Stable and Efficient Policy Evaluation

Daoming Lyu, Bo Liu, *Member, IEEE*, Matthieu Geist, Wen Dong, Saad Biaz, *Member, IEEE*, and Qi Wang, *Senior Member, IEEE*

***Abstract*— Policy evaluation algorithms are essential to reinforcement learning due to their ability to predict the performance of a policy. However, there are two long-standing issues lying in this prediction problem that need to be tackled: off-policy stability and on-policy efficiency. The conventional temporal difference (TD) algorithm is known to perform very well in the on-policy setting, yet is not off-policy stable. On the other hand, the gradient TD and emphatic TD algorithms are off-policy stable, but are not on-policy efficient. This paper introduces novel algorithms that are both off-policy stable and on-policy efficient by using the oblique projection method. The empirical experimental results on various domains validate the effectiveness of the proposed approach.**

***Index Terms*— Off-policy, policy evaluation, reinforcement learning (RL), temporal difference (TD) learning.**

I. Introduction

POLICY evaluation plays a crucial role in reinforcement learning (RL): it estimates a value function that can predict the long-term return for a given fixed policy. *Temporal difference* (TD) learning is the central and powerful policy evaluation method in RL. However, it has two fundamental problems. The first problem is the **off-policy stability**. Although TD converges when samples are drawn "on-policy" (from the policy to be evaluated), it is shown to be possibly divergent when samples are drawn "off-policy." Off-policy stable methods are of wider interest, since they can learn while executing an exploratory policy, learn from demonstrations, and learn multiple tasks in parallel. Several different approaches have been explored to address off-policy learning. The "averager" method [1] needs to store many training examples, and thus is not practical for large-scale applications. Off-policy LSTD [2] is off-policy convergent, but its per-step computational complexity is quadratic in the number of parameters d of the function approximator. The most state-of-the-art off-policy stable algorithms with linear computational complexity are gradient TD (GTD) [3] and proximal gradient TD (PGTD) [4], which use stochastic primal-dual-based methods as powerful solvers. The second problem is the **on-policy efficiency**. Although GTD and PGTD are off-policy stable, they usually tend to have inferior performances in on-policy learning settings, especially in small-scale problems with relatively few samples [5]. On the other hand, the TD method is well-known for its on-policy efficiency, which explains well its popularity among RL researchers and practitioners. It is intriguing, therefore, to propose model-free policy evaluation algorithms that offer both off-policy stability and on-policy efficiency.

The major contribution of this paper is to explore policy evaluation algorithms that yield both off-policy stability and on-policy efficiency. To this end, we propose novel algorithms based on the oblique projection framework [6]. A computationally feasible criterion is proposed and used to derive algorithms with linear computational complexity per step. The off-policy stability is rigorously proved, and the on-policy and off-policy performances are demonstrated via thorough experimental studies.

Here is a roadmap for the rest of this paper. Section II introduces some RL background, reviews existing approaches to tackle the problem of off-policy stability, and puts off-policy policy evaluation in the framework of (weighted) oblique projection. Section III provides the stable and efficient TD (SETD) algorithm and a more general SETD(λ) algorithm using weighted oblique projection. Related works are discussed in Section IV and compared empirically with the proposed SETD in Section V.

II. Preliminaries

A. Reinforcement Learning

RL [7], [8] and approximate dynamic programming [9], [10] is a class of learning problems in which an agent interacts with an unfamiliar, dynamical, and stochastic environment, where the agent's goal is to optimize some measure of its long-term performance. This interaction is conventionally modeled as a Markov decision process (MDP). An MDP is defined as the tuple $(\mathcal{S}, \mathcal{A}, P_{ss'}^{a}, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are finite sets of states and actions, the transition kernel $P_{ss'}^{a}$ specifies the probability of transition from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$, $R(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function bounded by $R_{\max}$, and $0 \leq \gamma < 1$ is a discount factor. A stationary

Manuscript received January 8, 2018; revised April 10, 2018 and August 5, 2018; accepted September 2, 2018. Date of publication October 30, 2018; date of current version May 23, 2019. The work of B. Liu was supported by Tencent Rhino-bird Gift Funds. *(Corresponding author: Bo Liu.)*

D. Lyu, B. Liu, and S. Biaz are with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: daoming.lyu@auburn.edu; boliu@auburn.edu; biazsaa@auburn.edu).

M. Geist is with the Laboratoire Interdisciplinaire des Environnements Continentaux, Centre national de la recherche scientifique, Université de Lorraine, F-57000 Metz, France, and also with Google Brain, 75009 Paris, France (e-mail: matthieu.geist@univ-lorraine.fr).

W. Dong is with the Department of Computer Science and Engineering, The State University of New York at Buffalo, Buffalo, NY 14260 USA (e-mail: wendong@buffalo.edu).

Q. Wang is with the School of Computer Science and the Center for OPTical IMagery Analysis and Learning, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: crabwq@nwpu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2871361

policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a probabilistic mapping from states to actions. The main objective of an RL algorithm is to find an optimal policy. In order to achieve this goal, a key step in many algorithms is to estimate the value function under a given policy π, i.e., $V_\pi : \mathcal{S} \to \mathbb{R}$, a process known as *policy evaluation*. It is known that V_π is the unique fixed-point of the *Bellman operator* T_π, i.e.,

$$V_\pi = T_\pi V_\pi = R_\pi + \gamma P_\pi V_\pi \tag{1}$$

where R_π and P_π are, respectively, the reward function and transition kernel of the Markov chain induced by policy π. In (1), we may think of V_π as an $|\mathcal{S}|$-dimensional vector and write everything in a vector/matrix form. In the following, to simplify the notation, we often drop the dependence of T_π, V_π, R_π, and P_π to π. We denote by π_b the behavior policy that generates the data, and by π the target policy that we would like to evaluate. They are the same in the on-policy setting and different in the off-policy scenario. For the ith state-action pair (s_i, a_i), such that $\pi_b(a_i|s_i) > 0$, we define the importance-weighting factor $\rho_i = \pi(a_i|s_i)/\pi_b(a_i|s_i)$.

When $\mathcal{S}$ is large or infinite, we often use a linear approximation architecture for V_π with parameters $\theta \in \mathbb{R}^d$ and K-bounded basis functions $\{\varphi_i\}_{i=1}^d$, i.e., $\varphi_i : \mathcal{S} \to \mathbb{R}$ and $\max_i \|\varphi_i\|_\infty \le K$. We denote by $\phi(\cdot) := (\varphi_1(\cdot), \ldots, \varphi_d(\cdot))^\top$ the feature vector and by $\mathcal{F}$ the linear function space spanned by the basis functions $\{\varphi_i\}_{i=1}^d$, i.e., $\mathcal{F} = \{f_\theta \mid \theta \in \mathbb{R}^d$ and $f_\theta(\cdot) = \phi(\cdot)^\top\theta\}$. We may write the approximation of V in $\mathcal{F}$ in the vector form as $\hat{v} = \Phi\theta$, where Φ is the $|\mathcal{S}| \times d$ feature matrix. $\xi \in \mathbb{R}^{|\mathcal{S}|}$ denotes the vector representing the stationary probability distribution over the state space $\mathcal{S}$ and depends on behavior policy π_b. We also denote by $\Xi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ the diagonal matrix whose elements are $\xi(s)$. The solution of the TD algorithm is the fixed-point solution of the following *projected Bellman equation*:

$$\hat{v} = \Pi T_\pi(\hat{v}) \tag{2}$$

where $\Pi = \Phi(\Phi^\top \Xi \Phi)^{-1}\Phi^\top \Xi$ is the weighted least-squares projection weighted by ξ.

