Online Reinforcement Learning Control by Direct Heuristic Dynamic Programming: From Time-Driven to Event-Driven

Qingtao Zhao, Jennie Si, *Fellow, IEEE*, and Jian Sun, *Senior Member, IEEE*

***Abstract*— In this work, time-driven learning refers to the machine learning method that updates parameters in a prediction model continuously as new data arrives. Among existing approximate dynamic programming (ADP) and reinforcement learning (RL) algorithms, the direct heuristic dynamic programming (dHDP) has been shown an effective tool as demonstrated in solving several complex learning control problems. It continuously updates the control policy and the critic as system states continuously evolve. It is therefore desirable to prevent the time-driven dHDP from updating due to insignificant system event such as noise. Toward this goal, we propose a new event-driven dHDP. By constructing a Lyapunov function candidate, we prove the uniformly ultimately boundedness (UUB) of the system states and the weights in the critic and the control policy networks. Consequently, we show the approximate control and cost-to-go function approaching Bellman optimality within a finite bound. We also illustrate how the event-driven dHDP algorithm works in comparison to the original time-driven dHDP.**

***Index Terms*— Direct heuristic dynamic programming (dHDP), event-driven/time-driven dHDP, reinforcement learning (RL).**

I. Introduction

In this work, time-driven learning refers to the machine learning method that updates parameters in a prediction model continuously as new data arrives. While important applications such as autonomous robots and intelligent agents on the web or in mobile devices require on-the-fly and continuous adaptation to environment changes, catastrophic forgetting [1], [2] can occur and thus drastically disrupt prediction performance. From a human learning perspective, such periodic, continuous update of the prediction model does not best represent biological learning. Long-established theory based on extensive experimental data shows that short-term memory decays very slowly when the environment is undisturbed, but decays amazingly fast otherwise. In machine learning, disruption of a learned model by new learning is a recognized feature of neural networks [2]–[8].

Forgetting after learning has plagued the machine learning field for many years and it has attracted attention from machine learning researchers who seek scalable and effective learning methods. A notable such progress was reported in [9] where several Atari 2600 games were learned sequentially by elastic weight consolidation (EWC) algorithm. To learn a new task, EWC tempers the network parameters based on previous task(s): it enables fast learning rates on parameters that are poorly constrained by the previous tasks and slow learning rates for those that are crucial. Incremental learning is an idea of triggering learning only if new patterns are sufficiently different from previous ones [10], [11]. For example, the Learn++ algorithm is to make the distribution update rule contingent on the ensemble error instead of the previous hypothesis error to allow for efficient incremental learning of new data that may introduce new classes. In [7], learning without forgetting (LwF) was proposed by employing convolutional neural networks. Given a set of shared weights across all tasks, it optimizes the shared weights, old task weights, and new task weights simultaneously based on an objective function including the predicted errors of old tasks, new tasks, and a regulation term. Similarly, to alleviate catastrophic forgetting, Zenke *et al*. [8] allowed individual synapses to estimate their importance for learned task. When the weights learn a new task, the cost-to-go function is modified by adding a penalty term of the summed parameter error functions of all previous tasks to reduce large changes in important parameters.

Continuous-time nonlinear systems have been considered to address the forgetting after learning problem in RL or ADP. In [12] and [13], the authors established a framework to update neural network weights in an event-triggered manner while the weight convergence and system stability were still guaranteed. But the results were obtained for nonlinear affine systems. Additionally, an observer network was required in [12] and a system identifier network was required in [13]. In [14]–[16], the authors considered a partially unknown affine nonlinear system. They showed the weight convergence of the neural networks and controlled system stability while taking into account control input constraints and under different event triggering conditions. Also dealing with nonlinear affine systems, [17]–[21] take into account an internal uncertainty or an external disturbance by the proposed event-triggered robust control using ADP structures. A different event-triggered condition based on value functions was proposed in [22] for continuous-time nonlinear systems using ADP. The convergence of the performance index was guaranteed while the system stability and weight boundedness were also considered. But such triggering condition is indirect and its dependence on value function may result in false triggers as value function approximation errors vary, sometimes to a large degree.

