Schedule Based Temporal Difference Algorithms

Rohan Deb*[1] Meet Gandhi*[1] Shalabh Bhatnagar[1]

2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton) | 979-8-3503-9998-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ALLERTON49937.2022.9929388

***Abstract*—Learning the value function of a given policy from data samples is an important problem in Reinforcement Learning. TD(λ) is a popular class of algorithms to solve this problem. However, the weights assigned to different n-step returns in TD(λ), controlled by the parameter λ, decrease exponentially with increasing n. In this paper, we present a λ-schedule procedure that generalizes the TD(λ) algorithm to the case when the parameter λ could vary with time-step. This allows flexibility in weight assignment, i.e., the user can specify the weights assigned to different n-step returns by choosing a sequence $\{\lambda_t\}_{t\geq 1}$. Based on this procedure, we propose an on-policy algorithm – TD(λ)-schedule, and two off-policy algorithms – GTD(λ)-schedule and TDC(λ)-schedule, respectively. We provide proofs of almost sure convergence for all three algorithms under a general Markov noise framework.**

I. Introduction

Reinforcement Learning (RL) problems can be categorised into two classes: prediction and control. The prediction problem deals with estimating the value function of a given policy as accurately as possible. Obtaining precise estimates of the value function is an important problem because value function provides useful information, such as, importance of being in different game positions in Atari games ([1]), taxi-out times at big airports ([2]), failure probability in large communication networks ([3]), etc. See [4] for a detailed discussion.

Evaluating the value function is an easy task when the state space is finite and the model of the system (transition probability matrix and single-stage reward function) is known. However, in many practical scenarios, the state space is large and the transition probability kernel is not available. Instead, samples in the form of (state, action, reward, next-state) are available and the value function needs to be estimated from these samples. In the RL community, learning using the samples generated from the actual or simulated interaction with the environment is called model-free learning.

Monte-Carlo (MC) and one-step Temporal Difference (TD) methods are popular algorithms for estimating the value function in model-free settings. The n-step TD is a generalization of one-step TD, wherein the TD error is obtained as the difference between the current estimate of the value function and the n-step return, instead of the one-step return. These n-step methods span a spectrum with MC at one end and one-step TD at the other. The TD(λ) algorithm takes the next logical step of combining the n-step returns for different values of n. A single parameter λ decides the weight assigned to different n-step returns, which decreases exponentially as n increases.

Work supported in part by the J.C.Bose Fellowship of SERB, a project under the ICPS program by DST as well as RBCCPS, IISc.

*Equal Contribution

[1]Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

A. Motivation and Contributions

In some situations, it is observed that n-step TD for some intermediate values of n outperforms the same for both small as well as large values of n (cf. Figure 7.2 of [5]). It's demonstrated empirically in [5] that neither MC nor one-step TD performs the best in terms of minimizing the Mean-Squared Error (MSE), because one-step TD is highly biased whereas MC has high variance. As a result, intermediate values of n are likely to achieve the lowest MSE. With the TD(λ) algorithm, we cannot combine only specifically chosen n step returns such as one described above, as it assigns a weight of $(1-\lambda)\lambda^n$ to the n-step return for each n. In this paper, we design a λ-schedule procedure that allows flexibility in weight assignment to the different n-step returns. Specifically, we generalize the TD(λ) algorithm to the case where the parameter λ depends on the time-step. This produces a sequence $\{\lambda_t\}_{t\geq 1}$. Using this procedure, we develop an on-policy algorithm called TD(λ)-schedule and two off-policy algorithms called GTD(λ)-schedule and TDC(λ)-schedule. We prove the convergence of all the three algorithms under a general Markov noise framework. Even though we consider the state space to be finite for ease of exposition, our proofs carry through easily to the case of general state spaces under additional assumptions. We point out here that while the TD(λ)-schedule and GTD(λ)-schedule algorithms are single-timescale algorithms with Markov noise, the TDC(λ)-schedule algorithm in fact involves two timescales with Markov noise in both the slower and faster iterates. Our proof techniques are more general as compared to others in the literature. For instance, [6] prove the convergence of TD(λ) using the results of [7]. However, to the best of our knowledge, there are no known generalizations of that result to the case of two-timescale algorithms such as TDC(λ)-schedule. Moreover, unlike [6], our results can further be extended to the case where the underlying Markov chain does not possess a unique stationary distribution but a set of such distributions that could even depend on an additional control process.

II. On-policy TD(λ)–schedule

In this section, we precisely define the on-policy TD(λ)-schedule algorithm for an infinite-horizon discounted reward Markov chain induced by the deterministic policy π. We note that though our results are applicable to Markov chains with general state space, we restrict our attention to the case where the state space is finite. Thus, the Markov chain can be defined in terms of a transition probability matrix as opposed to a transition probability kernel.