When only n training samples (collected by the behavior policy π_b) are available, the sample set is denoted as $\mathcal{D} = \{(s_i, a_i, r_i = r(s_i, a_i), s_i')\}_{i=1}^n$, $s_i \sim \xi$, $a_i \sim \pi_b(\cdot|s_i)$, $s_i' \sim P(\cdot|s_i, a_i)$. We denote by $\delta_i(\theta) := r_i + \gamma \phi_i'^\top\theta - \phi_i^\top\theta$ the TD error for the ith sample (s_i, a_i, r_i, s_i') and define $\Delta\phi_i = \phi_i - \gamma\phi_i'$, where ϕ_i (resp. ϕ_i') is the ith feature vector with respect to s_i (resp. s_i'). Finally, we define the covariance matrix C as $C := \mathbb{E}[\phi_i\phi_i^\top] = \Phi^\top\Xi\Phi$, where the expectations are with respect to ξ. For the ith sample in the training set $\mathcal{D}$, an unbiased estimate of C is $\hat{C}_i := \phi_i\phi_i^\top$.

B. Oblique Projection

This section introduces the oblique projection [11] and the oblique projected TD methods [6], and then extend it to weighted oblique projected TD framework. The oblique projection tuple (Φ, X) is defined as follows, where the rows of Φ are the basis vectors for the range of the projection and the rows of X are the basis vectors for the orthogonal complement of the null space of the projection.

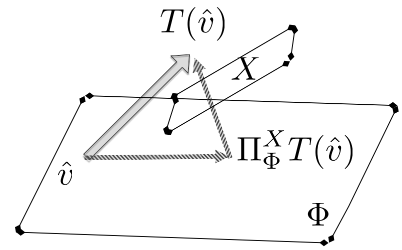


Fig. 1. Illustration of oblique projected TD.

Definition 1: The *Oblique Projection* operator Π_Φ^X

$$\Pi_\Phi^X = \Phi(X^\top\Phi)^{-1}X^\top$$

is a projection onto span(Φ) orthogonal to span(X).

Π_Φ^X is a projection, since it is idempotent: $(\Pi_\Phi^X)^2 = \Pi_\Phi^X$. This projection reduces to an orthogonal projection when the basis vectors for the range are orthogonal to the null space, and is more general than the orthogonal projection. For example, the weighted least-squares projection Π in (2) can be formulated as $\Pi = \Pi_\Phi^{\Xi\Phi}$, which defines the oblique projection onto the space spanned by Φ with basis $\{\phi(s) : s \in \mathcal{S}\}$ that is orthogonal to the space spanned by $\Xi\Phi$ with basis $\{\xi(s)\phi(s) : s \in \mathcal{S}\}$.

Next, we introduce the oblique projected TD as a more general framework to include the TD method and the residual gradient (RG) method [12]. Motivated by the extension from Π to Π_Φ^X, it is natural to extend the projected fixed-point equation (2) with oblique projection

$$\hat{v} = \Pi_\Phi^X T_\pi(\hat{v}). \tag{3}$$

Fig. 1 illustrates this. Instead of minimizing the distance between $\Pi T(\hat{v})$ and $\hat{v}$, the oblique projected TD aims to minimize the distance between $\Pi_\Phi^X T(\hat{v})$ and $\hat{v}$.

An intuitive question to ask is what the best oblique projection matrix X is. Is it TD, RG, some interpolation between them, or none of the above? To answer this, we present the following lemma, a workhorse of this paper.

Lemma 1 (Best Projection [6]): Given Φ, if V_π does not lie in $span(\Phi)$, the "best" approximation is

$$v^* = \Pi V_\pi = \Phi(\Phi^\top\Xi\Phi)^{-1}\Phi^\top\Xi V_\pi$$

which is also the solution of the oblique projected TD equation $v^* = \Pi_\Phi^{X^*} T v^*$ with

$$X^* = \left(L_\pi^\top\right)^{-1}\Xi\Phi \tag{4}$$

with $L_\pi := I - \gamma P_\pi$.

Proof: The solution of the oblique projected fixed-point equation $\hat{v} = \Pi_\Phi^X T(\hat{v})$ with respect to the oblique projection Π_Φ^X can be represented as the oblique projection $\Pi_\Phi^{L_\pi^\top X}$ of the true value function V [6], that is

$$\hat{v} = \Pi_\Phi^X T_\pi(\hat{v}) = \Pi_\Phi^{L_\pi^\top X} V.$$

As X^* satisfies $v^* = \Pi_\Phi^{(L_\pi^\top)X^*} V$. Let $\Pi_\Phi^{(L_\pi^\top)X^*} = \Pi$, we have $(L_\pi^\top)X^* = \Xi\Phi$ and thus we can have (4), which completes the proof. $\square$

C. Weighted Oblique Projection

The analytical formulation of X^* is often intractable to compute in real applications. The major reason is that P_π and consequently $(L_\pi^\top)^{-1}$ in X^* are not known in the RL setting. To address this challenge, we introduce the weighted oblique projection matrix $Y = \Xi^{-1}X$ and derive a stochastic approximation of X^* subject to a structural simplification assumption. From the definition of Y, it is evident that its optimum is attained at

$$Y^* = \Xi^{-1}X^* = \Xi^{-1}\left(L_\pi^\top\right)^{-1}\Xi\Phi$$

and the fixed point equation formulation in (3) becomes $\hat{v}_\theta = \Pi_\Phi^{\Xi Y} T_\pi \hat{v}_\theta$ accordingly. It turns out that both TD and RG solutions are weighted oblique projections with $Y_{\mathrm{TD}} = \Phi$ for TD, $Y_{RG} = L_\pi\Phi$ for RG. Next, we discuss the necessary conditions of the existence of the fixed-point solution.

Lemma 2 (Existence): The solution to the *weighted oblique projected Bellman equation*

$$\hat{v}_\theta = \Pi_\Phi^{\Xi Y} T_\pi \hat{v}_\theta \tag{5}$$

exists if $Y^\top\Xi\Phi$ and $Y^\top\Xi L_\pi\Phi$ are nonsingular, and the solution is

$$\theta = (Y^\top\Xi L_\pi\Phi)^{-1}Y^\top\Xi R. \tag{6}$$

Proof:

$$\begin{aligned}
\hat{v}_\theta &= \Pi_\Phi^{\Xi Y} T_\pi \hat{v}_\theta\\
\Leftrightarrow Y^\top\Xi\hat{v}_\theta &= Y^\top\Xi(R + \gamma P_\pi \hat{v}_\theta)\\
\Leftrightarrow Y^\top\Xi L_\pi\Phi\theta &= Y^\top\Xi R\\
\Leftrightarrow \theta &= (Y^\top\Xi L_\pi\Phi)^{-1}Y^\top\Xi R.
\end{aligned}$$

The first equality holds if $Y^\top\Xi\Phi$ is nonsingular, and the last equality holds if $Y^\top\Xi L_\pi\Phi$ is nonsingular. □

Therefore, the nonsingularity of $Y^\top\Xi\Phi$ and $Y^\top\Xi L_\pi\Phi$ guarantees the existence of $\Pi_\Phi^{\Xi Y}$ and θ as in (6). The extension from oblique projection to weighted oblique projection, though technically trivial, enables the design of stochastic approximation-based algorithms.

III. Algorithm Design

This section presents the design of the stable and efficient algorithm. We first present the motivation to use the oblique projection. Then, a computationally efficient criterion is proposed to overcome the computational intractability to compute X^*. Based on this criterion, an algorithm is proposed based on a diagonal approximation and is also extended to the multistep learning setting with eligibility trace.

A. Motivation

This paper aims at achieving off-policy stability for TD learning in off-policy settings. It is well-known that the TD method with linear function approximation has instability issues in off-policy learning settings [7], [13], which is largely due to the limitation of the projected fixed-point formulation in (2). TD solution, as a projected fixed-point formulation, is highly sensitive to the degree of "off-policyness," i.e., the difference between the behavior policy π_b and the target policy π. On the other hand, ΠV, being the "best" approximation [by the representation space span(Φ)] of the true value function V, is always unique and stable, yet is difficult to compute in RL settings. It is therefore desirable to propose a novel fixed-point formulation whose solution is close to the best approximation ΠV to enable off-policy stability. One possible way to achieve this is to use the weighted oblique projection operator $\Pi_\Phi^{\Xi Y}$ and change the vanilla projected Bellman formulation in (2) to the weighted oblique projected Bellman formulation in (5). Closeness to ΠV implies that the solution is less sensitive to the "off-policyness" than the TD solution. In a nutshell, this paper aims at proposing a weighted oblique projected TD framework in (5) to achieve off-policy stability via forcing proximity to the "best" approximation ΠV, with stochastic approximation methods.