A small set of papers have addressed discrete-time event-triggered ADP by proving system stability, weight convergence in the approximating neural networks, and optimality of the learned policy [23]–[25]. The methods in [23] and [24] are restricted to affine nonlinear systems and they also need an identifier neural network as the state at time $k+1$ is required to solve the HJB equation, the use of which will inevitably introduce errors into the solution and adversely affect the reproducibility of the results. In addition to requiring an identifier neural network for [24] and [25], the event-triggered condition in those works was obtained during the derivation of Lyapunov stability analysis of the nonlinear dynamic system. As a result, the event-triggered condition is quite involved, and it is not clear how to constructively apply this condition in design.

In recent years, ADP has developed into powerful tools for optimally and adaptively control nonlinear dynamic systems driven by data without (complete) knowledge of the system dynamics. Among those ADP methods that deal with continuous-time nonlinear

Manuscript received December 20, 2019; revised May 3, 2020 and October 16, 2020; accepted January 16, 2021. The work of J. Si was supported in part by the NSF under Grant #1563921 and Grant #1808752. *(Corresponding author: Jennie Si.)*

Qingtao Zhao and Jian Sun are with the Key Laboratory of Intelligent Control and Decision of Complex System, Beijing Institute of Technology, Beijing 100081, China (e-mail: zhaoqingtaophx@163.com; sunjian@bit.edu.cn).

Jennie Si is with the Department of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: si@asu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3053037.

Digital Object Identifier 10.1109/TNNLS.2021.3053037

system of different forms, [26]–[28] have considered the optimality of the policy iteration algorithms, but the design or the convergence of approximating neural network weights was not considered. For discrete-time systems, Bellman optimality of the designed control policy is proved based on policy iteration [29] and value iteration [30] for affine nonlinear systems, and Bellman optimality was also obtained in [31] for general nonaffine systems. However, system stability was not provided in [29]–[31], and additionally, neural network weight convergence was not provided in [30]. Aside from those important theoretical properties, practically useful ADP algorithms have always been sought after. In this regard, the dHDP [32] (an online, continuous learning algorithm) has shown its promise in solving realistic, complex, and meaningful engineering problems that few other ADP or RL algorithms have been able to demonstrate in the area of adaptive optimal control. Since its introduction, the dHDP has been applied to some complex control problems such as Apache helicopter stabilization, tracking, and reconfiguration control [33]–[35], damping low-frequency oscillation in large power systems [36], and most recently, robotic prosthesis control tested on amputee subjects [37], [38] and robot-assisted human rehabilitation [39]. However as previously described, we also noticed parameter drift due to noise after learning [38], [40]. An enhanced dHDP with an ability to learn only significant events is thus expected to further improve the applicability of the original dHDP [32].

Therefore, in this work, we aim to develop an easy-to-implement, event-based dHDP to update the policy and critic network weights driven by events directly reflected in system states.

Our proposed work differs from [12]–[22]. Almost all previous event-triggered ADP algorithms were developed for continuous-time systems, which also only deal with nonlinear dynamics that can be described as affine systems. Among the few existing works for discrete-time systems, Sahoo *et al*. [23] proposed a near optimal event-driven control for nonlinear systems using an ADP structure under some assumptions. However, this method was again, only applicable for affine systems and an identifier neural network was also necessary. More recently, Ha *et al*. [24] investigated an event-based controller for affine discrete-time systems with constrained inputs. The stability of the systems was guaranteed with Lyapunov analysis tools but the approximation errors of the neural networks were ignored. Additionally, a third model neural network was needed. In [25], the authors proved the convergence of the weights for general discrete-time systems in an input to state stable framework but a model neural network was still needed.

With previous works laid out and discussed in the above, our new contributions toward constructing theoretically suitable and practically useful event-driven dHDP are summarized below.

1) We base our current work on an established dHDP online learning control algorithm. Also, we have derived a new and simple event-triggered condition. Together, our event-driven dHDP is potentially useful in practical applications.
2) Compared with previous works, which require the nonlinear dynamics under consideration to be described as affine systems and/or the design of which requires an identifier/model neural network, our event-driven dHDP is applicable to general, discrete-time nonlinear systems and learns from data without the requirement of learning a dynamic system model either online or offline.
3) We show that our proposed algorithm retains important analytical properties (convergence of approximating weights, stability of the controlled system, and optimality of the learned solution) that are useful as assurances for applications.