We assume that the Markov chain induced by the fixed policy π is irreducible and aperiodic, whose states lie in a discrete state space $\mathcal{S}$. We can index the state space with positive

integers, and view $\mathcal{S} = \{1, 2, \ldots, n\}$. Each state $s \in \mathcal{S}$ has a corresponding feature vector $\phi(s) \in \mathcal{R}^d$ associated with it. We denote $\{s_t | t = 0, 1, \ldots\}$ as the sequence of states visited by the Markov chain. The transition probability matrix P induced by the Markov chain has $(i,j)^{th}$ entry, denoted by p_{ij}, as probability of going from state $s_t = i$ to $s_{t+1} = j$. Also, the scalar $R_{t+1} \equiv r(s_t, s_{t+1})$ represents the single-stage reward obtained when the system transitions from state s_t to state s_{t+1}. Since the state space is finite, $\sup_t |R_{t+1}| < \infty$ almost surely. We let $\gamma \in (0,1)$ be the discount factor. The value function $V^\pi : \mathcal{S} \to \mathcal{R}$ associated with this Markov chain is given by $V^\pi(s) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | s_0 = s\Big]$. The above expectation is well-defined because the single-stage reward is bounded as mentioned above. We consider approximations of $V^\pi : \mathcal{S} \to R$ using function $V_\theta : \mathcal{S} \times \mathcal{R}^d \to \mathcal{R}$, where V_θ is a linear function approximator parameterized by θ, i.e., $V_\theta(s) = \theta^T \phi(s)$. Our aim is to find the parameter $\theta \in \mathcal{R}^d$ to minimise the *Mean Squared Error (MSE)* between the true value function $V^\pi(\cdot)$ and the approximated value function $V_\theta(\cdot)$ for a given θ, where

$$MSE(\theta) = \sum_{s \in \mathcal{S}} d^\pi(s) \left[V^\pi(s) - V_\theta(s)\right]^2 = ||V^\pi - V_\theta||_D^2. \quad (1)$$

Here, $\{d^\pi(s)\}_{s \in \mathcal{S}}$ represents the steady-state distribution for the Markov chain and the matrix D is a diagonal matrix of dimension $n \times n$ with the entries $d^\pi(s)$ on its diagonals. Minimising MSE with respect to θ by stochastic-gradient descent gives the update equation for θ as,

$$\theta_{t+1} = \theta_t + \alpha_t [V^\pi(s_t) - V_\theta(s_t)] \phi(s_t), \quad (2)$$

where θ_t is the value of parameter θ at time t. We now motivate the main idea of the paper. We propose an algorithm where the user can assign the weights to different n-step returns to estimate $V^\pi(s_t)$. We use the discounted-aware setting as described in Section-5.8 of [5] to define the return, which is used as an estimate of $V^\pi(s_t)$.

$$\begin{aligned} G_t^{\Lambda(\cdot)}(\theta) &= (1-\gamma)[\Lambda_{11} R_{t+1}] \\ &+ \gamma(1-\gamma)[\Lambda_{21}(R_{t+1} + V_\theta(s_{t+1})) \\ &\qquad + \Lambda_{22}(R_{t+1} + R_{t+2})] \\ &+ \gamma^2(1-\gamma)[\Lambda_{31}(R_{t+1} + V_\theta(s_{t+1})) \\ &\qquad + \Lambda_{32}(R_{t+1} + R_{t+2} + V_\theta(s_{t+2})) \\ &\qquad + \Lambda_{33}(R_{t+1} + R_{t+2} + R_{t+3})] + \cdots \end{aligned} \quad (3)$$

The above equation is interpreted as follows: the episode ends in one step with probability $(1-\gamma)$, in two steps with probability $\gamma(1-\gamma)$, in three steps with probability $\gamma^2(1-\gamma)$ and so on. When the episode ends in one step, bootstrapping is not applicable and thus the flat return R_{t+1} is weighed by $\Lambda_{11} = 1$. When the episode ends in two steps, the following two choices are available: bootstrap after one step $(R_{t+1} + V_\theta(s_{t+1}))$ or use the flat return $(R_{t+1} + R_{t+2})$. We weight these two quantities by Λ_{21} and Λ_{22} respectively under the constraint that Λ_{21} and Λ_{22} are non-negative and sum to 1. Similarly, when the episode ends in 3 steps, we have three choices: bootstrap after one step, bootstrap after two steps or take the flat return. We weight these three quantities by Λ_{31}, Λ_{32} and Λ_{33} respectively, under the constraint that these three weights are non-negative and sum to one, and so on. These weights can be summarized in a matrix as below, where each $\Lambda_{ij} \in [0,1]$ and weights in each row sum to one.

$$\Lambda = \begin{bmatrix} 1 & 0 & 0 & 0 \cdots \\ \Lambda_{21} & \Lambda_{22} & 0 & 0 \cdots \\ \Lambda_{31} & \Lambda_{32} & \Lambda_{33} & 0 \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

To obtain an online equation for $G_t^{\Lambda(\cdot)}(\theta) - V_\theta(s_t)$, we add and subtract $V_\theta(s_{t+1})$ to all the terms starting from $R_{t+1} + R_{t+2}$ in (3). We notice that, on RHS, the coefficient of R_{t+1} is 1. Similarly, the coefficient of $V_\theta(s_{t+1})$ is γ (See Appendix A1 of [8] for details). Hence,

$$\begin{aligned} G_t^{\Lambda(\cdot)}(\theta) - V_\theta(s_t) &= R_{t+1} + \gamma V_\theta(s_{t+1}) - V_\theta(s_t) \\ &+ \gamma\Lambda_{22}\{(1-\gamma)(R_{t+2}) \\ &\quad + \gamma(1-\gamma)\Big[\frac{\Lambda_{32}}{\Lambda_{22}}(R_{t+2} + V_\theta(s_{t+2})) \\ &\quad + \frac{\Lambda_{33}}{\Lambda_{22}}(R_{t+2} + R_{t+3})\Big] \\ &\quad + \cdots - V_\theta(s_{t+1})\}. \end{aligned}$$