B. Approximation Criteria

We first introduce a simple but important property of the optimal projection matrix X^*. We denote $\Lambda := L_\pi\Phi$. As $X^* = (L_\pi^\top)^{-1}\Xi\Phi$, we have

$$\Lambda^\top X^* = \Phi^\top(L_\pi^\top)\left(L_\pi^\top\right)^{-1}\Xi\Phi = \Phi^\top\Xi\Phi = C. \tag{7}$$

Motivated by this, Proposition 1 is presented to formulate the cornerstone of this paper.

Proposition 1: If the weighted oblique projection Y satisfies $Y^\top\Xi L_\pi\Phi = C$, and if Φ has full row rank ($rank(\Phi) = |\mathcal{S}|$), then we have $Y = Y^*$

Proof:

$$\begin{aligned}
&Y^\top\Xi L_\pi\Phi = C\\
\Leftrightarrow\ &Y^\top\Xi L_\pi\Phi = \Phi^\top\Xi\Phi\\
\Leftrightarrow\ &L_\pi^\top\Xi Y = \Xi\Phi \ (\text{as rank}(\Phi) = |\mathcal{S}|)\\
\Leftrightarrow\ &\Xi Y = \left(L_\pi^\top\right)^{-1}\Xi\Phi = X^*\\
\Leftrightarrow\ &Y^* = Y.
\end{aligned}$$

□

Although the rank condition is restrictive in real applications, it still offers helpful directions to approximate X^* in a computationally efficient way.

C. SETD Algorithm Design

In this paper, we investigate a special type of weighted oblique projection: Y can be decomposed into the product of a $|\mathcal{S}| \times |\mathcal{S}|$ diagonal matrix Ω and Φ such that $Y = \Omega\Phi$. The optimal Ω, termed Ω^*, can be obtained via

$$\Omega^* = \arg\min_\Omega \|\Xi\Omega\Phi - X^*\|_F$$

which is impossible to compute, since X^* is unknown. With (7), an approximation of Ω^*, termed Ω_S, can be computed as

$$\Omega_S = \arg\min_\Omega \|\Lambda^\top\Xi\Omega\Phi - C||_F. \tag{8}$$

The optimization problem reduces to a matrix regression problem, with $\|\cdot\|_F$ the Frobenius norm. With the sample-based estimation of $\Lambda^\top \Xi \Omega_S \Phi$ and C matrices, i.e.,

$$\Lambda^\top \Xi \Omega_S \Phi \leftarrow \frac{1}{n}\sum_{i=1}^{n} \omega_i \big(\phi_i - \gamma \phi_i'\big)\phi_i^\top$$
$$C \leftarrow \frac{1}{n}\sum_{i=1}^{n} \phi_i \phi_i^\top$$

with ω_i the ith diagonal element of Ω_S; the problem is formulated as

$$\min_{\omega_i} \frac{1}{n}\left\|\sum_{i=1}^{n}\left(\omega_i \Delta\phi_i\phi_i^\top - \phi_i\phi_i^\top\right)\right\|_F.$$

Therefore, (8) can be approximated as

$$\Omega_S \approx \arg\min_{\omega_i} \frac{1}{n}\left\|\sum_{i=1}^{n}\left(\omega_i \Delta\phi_i\phi_i^\top - \phi_i\phi_i^\top\right)\right\|_F.$$

Now, we propose a relaxed method to address this problem based on two observations. First, it is desirable that $\forall i$, $\Omega_S(s_i, s_i)$ be positive. This is intuitive. Second, instead of solving the above objective function, a relaxed sample-separable objective function Ω_S using the triangle inequality can be formulated as follows by denoting $\omega_i := \Omega_S(s_i, s_i)$:

$$\forall i, \omega_i = \arg\min_{\omega} \left\|\omega\Delta\phi_i\phi_i^\top - \phi_i\phi_i^\top\right\|_F, \quad \text{s.t. } \omega_{\rm i} \geq 0.$$

The closed-form solution of ω_i is

$$\omega_i = \max\left(\frac{\Delta\phi_i^\top\phi_i}{\|\Delta\phi_i\|^2}, 0\right) \tag{9}$$

where $\|\cdot\|$ is the ℓ_2-norm of a vector.

Here, we show the detailed deduction. To obtain (9), we first introduce the following lemmas to compute the singular value of rank-1 matrices. We first introduce Lemma 3 without proof, which is instrumental in the theoretical proof.

Lemma 3: A rank-1 real-valued square matrix $G = pq^\top$ where p, q are the vectors of the same length, the eigenvalues of G are

$$\lambda(G) = \{p^\top q, 0, 0, 0, \ldots\}$$

i.e., G has only one nonzero eigenvalue $p^\top q$, and all other eigenvalues are 0 and $Tr(G) = p^\top q$, where $Tr(\cdot)$ is the trace of a matrix.

Then, we introduce Lemma 4.

Lemma 4: A rank-1 real matrix (not necessarily square) $M = uv^\top$ has only one nonzero singular value $\sigma_{\max}(M) = \|u\|_2 \cdot \|v\|_2$, where $\|\cdot\|_2$ is the ℓ_2-norm of a vector, and the Frobenius norm and the trace norm of M are identical, i.e.,

$$\|M\|_* = \|M\|_F = \sigma_{\max}(M) = \|u\|_2 \cdot \|v\|_2. \tag{10}$$

Proof: We use M^H to represent the conjugate transpose of the M matrix and $\lambda(\cdot)$ to represent the eigenvalues of a square matrix. Then, we have

$$\lambda(M^H M) = \lambda(vu^\top uv^\top) = (u^\top u)\lambda(vv^\top).$$

From Lemma 3, we know that $\lambda(vv^\top)$ are $\{v^\top v, 0, 0, \ldots\}$, and thus, M has only one nonzero singular value $\sigma_{\max}(M)$

$$\begin{aligned}\sigma_{\max}(M) &= \sqrt{\lambda(M^H M)} = \sqrt{\lambda(vu^\top uv^\top)} \\ &= \sqrt{(u^\top u)\lambda(vv^\top)} = \sqrt{(u^\top u)(v^\top v)} \\ &= \|u\|_2 \cdot \|v\|_2\end{aligned}$$

and all other singular values of M are 0. Thus, $\|M\|_* = \|M\|_F = \|u\|_2 \cdot \|v\|_2$, which completes the proof. □

Based on Lemma 4, we now show the derivation of (9). To tackle the following trace norm minimization formulation:

$$\omega_i = \arg\min_{\omega} \left\|\omega\Delta\phi_i\phi_i^\top - \phi_i\phi_i^\top\right\|_* \tag{11}$$

we need to use the structure of the rank-1 matrices. We have

$$\omega\Delta\phi_i\phi_i^\top - \phi_i\phi_i^\top = (\omega\Delta\phi_i - \phi_i)\phi_i{}^\top$$

we denote $q_i(\omega) := (\omega\Delta\phi_i - \phi_i)$, and thus, we have

$$\begin{aligned}\omega_i &= \arg\min_{\omega} \left\|q_i(\omega)\phi_i^\top\right\|_* \\ &= \arg\min_{\omega} \|\phi_i\|_2 \cdot \|q_i(\omega)\|_2 \\ &= \arg\min_{\omega} \|q_i(\omega)\|_2.\end{aligned} \tag{12}$$

The second equality above comes from (10), and the third equality from the fact that $\|\phi_i\|_2$ does not depend on ω.

