Optimal Control*, 3rd ed. (Athena Scientific, Nashua, NH, 2011), Vol. II.

[8] E. Altman and P. Nain, Closed-loop control with delayed information, ACM Sigmetrics Perform. Eval. Rev. **20**, 193 (1992).

[9] D. Blackwell, Discrete dynamic programming, Ann. Math. Statist. **33**, 719 (1962).

[10] J. Budd and Z. F. Kisvárday, Communication and wiring in the cortical connectome, Front. Neuroanat. **6**, 42 (2012).

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT, Cambridge, 2018).

[12] L. Behera, S. Kumar, and A. Patnaik, On adaptive learning rate that guarantees convergence in feedforward networks, IEEE Trans. Neural Networks **17**, 1116 (2006).

[13] N. F. Parker, A. Baidya, J. Cox, L. M. Haetzel, A. Zhukovskaya, M. Murugan, B. Engelhard, M. S. Goldman, and I. B. Witten, Choice-selective sequences dominate in cortical relative to thalamic inputs to NAc to support reinforcement learning, Cell Rep. **39**, 110756 (2022).

[14] N. Burgess, E. A. Maguire, and J. O'Keefe, The human hippocampus and spatial and episodic memory, Neuron **35**, 625 (2002).

[15] W. Schultz, P. Dayan, and P. R. Montague, A neural substrate of prediction and reward, Science **275**, 1593 (1997).

[16] P. R. Montague, P. Dayan, and T. J. Sejnowski, A framework for mesencephalic dopamine systems based on predictive Hebbian learning, J. Neurosci. **16**, 1936 (1996).

[17] S. Thorpe, D. Fize, and C. Marlot, Speed of processing in the human visual system, Nature (London) **381**, 520 (1996).

[18] A. A. Koulakov and D. B. Chklovskii, Orientation preference patterns in mammalian visual cortex: A wire length minimization approach, Neuron **29**, 519 (2001).

[19] D. B. Chklovskii, T. Schikorski, and C. F. Stevens, Wiring optimization in cortical circuits, Neuron **34**, 341 (2002).

[20] D. B. Chklovskii and A. A. Koulakov, Maps in the brain: What can we learn from them? Ann. Rev. Neurosci. **27**, 369 (2004).

[21] Y. Mi, C. C. A. Fung, K. Y. M. Wong, and S. Wu, Spike frequency adaptation implements anticipative tracking in continuous attractor neural networks, *Proceedings of the 27th International Conference on Neural Information Processing Systems*, Vol. 1 (MIT Press, Cambridge, 2014), pp. 505–513.

[22] A. G. E. Collins and J. Cockburn, Beyond dichotomies in reinforcement learning, Nat. Rev. Neurosci. **21**, 576 (2020).