To write the above equation recursively, we notice that we need to have the additional constraint $\Lambda_{32} + \Lambda_{33} = \Lambda_{22}$. In general we must ensure that $\Lambda_{j,j-1} + \Lambda_{j,j} = \Lambda_{j-1,j-1}\ \forall j \geq 2$. Setting $\Lambda_{22} = \lambda_1$ and using the above constraint, we obtain $\Lambda_{21} = (1-\lambda_1)$. Further, setting $\Lambda_{33} = \lambda_1\lambda_2$, $\Lambda_{32} = \lambda_1(1-\lambda_2)$ (to ensure that $\Lambda_{32} + \Lambda_{33} = \Lambda_{22}$), we obtain $\Lambda_{31} = (1-\lambda_1)$. We refer to the user-specified sequence $\lambda_j, j \in \mathbb{N}$ as the λ-*schedule* hereafter. The weight matrix Λ can be constructed from the user specified λ-schedule as below:

$$\Lambda = \begin{bmatrix} 1 & 0 & 0 & 0 \ldots \\ (1-\lambda_1) & \lambda_1 & 0 & 0 \ldots \\ (1-\lambda_1) & \lambda_1(1-\lambda_2) & \lambda_1\lambda_2 & 0 \ldots \\ (1-\lambda_1) & \lambda_1(1-\lambda_2) & \lambda_1\lambda_2(1-\lambda_3) & \ddots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}.$$

Thus,

$$\begin{aligned} G_t^{[\lambda_1:]}(\theta) - V_\theta(s_t) &= R_{t+1} + \gamma V_\theta(s_{t+1}) - V_\theta(s_t) \\ &+ \gamma\lambda_1\{(1-\gamma)(R_{t+2}) + \gamma(1-\gamma) \\ &[(1-\lambda_2)(R_{t+2} + V_\theta(s_{t+2})) + \lambda_2(R_{t+2} + R_{t+3})] \\ &+ \ldots - V_\theta(s_{t+1})\} \\ &= \delta_t + \gamma\lambda_1[G_{t+1}^{[\lambda_2:]} - V_\theta(s_{t+1})] \\ &= \delta_t + \gamma\lambda_1\delta_{t+1} + \gamma^2\lambda_1\lambda_2\delta_{t+2} + \cdots \end{aligned}$$

where, δ_t is the TD-error defined as $\delta_t = R_{t+1} + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)$. The superscript $[\lambda_1 :]$ in the return defined above denotes that the two-step flat return is weighed by λ_1, the three-step flat return is weighted by $\lambda_1\lambda_2$, etc., whereas the

superscript $[\lambda_2 :]$ denotes that the two-step flat return is weighted by λ_2, the three-step flat return by $\lambda_2\lambda_3$, etc. Then, for $t = 0, 1, \ldots$, the **TD(λ)-schedule** algorithm updates θ as follows:

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t z_t, \text{ where } z_t = \sum_{k=0}^{t} \Big(\prod_{j=1}^{t-k} \gamma \lambda_j \Big) \phi(s_k). \quad (4)$$

Here $\{\alpha_t\}_{t\geq 0}$ is the sequence of step-size parameters and θ_0 is the initial parameter vector. We make some key observations here: **(1)** If $\lambda_j = 1$ for $j \leq n$ and $\lambda_j = 0$ for $j > n$, for some $n > 0$, then we obtain the n-step TD algorithm. **(2)** If $\lambda_j = 1$ $\forall j \in \mathbb{N}$, then we obtain the MC algorithm. **(3)** If $\lambda_j = \lambda$ $\forall j \in \mathbb{N}$, then we obtain the TD(λ) algorithm.

For the remaining part of the paper, we only consider λ-schedules where $\exists L > 0$ such that $\lambda_j = 0$ for all $j > L$. We denote the return associated with such a schedule by $G_t^{[\lambda_1:\lambda_L]}(\theta)$. The equation of z_t then reduces to

$$z_t = \sum_{k=t-L}^{t} \left(\prod_{j=1}^{t-k} \gamma \lambda_j \right) \phi(s_k). \quad (5)$$

We point out that z_t can't be written recursively in terms of z_{t-1} unlike TD(λ) and therefore using schedules of the form as described above becomes essential to avoid explosion of space. Note that we need to store the last L states to compute the eligibility trace z_t. The algorithm for TD(λ)-schedule is given below:

Algorithm 1 TD(λ)-schedule

Input: Policy π, step-size sequence $\{\alpha_t\}_{t\geq 1}$, lambda-schedule $\{\lambda_t\}_{t=1}^{L}$ and the feature map $\phi(\cdot)$.
Initialize θ_0 and s_0 randomly, $z \leftarrow 0$.
for t=1,2,3,... **do**
  Choose an action $a_t \sim \pi(\cdot|s_{t-1})$
  Perform action a_t and observe reward R_{t+1} and next state s_{t+1}.
  Compute the eligibility trace as in (5).
  $\delta_t \leftarrow R_t + \gamma \phi(s_{t+1})^T \theta - \phi(s_t)^T \theta$
  $\theta_t \leftarrow \theta_t + \alpha_t \delta_t z_t$
end for

A. EqualWeights schedule

As already mentioned, it has been seen empirically that n-step TD performs better for some intermediate values of n. If for a particular problem, n-step TD achieves low MSE for some $n_1 \leq n \leq n_2$ (cf. Figure 7.2 of [5]), then it makes sense to combine only these n-step returns instead of all the n-step returns with exponentially decreasing weights. The TD(λ)-schedule algorithm lets us do this. Suppose, we want to assign equal weights to all n-step returns for $n_1 \leq n \leq n_2$. We can achieve this by selecting a λ-schedule as follows:

$$\lambda_i = \begin{cases} 1 & \text{if } i < n_1 \\ 1 - \frac{1}{n_2 - i + 1} & \text{if } n_1 \leq i \leq n_2 \\ 0 & \text{if } i > n_2. \end{cases}$$