On the other hand, using $\|\cdot\|_F^2$ instead of trace norm in (11), we have

$$\omega_i = \arg\min_{\omega} \left\|q_i(\omega)\phi_i^\top\right\|_F^2 \tag{13}$$

with

$$\begin{aligned}\|q_i(\omega)\phi_i^\top\|_F^2 &= Tr\big(\phi_i q_i^\top(\omega) q_i(\omega)\phi_i^\top\big) \\ &= \big(\phi_i^\top\phi_i\big) Tr\big(q_i(\omega) q_i^\top(\omega)\big) \\ &= \big(\phi_i^\top\phi_i\big)\big(q_i^\top(\omega) q_i(\omega)\big) \\ &= \|\phi_i\|_2^2 \|q_i(\omega)\|_2^2.\end{aligned}$$

The first equality comes from the fact that $\|M\|_F^2 = Tr(M^H M)$. The third equality comes from Lemma 3. Then, we can see that (13) is equivalent to (12), as verified by Lemma 4. Therefore, both the trace norm and Frobenius norm minimizations are equivalent to

$$\omega_i = \arg\min_{\omega} \|\omega\Delta\phi_i - \phi_i\|_2^2. \tag{14}$$

By zeroing the gradient of the right hand-side of (14), we will have (9) as the final result, which is also the vector projection weight of ϕ_i projected onto $\Delta\phi_i$.

For the on-policy case, the update rule is now ready as $\theta_{i+1} = \theta_i + \alpha_i\omega_i\delta_i\phi_i$, where $\alpha_i \in (0, 1]$ is the stepsize. For the off-policy case, importance weights $\rho_i = (\pi(a_i|s_i)/\pi_b(a_i|s_i))$ are used to enable the algorithm to consider the discrepancies between the behavior policy π and the target policy π_b by properly weighing the observation, which is a standard way in off-policy learning [14], [15]

$$\begin{aligned}
&\mathbb{E}_{\pi}[\delta_i \omega_i \phi_i] \\
&= \sum_{s_{i+1}} \sum_{a_i} \sum_{s_i} P(s_i, a_i, s_{i+1}) \delta_i \omega_i \phi_i \\
&\approx \sum_{s_{i+1}} \sum_{a_i} \sum_{s_i} P(s_{i+1}|s_i, a_i) \pi(a_i|s_i) \xi(s_i) \delta_i \omega_i \phi_i \\
&= \sum_{s_{i+1}} \sum_{a_i} \sum_{s_i} P(s_{i+1}|s_i, a_i) \pi_b(a_i|s_i) \xi(s_i) \frac{\pi(a_i|s_i)}{\pi_b(a_i|s_i)} \delta_i \omega_i \phi_i \\
&= \mathbb{E}_{\pi_b}[\rho_i \delta_i \omega_i \phi_i].
\end{aligned}$$

The update rule is thus defined as

$$\theta_{i+1} = \theta_i + \alpha_i \rho_i \omega_i \delta_i \phi_i \tag{15}$$

where $\alpha_i \in (0, 1]$ is the stepsize. The resulting SETD is in Algorithm 1.

Algorithm 1 SETD Algorithm

1: INPUT: Sample set $\{\phi_i, r_i, \phi_i{}'\}_{i=1}^{n}$
2: **for** $i = 1, \ldots, n$ **do**
3: Compute $\phi_i, \Delta\phi_i, \delta_i = r_i + \gamma \phi_i'^{\top} \theta_i - \phi_i^{\top} \theta_i$.
4: Compute ω_i according to Eq. (9).
5: Compute θ_{i+1} according to Eq. (15).
6: **end for**

The computational cost per step is $O(d)$, as can be seen from the computation of (9) and (15).

D. Extension to Eligibility Traces

Here, we extend SETD to eligibility traces. First, we introduce the general λ-return with bootstrapping and discounting based on the importance-weighting factor by using the TD forward view

$$G_t^{\lambda\rho}(V) = \rho_t\big(r_{t+1} + \gamma\left[(1-\lambda)V(S_{t+1}) + \lambda G_{t+1}^{\lambda\rho}\right]\big)$$

and define the value function at s for a given policy π

$$V_\pi(s) = \mathbb{E}\big[G_t^{\lambda\rho}(V_\pi)|S_t = s, \pi\big] = T_\pi^{\lambda\rho} V_\pi(s)$$

where $\lambda \in [0, 1]$ is the bootstrapping parameter and $T_\pi^{\lambda\rho}$ is the λ-weighted Bellman operator for policy π. Using linear function approximation, we get the TD equation

$$\begin{aligned}
b - A\theta &= \Phi^{\top} \Xi \Omega_S R - \Phi^{\top} \Xi \Omega_S \Lambda \theta \\
&= \Phi^{\top} \Xi \Omega_S \big(T_\pi^{\lambda\rho} V_\theta - V_\theta\big).
\end{aligned}$$

Define $\delta_t^{\lambda\rho}(\theta) := G_t^{\lambda\rho}(\theta) - \phi_t^{\top}\theta$ and

$$P_\xi^{\pi} \delta_t^{\lambda\rho}(\theta) \omega_t \phi_t := \sum_s \xi(s) \mathbb{E}\big[\delta_t^{\lambda\rho}(\theta)|S_t = s, \pi\big] \omega_t \phi_t$$

where P_ξ^{π} is an operator. We also have

$$\begin{aligned}
\mathbb{E}\big[\delta_t^{\lambda\rho}(\theta)|S_t = s, \pi\big] &= T_\pi^{\lambda\rho} V_\theta(s) - V_\theta(s) \\
P_\xi^{\pi} \delta_t^{\lambda\rho}(\theta) \omega_t \phi_t &= \sum_s \xi(s) \mathbb{E}\big[\delta_t^{\lambda\rho}(\theta)|S_t = s, \pi\big] \omega_t \phi_t \\
&= \sum_s \xi(s) \big[T_\pi^{\lambda\rho} V_\theta(s) - V_\theta(s)\big] \omega_t \phi_t \\
&= \Phi^{\top} \Xi \Omega_S \big(T_\pi^{\lambda\rho} V_\theta - V_\theta\big).
\end{aligned}$$

Therefore, we have $P_\xi^{\pi} \delta_t^{\lambda\rho}(\theta) \omega_t \phi_t = \mathbb{E}[\delta_t^{\lambda\rho}(\theta) \omega_t \phi_t]$.

Consider the following identities:

$$\begin{aligned}
\delta_t^{\lambda\rho}(\theta) &= G_t^{\lambda\rho}(\theta) - \phi_t^{\top}\theta \\
&= \rho_t\big(r_{t+1} + \gamma\left[(1-\lambda)\phi_{t+1}^{\top}\theta + \lambda G_{t+1}^{\lambda\rho}\right]\big) - \phi_t^{\top}\theta \\
&= \rho_t\big(r_{t+1} + \gamma \phi_{t+1}^{\top}\theta - \phi_t^{\top}\theta + \phi_t^{\top}\theta\big) \\
&\quad - \rho_t \gamma \lambda \phi_{t+1}^{\top}\theta + \rho_t \gamma \lambda G_{t+1}^{\lambda\rho} - \phi_t^{\top}\theta \\
&= \rho_t\big(r_{t+1} + \gamma \phi_{t+1}^{\top}\theta - \phi_t^{\top}\theta\big) \\
&\quad + \rho_t \gamma \lambda \big(G_{t+1}^{\lambda\rho} - \phi_{t+1}^{\top}\theta\big) + \rho_t \phi_t^{\top}\theta - \phi_t^{\top}\theta \\
&= \rho_t \delta_t(\theta) + \rho_t \gamma \lambda \delta_{t+1}^{\lambda\rho}(\theta) + (\rho_t - 1)\phi_t^{\top}\theta
\end{aligned}$$

and

$$\begin{aligned}
&\mathbb{E}\big[(\rho_t - 1)\phi_t^{\top}\theta\big] \\
&\quad = \sum_s \xi(s) \sum_a \pi_b(a|s)(\rho_t - 1)\phi_t^{\top}\theta \\
&\quad = \sum_s \xi(s) \Big(\sum_a \pi(a|s) - \sum_a \pi_b(a|s)\Big) \phi_t^{\top}\theta = 0.
\end{aligned}$$