In the remainder of this brief, Section II will be used to formulate the event-driven dHDP for general, discrete-time nonlinear systems. Section III provides the proofs of system stability and neural network weight convergence as well as the solution optimality. We use an example to illustrate the proposed algorithm in Section IV.

II. Problem Formulation

Consider a general nonlinear discrete-time system with unknown dynamics

$$x(k+1) = f(x(k), u(k)) \tag{1}$$

where $x \in \mathbb{R}^m$ is the system state and $u \in \mathbb{R}^n$ system input.

Assumption 1: System (1) is controllable. $x(k) = 0$ is a unique equilibrium point and $f(0, 0) = 0$.

In time-driven ADP, a control input $u(k)$ is generated at each sampling time k as a function of system state $x(k)$. In event-driven ADP, which is considered in this work, the control input is updated only when there is a driving event reflected in system states going beyond a threshold. As such, the control inputs will be kept constant in a zero-order holder (ZOH) after a driving event time instant. We define the event indices as $\{\delta_k\}_{k=0}^{\infty}$ ($\delta_0 = 0$). The control law can then be represented as

$$u(k) = u(\delta_k) \quad \forall \delta_k \leqslant k < \delta_{k+1}. \tag{2}$$

We introduce the event-driven state error as

$$e(k) = x(\delta_k) - x(k) \quad \forall \delta_k \leqslant k < \delta_{k+1}. \tag{3}$$

Then system (1) can be rewritten as

$$x(k+1) = f(x(k), u(e(k) + x(k))). \tag{4}$$

Let the event instants be determined by the following condition:

$$\|e(k)\| \leq e_T \tag{5}$$

where e_T is a time-varying threshold variable that will be investigated in next section.

Consider the following cost-to-go function with event-driven control:

$$V(x(k)) = \sum_{j=k}^{\infty} r(x(j), u(x(j) + e(j))) \tag{6}$$

where

$$\begin{aligned} &r(x(k), u(x(k) + e(k))) \\ &\quad = r(x(k), u(x(\delta_k))) \\ &\quad = x^T(k) Q x(k) + u^T(x(\delta_k)) R u(x(\delta_k)). \end{aligned} \tag{7}$$

In the above equation, both Q and R are positive definite matrices whose maximum eigenvalue and minimum eigenvalue are $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$, respectively. For simplicity, we denote $r(x(k), u(x(\delta_k))$ as $r(k)$.

It is noted that the $V(x(k))$ reflects a measure of the performance index at state $x(k)$ and satisfies the Bellman equation

$$V(x(k)) = r(k) + V(x(k+1)). \tag{8}$$

From Bellman's optimality principle, the optimal cost-to-go function is therefore

$$V^*(x(k)) = \min_{u(x(\delta_k))} \{r(k) + V^*(x(k+1))\}. \tag{9}$$

As a model of the environment is unknown for system (1), our design, as in the original dHDP, relies on memory in place of using predicted system states when formulating the Bellman error for training the critic network.

To achieve event-driven learning control by dHDP, we adopt the architecture as shown in Fig. 1, where both the critic network and