We call this schedule *EqualWeights*(n_1, n_2). To take an example consider *EqualWeights*$(3, 5)$. The λ-schedule is given by $\lambda_1 = 1, \quad \lambda_2 = 1, \quad \lambda_3 = \frac{2}{3}, \quad \lambda_4 = \frac{1}{2}, \quad \lambda_j = 0, \quad \forall j \geq 5$ and the associated weight matrix is:

$$\Lambda = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0\ldots \\ 0 & 1 & 0 & 0 & 0 & 0 & 0\ldots \\ 0 & 0 & 1 & 0 & 0 & 0 & 0\ldots \\ 0 & 0 & 1/3 & 2/3 & 0 & 0 & 0\ldots \\ 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0\ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{bmatrix}.$$

We notice that when the episode length ≥ 5, the above matrix assigns equal weights to 3-step, 4-step and 5-step TD returns. When the episode length ≤ 3, it takes the Monte Carlo return. Such an arbitrary weight assignment to n-step returns is not possible with TD(λ). Appendix A5 of [8] reports evaluation with the EqualWeights schedule on some standard MDPs.

III. Off Policy Gradient λ–schedule algorithms

While TD(λ)-schedule is an on-policy algorithm, we now present a couple of off-policy algorithms that are based on the λ-schedule procedure. We first describe the off-policy setting briefly. The agent selects actions according to a *behaviour policy* $\mu : \mathcal{S} \times \mathcal{A} \to [0, 1]$, while we are interested in computing the value function V^π associated with the *target policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. Let $d^\mu(s)$, $s \in \mathcal{S}$ denote the steady-state probabilities for the Markov chain under the behaviour policy μ and let the importance sampling ratio $\rho_t = \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}$, where a_t is the action picked at time-step t. Along the lines of per-decision importance sampling (Section 5.9, [5]), we obtain the off-policy λ-schedule return as

$$G_t^{[\lambda_1:\lambda_L]}(\theta) - V_\theta(s_t) = \rho_t \delta_t + \gamma \lambda_1 \rho_t \rho_{t+1} \delta_{t+1} + \cdots + \gamma^L \lambda_1 \ldots \lambda_L \rho_t \ldots \rho_{t+L} \delta_{t+L}.$$

We now obtain the *Off-Policy TD(λ)-schedule* algorithm by defining the eligibility vector z_t and the update equation for θ as below:

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t z_t, \text{ with } z_t = \sum_{k=t-L}^{t} \rho_t \Big(\prod_{j=1}^{t-k} \rho_{t-j} \gamma \lambda_j \Big) \phi(s_k). \quad (6)$$

We observe that the above algorithm diverges on Baird's Counterexample ([9]) (See Appendix A5 of [8]). Gradient based algorithms [10] are observed to converge in the off-policy setting. Inspired by this, we develop two gradient-based schedule algorithms, *GTD(λ)-Schedule* and *TDC(λ)-Schedule*, as described below.

We note that the λ-schedule return defined in (3) can also be written as below (See Appendix A2 of [8]):

$$G_t^{[\lambda_1:\lambda_L]}(\theta) = R_{t+1} + \gamma \left[(1 - \lambda_1) V_\theta(s_{t+1}) + \lambda_1 G_{t+1}^{[\lambda_2:\lambda_L]}(\theta) \right]. \quad (7)$$

Next we define the value function associated with state s as

$$V^\pi(s) = \mathbb{E}\left[G_t^{[\lambda_1:\lambda_L]}(\theta) | s_t = s, \pi \right] \triangleq \left(T^{\pi,[\lambda_1:\lambda_L]} V^\pi \right)(s). \quad (8)$$

Here, $T^{\pi,[\lambda_1:\lambda_L]}$ denotes the λ-schedule Bellman operator. The objective function $J(\theta)$ on which gradient descent is performed is the *Mean Squared Projected Bellman Error* defined as follows:

$$\begin{aligned} J(\theta) &\triangleq ||V_\theta - \Pi T^{\pi,[\lambda_1:\lambda_L]} V_\theta||^2_{D_\mu} \\ &= \sum_s d^\mu(s) \Big(V_\theta(s) - \Pi T^{\pi,[\lambda_1:\lambda_L]} V_\theta(s)\Big)^2, \end{aligned} \tag{9}$$

where $d^\mu(s)$ is the visitation probability to state s under the steady-state distribution when the behaviour policy μ is followed and D_μ is an $n \times n$ diagonal matrix with $d^\mu(s)$ as its s^{th} diagonal entry. We also define

$$\delta_t^{[\lambda_1:\lambda_L]}(\theta) \triangleq G_t^{[\lambda_1:\lambda_L]}(\theta) - \theta^T \phi_t, \tag{10}$$

$$P_\mu^\pi \delta_t^{[\lambda_1:\lambda_L]}(\theta)\phi_t \triangleq \sum_s d^\mu(s) \mathbb{E}_\pi \left[\delta_t^{[\lambda_1:\lambda_L]}(\theta)\phi_t | S_t = s, \pi\right]. \tag{11}$$

As with [10], using these definitions, the *Projected Bellman Error* is expressed as the product of three expectations in the following lemma. The proofs of the results below are provided in Appendix A3 of [8].

Lemma III.1. *The objective function* $J(\theta) = ||V_\theta - \Pi T^{\pi,[\lambda_1:\lambda_L]} V_\theta||^2_D$ *can be equivalently written as* $J(\theta) = \left(P_\mu^\pi \delta_t^{[\lambda_1:\lambda_L]}(\theta)\phi_t\right)^T \mathbb{E}_\mu \left[\phi_t \phi_t^T\right]^{-1} \left(P_\mu^\pi \delta_t^{[\lambda_1:\lambda_L]}(\theta)\phi_t\right)$.