Hence, we can get

$$\begin{aligned}
b - A\theta &= \Phi^{\top} \Xi \Omega_S \big(T_\pi^{\lambda\rho} V_\theta - V_\theta\big) = P_\xi^{\pi} \delta_t^{\lambda\rho}(\theta) \omega_t \phi_t \\
&= \mathbb{E}\big[\delta_t^{\lambda\rho}(\theta) \omega_t \phi_t\big] \\
&= \mathbb{E}\big[\rho_t \delta_t(\theta) \omega_t \phi_t + \rho_t \gamma \lambda \delta_{t+1}^{\lambda\rho}(\theta) \omega_t \phi_t \\
&\qquad + (\rho_t - 1)\big(\phi_t^{\top}\theta\big) \omega_t \phi_t\big] \\
&= \mathbb{E}\big[\rho_t \delta_t(\theta) \omega_t \phi_t + \rho_t \gamma \lambda \delta_{t+1}^{\lambda\rho}(\theta) \omega_t \phi_t\big] \\
&= \mathbb{E}\big[\rho_t \delta_t(\theta) \omega_t \phi_t + \rho_{t-1} \gamma \lambda \delta_t^{\lambda\rho}(\theta) \omega_{t-1} \phi_{t-1}\big] \\
&= \mathbb{E}\big[\rho_t \delta_t(\theta) \omega_t \phi_t + \rho_{t-1} \gamma \lambda \big(\rho_t \delta_t(\theta) \\
&\qquad + \rho_t \gamma \lambda \delta_{t+1}^{\lambda\rho}(\theta) + (\rho_t - 1)\phi_t^{\top}\theta\big) \omega_{t-1} \phi_{t-1}\big] \\
&= \mathbb{E}[\rho_t \delta_t(\theta)(\omega_t \phi_t + \rho_{t-1} \gamma \lambda \omega_{t-1} \phi_{t-1} \\
&\qquad + \rho_{t-2} \rho_{t-1} \gamma^2 \lambda^2 \omega_{t-2} \phi_{t-2} + \cdots)] \\
&= \mathbb{E}[\delta_t(\theta) e_t]
\end{aligned}$$

where the eligibility trace vector is defined as $e_t = \rho_t(\omega_t \phi_t + \gamma \lambda e_{t-1})$. Therefore, we can now specify our final new algorithm, SETD(λ), by the following steps, for $t > 0$:

$$\begin{aligned}
w_t &= \max\left(\frac{\Delta\phi_t^{\top}\phi_t}{\|\Delta\phi_t\|^2}, 0\right) \\
e_t &= \rho_t(\lambda\gamma e_{t-1} + w_t \phi_t) \\
\theta_{t+1} &= \theta_t + \alpha \delta_t e_t
\end{aligned} \tag{16}$$

where $e_0 = \vec{0}$. It is shown in Algorithm 2.

Algorithm 2 SETD(λ)

1: INPUT: Sample set $\{\phi_t, r_t, \phi_t{}'\}_{t=1}^{n}$
2: Initialize $e_0 = \vec{0}$.
3: **for** $t = 1, \ldots, n$ **do**
4: Compute $\phi_t, \Delta\phi_t, \delta_t = r_t + \gamma \phi_t'^{\top} \theta_t - \phi_t^{\top} \theta_t$.
5: Compute ω_t according to Eq. (16).
6: Compute $e_t = \rho_t(\lambda\gamma e_{t-1} + \omega_t \phi_t)$.
7: Compute $\theta_{t+1} = \theta_t + \alpha_t \delta_t e_t$.
8: **end for**

IV. Related Work

This section presents the related work, which is primarily the emphatic TD (ETD) algorithm [16] and the Retrace(λ) algorithm [17].

The Retrace(λ) algorithm shares the same motivation with the SETD algorithm, i.e., off-policy stability and on-policy efficiency. To this end, it uses a capped importance ratio technique, which is shown to be superior to the conventional importance sampling method and the Tree backup method [18]. This research direction is complementary to our research and has the potential to combine with the SETD method, which is left for future research due to space limitations.

To the best of our knowledge, the closest work to ours is the ETD method [16]. ETD has indeed a weighted oblique projection structure similar to SETD, with $Y_E = \Omega_E \Phi$. The ith diagonal element $\Omega_E(i,i)$ is computed as

$$\Omega_E(0,0) = 1$$
$$\Omega_E(i,i) = 1 + \gamma \rho_{i-1} \Omega_E(i-1, i-1), \quad i > 0$$

and the ETD algorithm update law is

$$\theta_{i+1} = \theta_i + \alpha_i \Omega_E(i,i) \rho_i \delta_i \phi_i.$$

A more detailed explanation of the ETD algorithm from the oblique projection perspective is shown as follows. Similar to SETD, ETD also assumes that the weighted oblique projection Y_E can be approximated by the product of a diagonal matrix (termed Ω_E) and Φ, i.e., $Y_E = \Omega_E \Phi$. Then, a different technique is used based on the power series expansion, i.e.,

$$(L_\pi)^{-1} = (I - \gamma P_\pi)^{-1} = \sum_{k=0}^{\infty} (\gamma P_\pi)^k.$$

Then, the power series expansion is used to compute $\Xi\Omega$ as a whole. Since the optimal oblique projection matrix is $X^* = (L_\pi^\top)^{-1} \Xi \Phi$, it is evident that $\hat{X}_E = \Xi \Omega_E \Phi$ should be as close as possible to X^*, especially the diagonal elements. The diagonal elements of $\hat{X}_E$ are represented as a (column) vector f. One conjecture is that for the diagonal matrix of $\hat{X}_E$, it is desired that $f = (L_\pi^\top)^{-1} \xi$. By using the power series expansion, f can be expanded as

$$\begin{aligned} f &= \left(L_\pi^\top\right)^{-1} \xi = \left(\sum_{k=0}^{\infty} \left(\gamma P_\pi^\top\right)^k \right) \xi \\ &= \left(I + \gamma P_\pi^\top + \left(\gamma P_\pi^\top\right)^2 + \cdots + \left(\gamma P_\pi^\top\right)^k + \cdots \right) \xi. \end{aligned}$$

Readers familiar with the ETD learning algorithm know that this is actually identical to [16, eq. (13)], where a scalar follow-on trace is computed as[1]

$$\Omega_E(0,0) = 1$$
$$\Omega_E(t,t) = 1 + \gamma \rho_{t-1} \, \Omega_E(t-1, t-1), \quad t > 0$$

with $\Omega_E(t,t)$ denoting the t diagonal element of Ω_E. It turns out that

$$f_i = \xi(i) \lim_{t \to \infty} \mathbb{E}[\Omega_E(i,i) | S_t = s_i]$$

[1]We use subscript $\bullet_t$ to denote sequential samples, and subscription $\bullet_i$ to denote samples that are randomly sampled with replacement.

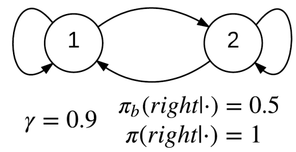


Fig. 2. Two-state MDP.

which leads to the standard ETD(0) algorithm

$$\theta_{t+1} = \theta_t + \alpha_t \Omega_E(t,t) \rho_t \delta_t \phi_t.$$

Previous works [19], [20] also associated ETD with oblique projection. This sheds a helpful light on understanding the family of the ETD learning algorithms. However, the ETD algorithm requires the sequential sampling condition, i.e., $s'_i = s_{i+1}, \forall i > 0$, which is not suitable for a set of samples collected from many episodes. This restriction is alleviated for the SETD algorithm.

We compare the two algorithms on the two-state MDP of [16]. As shown in Fig. 2, this environment has two actions, left and right, which take the process to the left or right states. The single feature is $[1, 2]^\top$ in the two states, and the discount factor $\gamma = 0.9$. The behavior policy π_b is to go left and right with equal probability from both the states, while the target policy π is to go right in both the states.

Since $P_\pi = [0\ 1\ 0\ 1]$, we have

$$L_\pi = I - \gamma P_\pi = \begin{bmatrix} 1 & -0.9 \\ 0 & 0.1 \end{bmatrix}$$

and

$$X^* = (L_\pi^\top)^{-1} \Xi \Phi = \begin{bmatrix} 0.5 \\ 14.5 \end{bmatrix}$$

with X^* is the optimal oblique projection matrix and $\Xi = 0.5I$.

Next, we compute the oblique projection matrices, X_{Ω_E} and X_{Ω_S}, for SETD and ETD, respectively.