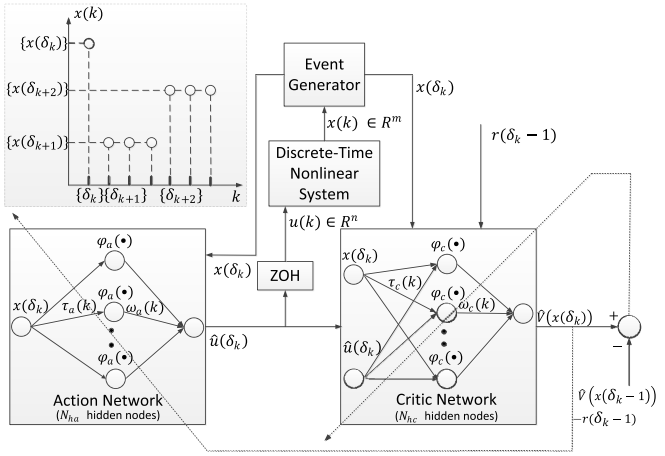

Fig. 1. Relationship between event time instants δ_k and general time k is shown in the upper left corner. As (2) and (3) show, only at event time instants δ_k, the state $x(k) = x(\delta_k)$ is used to update $u(\delta_k)$. At other time instants, the ZOH element provides the needed control input to interact with the system. Correspondingly as shown in (14) and (17), the weights in the critic and action networks learn at event time instants $k = \delta_k$ and remain unchanged without triggering event. The critic network approximates the cost-to-go as a function of $x(\delta_k)$ and $u(\delta_k)$, while the action network approximates the input control as a function of $x(\delta_k)$.

the action network are universal approximating neural networks. The nonlinear thresholding functions $\phi_c(.)$ and $\phi_a(.)$ in the hidden layer are hyperbolic tangents. We use $\tau_c(k), \tau_a(k)$ and $\omega_c(k), \omega_a(k)$ to denote the input-to-hidden layer weights and the hidden-to-output layer weights for the critic and the action networks, respectively.

Given Fig. 1, we consider the approximate cost-to-go function $\hat{V}(x(k))$ and the approximate input $\hat{u}(k)$ of the following form:

$$\hat{V}(x(k)) = \hat{\omega}_c^T(k)\phi_c(k) \tag{10}$$

$$\hat{u}(k) = \hat{\omega}_a^T(k)\phi_a(k). \tag{11}$$

In this work, the input-to-hidden layer weights $\tau_c(k), \tau_a(k)$ are chosen initially at random and kept constants as was the case in [41] and only the output layer weights $\omega_c(k), \omega_a(k)$ are updated during learning. As shown in [41], such neural networks can be universally approximating.

The critic network weights are updated in order to minimize the following approximation error:

$$E_c(k) = \frac{e_c^T(k)e_c(k)}{2} \tag{12}$$

$$e_c(k) = \hat{V}(x(k)) - [\hat{V}(x(k-1) - r(k-1)]. \tag{13}$$

Under the event-driven learning mechanism (2), learning and adaptation takes place only at event instants, that is $k = \delta_k$ and the control input holds as a constant during the event intervals. Then the critic network weights are updated as

$$\hat{\omega}_c(k+1) = \begin{cases} \hat{\omega}_c(k) - l_c\phi_c(k)\big[\hat{\omega}_c^T(k)\phi_c(k) + r(k-1) \\ \qquad -\hat{\omega}_c^T(k-1)\phi_c(k-1)\big]^T, & k = \delta_k \\ \hat{\omega}_c(k), \quad \delta_k < k < \delta_{k+1} \end{cases} \tag{14}$$

where l_c denotes the learning rate of the critic network.

Similar to [32], the action network weights are adjusted to minimize the following approximation error:

$$E_a(k) = \frac{e_a^T(k)e_a(k)}{2} \tag{15}$$

$$e_a(k) = \hat{V}(x(k)). \tag{16}$$

Similar to the critic network, the action network weights are updated as follows considering that the weight updates are driven by significant events in the system states:

$$\hat{\omega}_a(k+1) = \begin{cases} \hat{\omega}_a(k) - l_a\phi_a(k)\big[\hat{\omega}_c^T(k)C(k)\big] \\ \quad \times \big[\hat{\omega}_c^T(k)\phi_c(k)\big]^T, & k = \delta_k \\ \hat{\omega}_a(k), \quad \delta_k < k < \delta_{k+1} \end{cases} \tag{17}$$

where l_a denotes the learning rate of the action network and the components of $C(k) \in \mathbb{R}^{N_{hc} \times n}$ are expressed as

$$C_{li}(k) = \frac{1}{2}\big(1 - \phi_{cl}^2(k)\big)\tau_{c_{l,m+i}} \qquad l = 1, \ldots, N_{hc}, \quad i = 1, \ldots, n \tag{18}$$

where l and i are the indices of the hidden and input neurons, while m and n denote the dimension of system state and input.