The above lemma gives an expression for the objective function. However, the expectation is with respect to the target policy π, but needs to be computed from samples of the trajectory generated by the behaviour policy μ. Secondly, the above is a forward-view equation and needs to be converted to an equivalent backward view. Theorem III.2 converts the expectation with respect to π to an expectation with respect to μ. In order to do so, as in [10], we define the following terms:

$$\begin{aligned} G_t^{[\lambda_1:\lambda_L],\rho}(\theta) = \rho_t \Big(R_{t+1} + \gamma \Big((1-\lambda_1) V_\theta(s_{t+1}) \\ + \lambda_1 G_{t+1}^{[\lambda_2:\lambda_L],\rho}(\theta)\Big)\Big), \end{aligned} \tag{12}$$

$$\delta_t^{[\lambda_1:\lambda_L],\rho}(\theta) \triangleq G_t^{[\lambda_1:\lambda_L],\rho}(\theta) - \theta^T \phi_t.$$

Theorem III.2. $P_\mu^\pi \delta_t^{[\lambda_1:\lambda_L]}(\theta)\phi_t = \mathbb{E}_\mu \left[\delta_t^{[\lambda_1:\lambda_L],\rho}(\theta)\phi_t\right]$.

Theorem III.4 converts the forward view into an equivalent backward view using the lemma below.

Lemma III.3. $\mathbb{E}_\mu \left[\rho_t \gamma \lambda_1 \delta_{t+1}^{[\lambda_2:\lambda_L],\rho}(\theta)\phi_t\right] = \mathbb{E}_\mu \left[\rho_{t-1} \gamma \lambda_1 \delta_t^{[\lambda_2:\lambda_L],\rho}(\theta)\phi_{t-1}\right]$.

Theorem III.4. *Define the eligibility trace vector* $z_t = \sum_{i=t-L}^{t} \left[\rho_t \left(\Pi_{j=1}^{t-i} \rho_{t-j} \gamma \lambda_j\right) \phi_i\right]$. *Then,* $\mathbb{E}_\mu \left[\delta_t^{[\lambda_1:\lambda_L],\rho}(\theta)\phi_t\right] = \mathbb{E}_\mu \left[\delta_t(\theta) z_t\right]$.

Using the above results, we can express the objective function as:

$$J(\theta) = \mathbb{E}_\mu \left[\delta_t(\theta) z_t\right]^T \mathbb{E}\left[\phi_t \phi_t^T\right]^{-1} \mathbb{E}\left[\delta_t(\theta) z_t\right], \text{ hence,}$$

$$\begin{aligned} -\frac{1}{2}\nabla J(\theta) &= -\nabla \mathbb{E}\left[\delta_t(\theta) z_t^T\right] \mathbb{E}\left[\phi_t \phi_t^T\right]^{-1} \mathbb{E}\left[\delta_t(\theta) z_t\right] \\ &= \mathbb{E}\left[(\phi_t - \gamma \phi_{t+1}) z_t^T\right] \mathbb{E}\left[\phi_t \phi_t^T\right]^{-1} \mathbb{E}\left[\delta_t(\theta) z_t\right]. \end{aligned}$$

We keep a stationary average for the second and third expectations in a parameter vector w and sample the terms in the first expectation. We call the resultant algorithm **GTD(λ)-schedule** whose iterates are as given below:

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha_t \left((\phi_t - \gamma \phi_{t+1}) z_t^T w_t\right), \\ w_{t+1} &= w_t + \beta_t \left(\delta_t(\theta_t) z_t - \phi_t \phi_t^T w_t\right). \end{aligned} \tag{13}$$

Next, as with [10], an alternative is to express the gradient direction as:

$$\begin{aligned} -\frac{1}{2}\nabla J(\theta) &= \left(\mathbb{E}\left[\phi_t \phi_t^T\right] + \mathbb{E}\left[(\phi_t - \gamma \phi_{t+1}) z_t^T - \phi_t \phi_t^T\right]\right) \\ &\quad \mathbb{E}\left[\phi_t \phi_t^T\right]^{-1} \mathbb{E}\left[\delta_t(\theta) z_t\right] \\ &= \mathbb{E}\left[\delta_t(\theta) z_t\right] - \left(\mathbb{E}\left[(\gamma \phi_{t+1} - \phi_t) z_t^T + \phi_t \phi_t^T\right]\right) \\ &\quad \left(\mathbb{E}\left[\phi_t \phi_t^T\right]^{-1} \mathbb{E}\left[\delta_t(\theta) z_t\right]\right). \end{aligned}$$

As before, we maintain a stationary estimate for the last two terms and sample the remaining terms to obtain the iterates for **TDC(λ)-schedule**:

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha_t \left(\delta_t(\theta_t) z_t\right) \\ &\quad - \alpha_t \left((\gamma \phi_{t+1} - \phi_t) z_t^T w_t + \phi_t \phi_t^T w_t\right), \\ w_{t+1} &= w_t + \beta_t \left(\delta_t(\theta_t) z_t - \phi_t \phi_t^T w_t\right). \end{aligned} \tag{14}$$

Appendix A5 of [8] compares GTD(λ)-schedule and TDC(λ)-schedule with GTD and TDC.