According to the definition of matrix Ω_E in ETD algorithm by [16], let us use a (column) vector f_{ETD} to represent the diagonal elements of matrix Ω_E, and it is calculated as

$$f_{\mathrm{ETD}} = (I - \gamma P_\pi^\top)^{-1} \xi = \xi + \gamma P_\pi^\top \xi + (\gamma P_\pi^\top)^2 \xi + \cdots$$

Therefore, $f_{\mathrm{ETD}}(1) = 0.5$ and $f_{\mathrm{ETD}}(2) = 0.5 + 0.9 + (0.9)^2 + (0.9)^3 + \cdots = 9.5$. Therefore, we can get

$$\Omega_E = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.95 \end{bmatrix}. \tag{17}$$

Then, we have

$$X_{\Omega_E} = \Xi \Omega_E \Phi = \begin{bmatrix} 0.25 \\ 9.5 \end{bmatrix}. \tag{18}$$

For the SETD algorithm, each diagonal entry of Ω_S can be computed according to (9), which is

$$\Omega_S = \begin{bmatrix} 0 & 0 \\ 0 & 10 \end{bmatrix}. \tag{19}$$

TABLE I
COMPARISON ON THE TWO-STATE MDP

Algorithm	Ω	$\|\|\Lambda^\top \Xi\Omega\Phi - C\|\|_F^2$	X	$\|\|X - X^*\|\|_2$
SETD	Eq.(19)	**0.25**	Eq.(20)	**4.5277**
ETD	Eq.(17)	0.64	Eq.(18)	5.0062
TD	Eq.(21)	7.29	Eq.(22)	13.5

TABLE II
CONSIDERED VALUES FOR GRID-SEARCH

Parameter	**Evaluated Values**
α	$1*10^{-7}, 1*10^{-6}, 1*10^{-5}, 1*10^{-4}, 0.001, 0.01, 0.1,$ $3*10^{-6}, 5*10^{-6}, 7*10^{-6}, 9*10^{-6}, 2*10^{-6},$ $2.5*10^{-6}, 2*10^{-4}, 4*10^{-4}, 6*10^{-4}, 8*10^{-4},$ $0.002, ..., 0.009, 0.02, ..., 0.09, 0.2, 0.3, 0.4, 0.5, 0.6$
μ	$1*10^{-4}, 1*10^{-3}, 0.01, 0.1, 1, 4, 8, 16, 0.005, 0.05, 0.5$
λ	$0.4, 0.8$

Then, we have

$$X_{\Omega_S} = \Xi\Omega_S\Phi = \begin{bmatrix} 0 \\ 10 \end{bmatrix}. \tag{20}$$

Here is a summary of comparing the SETD and ETD on the two-state MDP domain, as shown in Table I. It should be noted that though TD will diverge on this domain, the solution to TD is the upper bound. As mentioned in Section II-C, the weighted oblique projection structure for TD is $Y_{\text{TD}} = \Phi$. This is equivalent to $Y_{\text{TD}} = \Omega_{\text{TD}}\Phi$ where

$$\Omega_{\text{TD}} = I. \tag{21}$$

Then, we have

$$X_{\text{TD}} = \Xi\Omega_{\text{TD}}\Phi = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \tag{22}$$

for this 2-state MDP domain. Table I shows the comparison of SETD and ETD in the diagonalized approximation of X^*. For TD

$$\|\Lambda^\top \Xi\Omega\Phi - C\| = \|\Lambda^\top \Xi\Phi - \Phi^\top \Xi\Phi\|_F^2 = \|\gamma\,(\Phi'^\top \Xi\Phi)\|_F^2.$$

According to the result, it can be seen that SETD performs better than ETD in X^* estimation.

V. EXPERIMENTAL STUDY

This section evaluates the effectiveness of the proposed algorithms. GTD2, TD with gradient correction term (TDC), and ETD are used for off-policy learning comparison, and TD, GTD2, TDC, and ETD are used for the on-policy case. It should be mentioned that since the major focus of this paper is value function approximation, comparisons on control learning performance are not reported here. We use α_{TD}, α_{ETD}, α_{SETD}, α_{GTD2}, μ_{GTD2} ($\beta_{\text{GTD2}} = \alpha_{\text{GTD2}} * \mu_{\text{GTD2}}$), and α_{TDC}, μ_{TDC} ($\beta_{\text{TDC}} = \alpha_{\text{TDC}} * \mu_{\text{TDC}}$) to denote the stepsizes for TD, ETD, SETD, GTD2, and TDC, respectively. In order to focus on the algorithm itself and make the comparison fair, which is similar to [5] and [14], only constant stepsize is considered in this paper. All stepsizes are chosen via a range of parameters similar to [14] that are based on the grid search method, as shown in Table II.

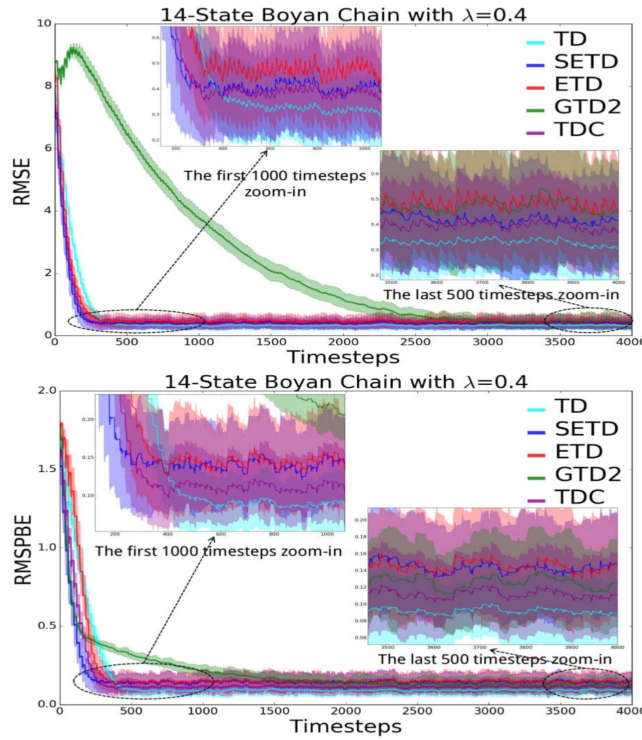


Fig. 3. 14-State Boyan Chain MDP, $\lambda = 0.4$.

Two metrics, root mean-squares error (RMSE) and root mean-squares projected Bellman error (RMSPBE) [3], are used as the performance measure

$$\text{RMSE} = \sqrt{\|\hat{v} - V\|_\Xi^2} \quad \text{RMSPBE} = \sqrt{\|\hat{v} - \Pi T\hat{v}\|_\Xi^2}.$$

A. On-Policy Comparison

1) Boyan Chain: Comparison studies are conducted on the Boyan Chain MDP, which has 14 states and one action with a 4-D state representation [21]. Algorithm 2 is compared with $\lambda = 0.4, 0.8$, as shown in Figs. 3 and 4, respectively. To enable the visibility of details, we zoom in the first 1000 timesteps and the last 500 timesteps in the figures. For $\lambda = 0.4$, constant stepsizes are $\alpha_{\text{TD}} = 0.2$, $\alpha_{\text{SETD}} = 0.4$, $\alpha_{\text{ETD}} = 0.04$, $\alpha_{\text{GTD2}} = 0.5$, $\mu_{\text{GTD2}} = 1$,$\alpha_{\text{TDC}} = 0.3$, and $\mu_{\text{TDC}} = 0.001$. For $\lambda = 0.8$, constant stepsizes are $\alpha_{\text{TD}} = 0.2$, $\alpha_{\text{SETD}} = 0.3$, $\alpha_{\text{ETD}} = 0.04$, $\alpha_{\text{GTD2}} = 0.3$, $\mu_{\text{GTD2}} = 1$, $\alpha_{\text{TDC}} = 0.3$, and $\mu_{\text{TDC}} = 0.001$. The learning curves are averaged over the results of 20 runs. Compared with all of the other approaches, SETD tends to have the fastest convergence speed and reaches similar steady-state performance on both RMSE and RMSPBE.