We refer the critic and action update rules in (14) and (17) as event-driven dHDP. Let ω_c^* and ω_a^* be the optimal weights of the critic network and the action network, respectively, that is,

$$\omega_c^* = \arg\min_{\hat{\omega}_c} \|\hat{V}(x(k)) - [\hat{V}(x(k-1) - r(k-1)]\| \tag{19}$$

$$\omega_a^* = \arg\min_{\hat{\omega}_a} \|\hat{V}(x(k))\|. \tag{20}$$

Then we have

$$V^*(x(k)) = \omega_c^{*T}(k)\phi_c(k) + \epsilon_c \tag{21}$$

$$u^*(k) = \omega_a^{*T}(k)\phi_a(k) + \epsilon_a \tag{22}$$

where $V^*(x(k))$ and $u^*(k)$ denote the optimal cost-to-go function and optimal control input, while ϵ_c and ϵ_a are the neural network reconstruction errors of the critic network and the action network, respectively.

III. Main Results

As previously described, our goal is to devise a dHDP-based ADP online learning method that adapts the action and the critic weights during significant system events reflected in system states. That is to say that learning and adaptation does not necessarily take place regularly at each sampling time when an observation is made. We therefore will provide a sufficient condition under which learning takes place as driven by events while the closed-loop system stability is guaranteed. Specifically, we will show that both the weight parameters and the system states will remain UUB.

We define the weight estimation errors and some auxiliary variables as follows:

$$\begin{aligned} \tilde{\omega}_c(k) &= \hat{\omega}_c(k) - \omega_c^* \\ \xi_c(k) &= \big(\hat{\omega}_c(k) - \omega_c^*\big)^T\phi_c(k) = \tilde{\omega}_c^T(k)\phi_c(k) \\ \tilde{\omega}_a(k) &= \hat{\omega}_a(k) - \omega_a^* \\ \xi_a(k) &= \big(\hat{\omega}_a(k) - \omega_a^*\big)^T\phi_a(k) = \tilde{\omega}_a^T(k)\phi_a(k). \end{aligned} \tag{23}$$

Assumption 2: For the critic network and the action network, the optimal weights, the activation functions and the reconstruction errors are bounded, that is $\|\omega_c^*\| \leqslant \omega_{cm}$, $\|\omega_a^*\| \leqslant \omega_{am}$, $\|\phi_c\| \leqslant \phi_{cm}$,$\|\phi_a\| \leqslant \phi_{am}$, $\|\epsilon_c\| \leqslant \epsilon_{cm}$, $\|\epsilon_a\| \leqslant \epsilon_{am}$. Besides, the activation function ϕ_a is Lipschitz continuous and satisfies

$$\|\phi_a(x_1) - \phi_a(x_2)\| \leq L_a\|x_1 - x_2\| \tag{24}$$

for all $x_1, x_2 \in \Omega$ where L_a is a positive constant and Ω is the domain of function $f(x,u)$.

Theorem 1: Consider nonlinear system (1) and let Assumptions 1-2 hold. Given a stabilizing initial weight $\hat{\omega}_a(0)$, let the critic and action network weights update iteratively according to (14) and (17). If a learning event occurs based on the following event-driven condition:

$$\|e(k)\|^2 \leqslant \frac{\lambda_{\min}(Q)\beta}{2\lambda_{\max}(R)\|\hat{\omega}_a\|^2 L_a^2}\|x(k)\|^2 \tag{25}$$

where $0 \leqslant \beta < 1$ and let the learning rates of the action and critic networks satisfy

$$l_c < \frac{1}{\|\phi_c(k)\|^2}, \quad l_a < \frac{1}{\|\phi_a(k)\|^2} \tag{26}$$

then we have the following results:

1) The errors between the optimal network weights ω_c^*, ω_a^* and their estimates $\hat{\omega}_c(k)$ and $\hat{\omega}_a(k)$ are UUB.
2) The control $u(k)$, which is parameterized by $\omega_a(k)$, is a stabilizing control to guarantee the UUB of the nonlinear system under the proposed event-driven dHDP algorithm.