IV. Convergence Analysis

Our proof technique differs significantly from other references in the asymptotic analysis of our algorithm. In particular, we follow the ordinary differential equation (ODE) based analysis under Markov noise for single and multiple timescale algorithms (cf. [11], [12], [13] and [14]). We begin with the convergence analysis of the TD(λ)-Schedule algorithm. Starting from some initial state s_0, we generate a single infinitely long trajectory $(s_0, s_1, \ldots)$. Suppose at time t, value of the parameter θ is θ_t. We consider a linear parameterisation of the value function as $V_\theta(s_t) = \phi_t^T \theta_t$, where $\phi_t \equiv \phi(s_t)$. After the transition from state s_t to s_{t+1}, we evaluate the temporal difference term and update the parameter θ_t according to (4), assuming the product $\prod_{j=n+1}^n = 1$, $\forall n$.

As mentioned above, we only consider λ schedules where $\exists L > 0$ such that $\lambda_j = 0$ for all $j > L$. With such a choice of schedule, we need to store only the last L states. We make the following assumptions:

Assumption 1. *The step-sizes* α_t *are positive and satisfy* $\sum_t \alpha_t = \infty$ *and* $\sum_t \alpha_t^2 < \infty$.

Assumption 2. *There exists a distribution* $d^\pi(j), j \in S$ *such that* $\lim_{t\to\infty} P(s_t = j | s_0 = i) = d^\pi(j)\ \forall i, j$.

Assumption 3. *The matrix* Φ *has full rank, where* Φ *is an* $n \times d$ *matrix where the* s^{th} *row is* $\phi(s)^T$.

Let $X_t = (s_{t-L}, s_{t-L+1}, \ldots, s_t, s_{t+1})$. Clearly X_t is a Markov chain because s_{t+2} only depends on s_{t+1}. Note that z_t is not included in the Markov chain as it can be constructed from X_t. The steady state version of the Markov chain can be constructed from s_t, $-\infty < t < \infty$, whose transition probabilities are given by P. We then let

$$z_t = \sum_{\tau=-\infty}^{t} \left(\prod_{j=1}^{t-\tau} \gamma \lambda_j \phi(s_\tau) \right) = \sum_{\tau=t-L}^{t} \left(\prod_{j=1}^{t-\tau} \gamma \lambda_j \phi(s_\tau) \right).$$

We use $\mathbb{E}_0[\cdot]$ to denote the expectation with respect to the steady-state distribution of X_t. Now, we can write $\delta_t z_t$ as: $\delta_t z_t = A(X_t)\theta_t + b(X_t)$, where, $b(X_t) = z_t R_{t+1}$ and $A(X_t) = z_t(\gamma \phi_{t+1}^T - \phi_t^T)$. Let D be the diagonal matrix with $d^\pi(s)$, $s \in S$ as it's diagonal elements. Further, let $A = \mathbb{E}_0[A(X_t)]$ and $b = \mathbb{E}_0[b(X_t)]$.

Proposition IV.1. *The matrix A is negative definite.*

Proof. See Appendix A4 of [8]. □

A. Convergence of TD(λ)–schedule

We now present a result from Chapter 6 of [11] (see also [12]) that gives the stability and convergence of a stochastic approximation recursion under Markov noise. Let $\check{S}$ denote the set in which the process $\{X_t\}$ takes values in.

Theorem IV.2. *Consider the following recursion in $\mathbb{R}^d$:*

$$\theta_{t+1} = \theta_t + \alpha_t \left(h(\theta_t, X_t) + M_{t+1} \right). \tag{15}$$

Consider now a sequence $\{t(n)\}$ of time points defined as follows: $t(0) = 0$, $t(n) = \sum_{k=0}^{n-1} \alpha_k$, $n \geq 1$. Now define the algorithm's trajectory $\bar{\theta}(t)$ according to: $\bar{\theta}(t(n)) = \theta_n$, $\forall n$, and with $\bar{\theta}(t)$ defined as a continuous linear interpolation on all intervals $[t(n), t(n+1)]$. Finally, consider the following assumptions:

- **(B1)** *$h : \mathbb{R}^d \times \check{S} \to \mathbb{R}^d$ is Lipschitz continuous in the first argument, uniformly with respect to the second.*
- **(B2)** *For any given $\theta \in \mathbb{R}^d$, the set $D(\theta)$ of ergodic occupation measures of $\{X_n\}$ is compact and convex.*
- **(B3)** *$\{M_t\}_{t \geq 1}$ is a square-integrable martingale difference sequence. Further, $\mathbb{E}\left[||M_{t+1}||^2 | \mathcal{F}_t\right] \leq K(1 + ||\theta_t||^2)$, where $\mathcal{F}_t = \sigma(\theta_m, X_m, M_m, m \leq t)$, $t \geq 0$.*
- **(B4)** *The step-size sequence $\{\alpha_t\}_{t \geq 0}$ satisfies $\alpha_t > 0, \forall t$. Further, $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$.*
- **(B5)** *Let $\tilde{h}(\theta, \nu) = \int h(\theta, x) \nu(dx)$. Also, $h_c(\theta, \nu) = \dfrac{\tilde{h}(c\theta, \nu(c\theta))}{c}$.*
  - (i) *The limit $\tilde{h}_\infty(\theta, \nu) = \lim_{c \to \infty} \tilde{h}_c(\theta, \nu)$ exists uniformly on compacts.*
  - (ii) *There exists an attracting set $\mathcal{A}$ associated with the differential inclusion (DI) $\dot{\theta}(t) \in H(\theta(t))$ where $H(\theta) = \bar{co}(\{\tilde{h}_\infty(\theta, \nu) : \nu \in D(\theta)\})$ such that $\sup_{u \in \mathcal{A}} ||u|| < 1$ and $\bar{B}_1(0) \stackrel{\triangle}{=} \{x \mid ||x|| \leq 1\}$ is a fundamental neighborhood of $\mathcal{A}$.*

Under (B1)-(B5), $\{\bar{\theta}(s + \cdot), s \geq 0\}$ converges to an internally chain transitive invariant set of the differential inclusion $\dot{\theta}(t) \in \hat{h}(\theta(t))$, where $\hat{h}(\theta) = \{\tilde{h}(\theta, \nu) \mid \nu \in D(\theta)\}$. In particular, $\{\theta_t\}$ converges almost surely to such a set.