2) Mountain Car: The mountain car problem is also used to evaluate the validity of SETD. The mountain car MDP is an optimal control problem with a continuous 2-D state space. The steep discontinuity in the value function makes learning difficult. The Fourier basis [22] is used, which is a kind of fixed basis set. In this experiment, an empirically good policy π was first obtained, and then, we ran this policy π to collect trajectories that comprise the data set.

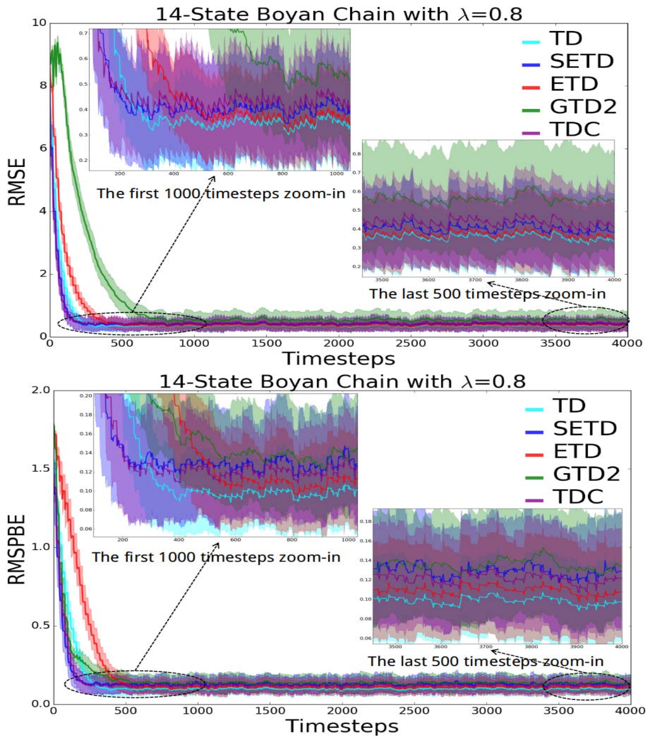


Fig. 4. 14-State Boyan Chain MDP, $\lambda = 0.8$.

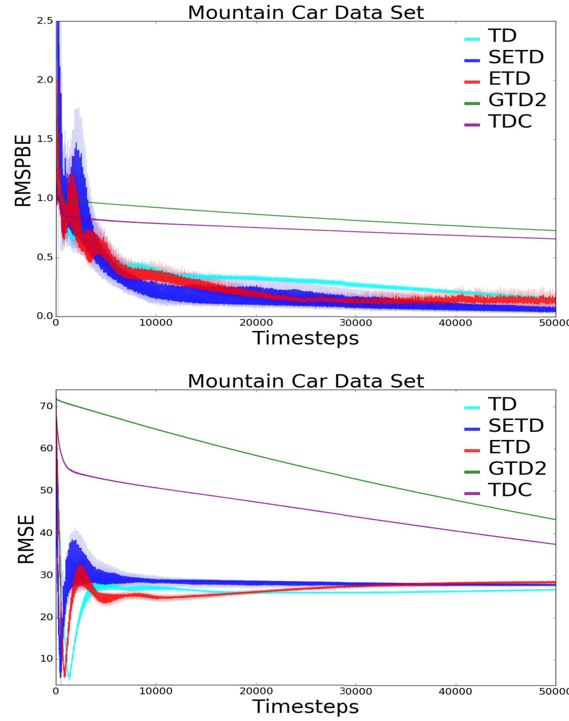


Fig. 5. Mountain Car with on-policy task.

On-policy policy evaluation of π is then conducted using the collected samples. The constant stepsizes are chosen as $\alpha_{\text{TD}} = 0.1$, $\alpha_{\text{ETD}} = 0.002$, $\alpha_{\text{SETD}} = 0.4$, $\alpha_{\text{GTD2}} = 0.05$, $\mu_{\text{GTD2}} = 1$, $\alpha_{\text{TDC}} = 0.04$, and $\mu_{\text{TDC}} = 0.05$. The learning curves are averaged over the results of 20 runs. The Monte Carlo estimation of V is estimated via 100 runs and each run has a maximum of 200 time steps.

As shown in Fig. 5, TD performs slightly better than SETD and ETD. TDC and GTD2 converge slowly in this domain.

B. Off-Policy Comparison

1) Baird Domain: The Baird example [12] is a well-known example to test the performance of off-policy convergent algorithms. Constant stepsizes $\alpha_{\text{SETD}} = 0.006$, $\alpha_{\text{GTD2}} = 0.005$, $\mu_{\text{GTD2}} = 1$, $\alpha_{\text{TDC}} = 0.006$, and $\mu_{\text{TDC}} = 16$, which are chosen via comparison studies as in [14].

Figs. 6 and 7 show the RMSPBE curve and RMSE curve of GTD2, SETD, and TDC of 4000 steps averaged over 20 runs. TDC has the largest variance of all; although the variance of SETD is larger than that of GTD2, SETD has a significant improvement over the GTD2 algorithm wherein the RMSPBE, the RMSE, and the variance are all substantially reduced. The low variance of the GTD2 learning curve can be explained by the advantage of the stochastic gradient method [4].

2) 400-State Random MDP: This domain is a randomly generated MDP with 400 states and 10 actions [14]. The transition probabilities are defined as $P(s'|s,a) \propto p_{ss'}^a + 10^{-5}$, where $p_{ss'}^a \sim U[0,1]$. The behavior policy π_b, the target policy π, as well as the starting distribution are sampled in a similar manner. Each state is represented by a 201-D feature vector, where the first 200 features were sampled from a uniform distribution, and the last feature was a constant one and the discount factor is set to $\gamma = 0.95$. The constant stepsizes are chosen as $\alpha_{\text{ETD}} = 2.5 * 10^{-6}$, $\alpha_{\text{SETD}} = 0.0008$, $\alpha_{\text{GTD2}} = 0.002$, $\mu_{\text{GTD2}} = 1$, $\alpha_{\text{TDC}} = 0.002$, and $\mu_{\text{TDC}} = 0.05$. Both the RMSE curve and the RMSPBE curve are averaged over 20 runs, and each run has 10 000 time steps. ETD is very sensitive to stepsizes on this domain and tends to diverge with a large stepsize, thus making the convergence very slow. As shown in Fig. 8, SETD performs better than GTD2 and ETD, although the variance is relatively larger than that in GTD2.

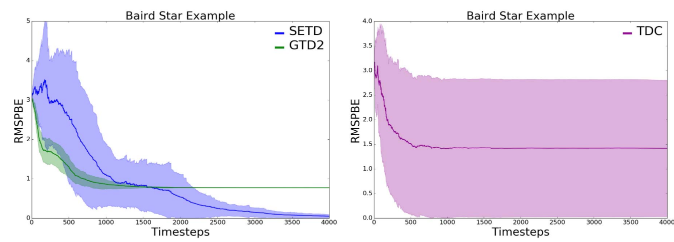


Fig. 6. Off-policy Baird domain (RMSPBE).

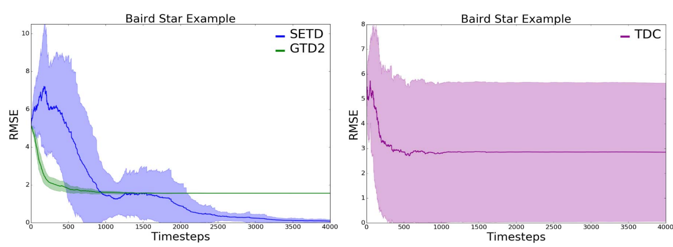


Fig. 7. Off-policy Baird domain (RMSE).