We now present our main result on the TD(λ)-schedule algorithm.

Theorem IV.3. *Under Assumptions 1–3, the TD(λ)-schedule algorithm given by* (4) *satisfies $\theta_t \to \theta^* \stackrel{\triangle}{=} -A^{-1}b$ almost surely as $t \to \infty$.*

Proof. See proof of Theorem 4.3 in [8]. □

B. Convergence of GTD(λ)–schedule and TDC(λ)–schedule

We make the following assumption for the convergence analysis of GTD(λ)-schedule.

Assumption 4. *The step-size sequence β_t satisfies $\beta_t > 0$, $\forall t$, $\sum_t \beta_t = \infty$ and $\sum_t \beta_t^2 < \infty$. Further, we assume $\frac{\beta_k}{\alpha_k} = \eta \ \ \forall k \geq 0$.*

Theorem IV.4. *Under Assumptions 1-4, $\{\theta_t\}$ in the GTD(λ)-Schedule iterate given in equation* (13) *converges almost surely to $-A^{-1}b$.*

Proof. See Appendix A4 in [8]. □

We now make the following assumption for the convergence analysis of TDC(λ)-schedule.

Assumption 5. *The step-size sequence β_t satisfies $\beta_t > 0$, $\forall t$, $\sum_t \beta_t = \infty$ and $\sum_t \beta_t^2 < \infty$. Further, we assume, $\frac{\alpha_k}{\beta_k} \to 0$ as $k \to \infty$.*

The TDC(λ)-schedule update rule can be rewritten in the form:

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha_t h(\theta_t, w_t, X_t), \\ w_{t+1} &= w_t + \beta_t g(\theta_t, w_t, X_t), \end{aligned} \tag{16}$$

where $h : \mathbb{R}^d \times \mathbb{R}^d \times \check{S} \to \mathbb{R}^d$ and $g : \mathbb{R}^d \times \mathbb{R}^d \times \check{S} \to \mathbb{R}^d$ are defined as $h(\theta_t, w_t, X_t) = A(X_t)\theta_t + b(X_t) - A(X_t)^T w_t - C(X_t)w_t$ and $g(\theta_t, w_t, X_t) = A(X_t)\theta_t + b(X_t) - C(X_t)w_t$, respectively. Our analysis here is based on stability and convergence results of two-timescale stochastic approximation from [13] and [14].

Define functions $\bar{h}, \bar{g} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ according to

$$\bar{h}(\theta, w) = \int h(\theta, w, X) \nu(dX) = A\theta + b - A^T w - Cw,$$

$$\bar{g}(\theta, w) = \int g(\theta, w, X) \nu(dX) = A\theta + b - Cw,$$

respectively, with A, b, C as before. We shall first present below the main result for which we need the following assumptions:

- **(C1)** The functions $h(\theta, w, X)$ and $g(\theta, w, X)$ are Lipschitz continuous in (θ, w) for given $X \in \check{S}$.
- **(C2)** $\{\alpha_t\}$ and $\{\beta_t\}$ are step-size schedules that satisfy: $\alpha_t, \beta_t > 0$, $\forall t$, $\sum_t \alpha_t = \sum_t \beta_t = \infty$, $\sum_t (\alpha_t^2 + \beta_t^2) < \infty$, $\lim_{t \to \infty} \dfrac{\alpha_t}{\beta_t} = 0$.

(C3) The sequence of functions $\bar{g}_c(\theta, w) \stackrel{\triangle}{=} \frac{\bar{g}(c\theta, cw)}{c}$, $c \geq 1$, satisfy $\bar{g}_c \to \bar{g}_\infty$ uniformly on compacts for some $\bar{g}_\infty : \mathbb{R}^d \to \mathbb{R}^d$. Also, the limiting ODE $\dot{w}(t) = \bar{g}_\infty(\theta, w(t))$, i.e., with $\theta(t) \equiv \theta$, has a unique globally asymptotically stable equilibrium $\lambda_\infty(\theta)$ where $\lambda_\infty : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz continuous. Further, $\lambda_\infty(0) = 0$, i.e., $\dot{w}(t) = \bar{g}_\infty(0, w(t))$ has the origin in $\mathbb{R}^d$ as it's unique globally asymptotically stable equilibrium.

(C4) The functions $\bar{h}_c(\theta) \stackrel{\triangle}{=} \frac{\bar{h}(c\theta, c\lambda_\infty(\theta))}{c}$, $c \geq 1$ satisfy $\bar{h}_c \to \bar{h}_\infty$ as $c \to \infty$ uniformly on compacts for some $\bar{h}_\infty : \mathbb{R}^d \to \mathbb{R}^d$. Also, the limiting ODE $\dot{\theta}(t) = \bar{h}_\infty(\theta(t))$ has the origin in $\mathbb{R}^d$ as it's unique globally asymptotically stable equilibrium.

(C5) The ODE $\dot{w}(t) = \bar{g}(\theta, w(t))$ has a globally asymptotically stable equilibrium $\lambda(\theta)$ (uniformly in θ), where $\lambda : \mathbb{R}^d \to \mathbb{R}^d$ is Lispchitz continuous.

(C6) The ODE $\dot{\theta}(t) = \bar{h}(\theta(t), \lambda(\theta(t)))$ has a globally asymptotically stable equilibrium θ^*.

We now state the key result.

Theorem IV.5. *Under (C1)-(C6), the recursions* (16) *satisfy: (a)* $\sup_n(||\theta_n|| + ||w_n||) < \infty$ *and (b)* $(\theta_n, w_n) \to (\theta^*, \lambda(\theta^*))$ *almost surely.*

We now have our main result for TDC(λ)-schedule.

Theorem IV.6. *Under Assumptions 1-3 and 5, the TDC(λ)-schedule algorithm given by* (14) *satisfies* $\theta_t \to \theta^* = -A^{-1}b$ *almost surely as* $t \to \infty$.

Proof. See proof of Theorem 4.6 in [8]. □

V. Related Work and Conclusion

Recent work by [15], [5] and an earlier work by [4] provide a comprehensive survey of TD based algorithms. However, for the sake of completeness we discuss some of the relevant works. TD(λ) with variable λ presented in Chapter 12 of [5] and [16] come close to our algorithms. However, the parameter λ in those algorithms is a function of state. Moreover, such a λ-function does not give arbitrary weights to different n-step returns. In fact, to the best of our knowledge, no other variant of TD has looked into letting the user assign weights to different n-step returns. However, our λ-schedule procedure allows this by choosing appropriate λ schedules. State-dependent λ can be derived as a special case of our λ-schedule procedure by letting $\lambda_1 = \lambda(s_t)$, $\lambda_2 = \lambda(s_{t+1})$, etc.

The convergence proofs presented in our paper differ from the proofs presented earlier in the TD(λ)-literature and require much less verification since they are based on the ODE method. Our two-timescale proof is novel in that such a proof under the Markov noise setting has not been presented before. Providing the proof of two-time scale iterates under Markov noise as here has been mentioned in [10] as future work. We also mention that our proofs are presented under fairly general conditions and could be generalized further for a more general state-valued process. These proofs also work for the case where the underlying Markov process does not possess a unique stationary distribution that can in turn also depend on the underlying parameters. See remarks in Appendix A6 of [8] for some further discussions on the proof techniques.

Our work calls in for a comparative analysis of bias variance trade-off of all these variants of TD algorithm. Devising and comparing different λ-schedules is left for future work. Another possible direction would be to extend the schedule-based algorithms to the control setting, for instance, through SARSA(λ) ([5]) or actor-critic methods ([17], [18]).

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[2] P. Balakrishna, R. Ganesan, and L. Sherry, "Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of tampa bay departures," *Transportation Research Part C: Emerging Technologies*, vol. 18, pp. 950–962, 12 2010.

[3] J. Frank, S. Mannor, and D. Precup, "Reinforcement learning in the presence of rare events," in *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, ser. ACM International Conference Proceeding Series, W. W. Cohen, A. McCallum, and S. T. Roweis, Eds., vol. 307. ACM, 2008, pp. 336–343. [Online]. Available: https://doi.org/10.1145/1390156.1390199

[4] C. Dann, G. Neumann, and J. Peters, "Policy evaluation with temporal differences: A survey and comparison," *Journal of Machine Learning Research*, vol. 15, no. 24, pp. 809–883, 2014. [Online]. Available: http://jmlr.org/papers/v15/dann14a.html

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

[6] J. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 674–690, 1997.

[7] A. Benveniste, P. Priouret, and M. Métivier, *Adaptive Algorithms and Stochastic Approximations*. Berlin, Heidelberg: Springer-Verlag, 1990.

[8] R. Deb, M. Gandhi, and S. Bhatnagar, "Schedule based temporal difference algorithms," 2021. [Online]. Available: https://arxiv.org/abs/2111.11768

[9] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *In Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann, 1995, pp. 30–37.

[10] H. R. Maei, "Gradient temporal-difference learning algorithms," Ph.D. dissertation, University of Alberta, CAN, 2011, aAINR89455.

[11] V. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008. [Online]. Available: https://books.google.co.in/books?id=QLxIvgAACAAJ

[12] A. Ramaswamy and S. Bhatnagar, "Stability of stochastic approximations with "controlled markov" noise and temporal difference learning," *IEEE Transactions on Automatic Control*, vol. 64, no. 6, pp. 2614–2620, 2019.

[13] C. Lakshminarayanan and S. Bhatnagar, "A stability criterion for two-timescale stochastic approximation schemes," *Automatica*, vol. 79, pp. 108–114, 2017.

[14] P. Karmakar and S. Bhatnagar, "Two time-scale stochastic approximation with controlled markov noise and off-policy temporal-difference learning," *Mathematics of Operations Research*, vol. 43, no. 1, pp. 130–151, 2018.

[15] S. Ghiassian, A. Patterson, M. White, R. Sutton, and A. White, "Online off-policy prediction," *ArXiv*, vol. abs/1811.02597, 2018.

[16] R. Sutton, A. R. Mahmood, D. Precup, and H. Hasselt, "A new q(λ) with interim forward view and monte carlo equivalence," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32. Bejing, China: PMLR, 22–24 Jun 2014, pp. 568–576. [Online]. Available: http://proceedings.mlr.press/v32/sutton14.html

[17] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[18] S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee, "Natural Actor-Critic Algorithms," *Automatica*, vol. 45, no. 11, Jul. 2009. [Online]. Available: https://hal.inria.fr/hal-00840470