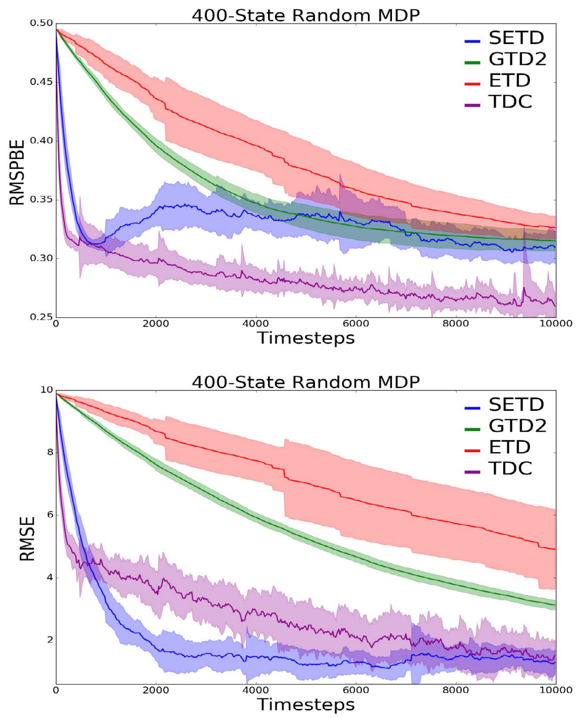


Fig. 8. Random MDP with off-policy task.

Overall, although SETD tends to have a relatively large variance in the initial phase, it is: 1) off-policy convergent; 2) converging much faster than GTD2 and TDC in both the on-policy and off-policy settings; and 3) less sensitive to stepsizes than ETD in the off-policy setting.

VI. Conclusion

This paper addressed the question: *how to design a policy evaluation algorithm that is both off-policy stable and on-policy efficient?* Novel algorithms have been proposed, based on oblique projection. Empirical experimental studies showed the effectiveness of the proposed algorithms in different learning settings.

There are numerous promising future work potentials along this direction of research. One is to conduct a sample complexity analysis such as a high probability error bound. This is important, because the performance of machine learning algorithms is always evaluated with a finite number of samples in real applications. Another interesting direction is to explore new approximation criteria. Current computationally tractable criteria of computing X^* are based on Proposition 1 (as used in SETD) or on the power series expansion of $(L_\pi^\top)^{-1}\Xi\Phi$ (as used in ETD). It is intriguing to explore if there exists other computationally tractable criteria. Finally, how to design a true-online SETD(λ) algorithm and compare its performance with other true-online algorithms [5], such as true-online GTD(λ) and true-online ETD(λ), is also worth exploring.

References

[1] G. J. Gordon, "Stable fitted reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 1052–1058.

[2] H. Yu, "Convergence of least squares temporal difference methods under general conditions," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 1–8.

[3] R. Sutton *et al.*, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 993–1000.

[4] B. Liu, J. Liu, M. Ghavamzadeh, S. Mahadevan, and M. Petrik, "Finite-sample analysis of proximal gradient TD algorithms," in *Proc. Conf. Uncertainty Artif. Intell.*, 2015, pp. 504–513.

[5] A. Adam and M. White, "Investigating practical linear temporal difference learning," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2016, pp. 494–502.

[6] B. Scherrer, "Should one compute the temporal difference fix point or minimize the Bellman Residual? The unified oblique projection view," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 959–966.

[7] R. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[8] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2042–2062, Jun. 2018.

[9] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 3, pp. 621–634, Mar. 2014.

[10] D. Wang, H. He, and D. Liu, "Adaptive critic nonlinear robust control: A survey," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3429–3451, Oct. 2017.

[11] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Philadelphia, PA, USA: SIAM, 2003.

[12] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Proc. Int. Conf. Mach. Learn.*, 1995, pp. 30–37.

[13] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 674–690, May 1997.

[14] C. Dann, G. Neumann, and J. Peters, "Policy evaluation with temporal differences: A survey and comparison," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 809–883, 2014.

[15] M. Geist and B. Scherrer, "Off-policy learning with eligibility traces: A survey," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 289–333, 2014.

[16] R. S. Sutton, A. R. Mahmood, and M. White, "An emphatic approach to the problem of off-policy temporal-difference learning," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2603–2631, 2015.

[17] R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare, "Safe and efficient off-policy reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1–9.

[18] D. Precup, R. S. Sutton, and S. P. Singh, "Eligibility traces for off-policy policy evaluation," in *Proc. Int. Conf. Mach. Learn.*, 2000, pp. 759–766.

[19] H. Yu, "On convergence of emphatic temporal-difference learning," in *Proc. Conf. Learn. Theory*, 2015, pp. 1724–1751.

[20] A. Mahmood, "Incremental off-policy reinforcement learning algorithms," Ph.D. dissertation, Dept. Comput. Sci., Univ. Alberta, Edmonton, AB, Canada, 2017.

[21] J. A. Boyan, "Technical update: Least-squares temporal difference learning," *Mach. Learn.*, vol. 49, nos. 2–3, pp. 233–246, 2002.

[22] G. Konidaris, S. Osentoski, and P. S. Thomas, "Value function approximation in reinforcement learning using the Fourier basis," in *Proc. 25th Conf. Artif. Intell.*, 2011, pp. 380–385.

Daoming Lyu received the B.S. degree in electrical engineering from Southwest University, Chongqing, China, in 2011, and the M.E. degree in biomedical engineering from Zhejiang University, Hangzhou, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL, USA.

His current research interests include reinforcement learning, symbolic planning, healthcare informatics, and artificial intelligence.

Bo Liu (M'18) received the Ph.D. degree from the University of Massachusetts Amherst, Amherst, MA, USA, in 2015. He is currently a Tenure-Track Assistant Professor with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL, USA. He has over 30 publications on several notable venues, such as the Conference on Neural Information Processing Systems, the Conference on Uncertainty in Artificial Intelligence, the Association for the Advancement of Artificial Intelligence, the International Joint Conferences on Artificial Intelligence, the *Journal of Artificial Intelligence Research*, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and the *ACM Transactions on Embedded Computing Systems*.

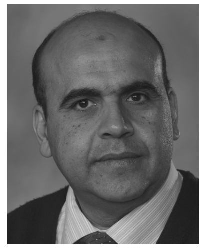
Dr. Liu was a recipient of the UAI-2015 Facebook Best Student Paper Award.

Matthieu Geist received the Electrical Engineering and the M.Sc. degrees in applied mathematics from Supélec, Gif-sur-Yvette, France, in 2006, the Ph.D. degree in applied mathematics from the University Paul Verlaine of Metz, Metz, France, in 2009, and the Habilitation degree from University Lille 1, Lille, France, in 2016.

From 2010 to 2017, he was an Assistant Professor with CentraleSupélec, Châtenay-Malabry, France. In 2017, he joined the Interdisciplinary Laboratory for Continental Environments, Centre national de la recherche scientifique, Université de Lorraine, Lorraine, France, as a Full Professor in applied mathematics. Since 2018, he has been with Google Brain, Paris, France. His current research interests include machine learning, especially reinforcement learning and imitation learning, as well as various applications.

Wen Dong received the Ph.D. degree in media arts and sciences from the Massachusetts Institute of Technology, Cambridge, MA, USA.

He is currently an Assistant Professor of computer science and engineering with The State University of New York at Buffalo, Buffalo, NY, USA, with a joint appointment with the Institute of Sustainable Transportation and Logistics. He focuses on modeling human interaction dynamics with the stochastic process theory through combining the power of big data and the logic/reasoning power of agent-based models, to solve our societies' most challenging problems such as transportation sustainability and efficiency.



Saad Biaz (M'98) received the Ph.D. degree in electrical engineering from University Henri Poincaré, Nancy, France, in 1989, and the Ph.D. degree in computer science from Texas A&M University, College Station, TX, USA, in 1999.

He has held faculty positions with the Ecole Supérieure de Technologie de Fès, Fes, Morocco, and Al Akhawayn University, Ifrane, Morocco. He is currently a Professor of computer science and software engineering with Auburn University, Auburn, AL, USA. His research is funded by the National Science Foundation and the Department of Defense. His current research interests include distributed systems, wireless networking, mobile computing, and unmanned flight.

Dr. Biaz was a recipient of the Excellence Fulbright Scholarship in 1995. He has served on the committees of several conferences and as reviewer for several journals.

Qi Wang (M'15–SM'15) received the B.E. degree in automation and the Ph.D. degree in pattern recognition and intelligent systems from the University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively.

He is currently a Professor with the School of Computer Science, with the Unmanned System Research Institute, and with the Center for OPTical IMagery Analysis and Learning, Northwestern Polytechnical University, Xi'an, China. His current research interests include computer vision and pattern recognition.