Proof: Consider the Lyapunov function candidate defined as follows:

$$L(k) = L_1(k) + L_2(k) + L_3(k) + L_4(k) \tag{27}$$

where

$$L_1(k) = \frac{1}{l_c}\mathrm{tr}\big[\tilde{\omega}_c^T(k)\tilde{\omega}_c(k)\big], \quad L_2(k) = \frac{1}{\gamma\, l_a}\mathrm{tr}\big[\tilde{\omega}_a^T(k)\tilde{\omega}_a(k)\big]$$
$$L_3(k) = \frac{1}{2}\|\xi_c(k-1)\|^2, \quad L_4(k) = V(x(k)).$$

Given an initially stabilizing $\hat{\omega}_a(0)$, $L_4(k)$ is finite at time $k=0$. Then we could prove $\hat{\omega}_a(k)$ is stabilizing for each k by means of the mathematical induction. First, we assume $\hat{\omega}_a(k)$ is stabilizing.

Now we consider the following two cases.

Case 1 *(No Learning Event Is Observed at Time Instant k)*: As $\tilde{\omega}_c(k+1) = \tilde{\omega}_c(k)$ and $\tilde{\omega}_a(k+1) = \tilde{\omega}_a(k)$, we find $\triangle L_1(k) = \triangle L_2(k) = 0$

$$\triangle L_3(k) = \frac{1}{2}[\|\xi_c(k)\|^2 - \|\xi_c(k-1)\|^2] \tag{28}$$

$$\begin{aligned}\triangle L_4(k) &= V(x(k+1)) - V(x(k)) = -r(k)\\ &= -x^T(k)Qx(k) - u^T(x(\delta_k))Ru(x(\delta_k))\\ &= -x^T(k)Qx(k)\\ &\quad -\{-[u^*(x(k)) - u(x(\delta_k))] + u^*(x(k))\}^T R\\ &\quad \times\{-[u^*(x(k)) - u(x(\delta_k))] + u^*(x(k))\}\\ &= -x^T(k)Qx(k) - u^{*T}(x(k))Ru^*(x(k))\\ &\quad -[u^*(x(k)) - u(x(\delta_k))]^T R[u^*(x(k)) - u(x(\delta_k))]\\ &\quad +2u^{*T}(x(k))R[u^*(x(k)) - u(x(\delta_k))].\end{aligned} \tag{29}$$

From (22), we have

$$\begin{aligned}u^*(x(k)) - u(x(\delta_k)) &= \omega_a^{*T}(k)\phi_a(k) + \epsilon_a - \hat{\omega}_a^T(k)\phi_a(x(\delta_k))\\ &= \hat{\omega}_a^T(k)[\phi_a(x(k)) - \phi_a(x(\delta_k))]\\ &\quad -\xi_a(k) + \epsilon_a.\end{aligned} \tag{30}$$

Substituting (24) and (30) into (29), we obtain

$$\begin{aligned}\triangle L_4(k) &\leqslant -x^T(k)Qx(k) + \lambda_{\max}(R)\|u^*(x(k))\|^2\\ &\quad +\lambda_{\max}(R)\big[\hat{\omega}_a^T(k)[\phi_a(x(k)) - \phi_a(x(\delta_k))]\\ &\quad -\tilde{\omega}_a^T(k)\phi_a(x(k)) + \epsilon_a\big]^2\\ &\leqslant -\lambda_{\min}(Q)\|x(k)\|^2 + 2\lambda_{\max}(R)\|\hat{\omega}_a\|^2 L_a^2\|e(k)\|^2\\ &\quad +6\lambda_{\max}(R)\big(\omega_{am}^2\phi_{am}^2 + \epsilon_{am}^2\big).\end{aligned} \tag{31}$$

Recall (25), from (28) and (31), we have

$$\triangle L(k) \leqslant -(1-\beta)\lambda_{\min}(Q)\|x(k)\|^2 + D_{1m}^2 \tag{32}$$

where

$$D_{1m}^2 = 6\lambda_{\max}(R)\big(\omega_{am}^2\phi_{am}^2 + \epsilon_{am}^2\big) + 2\omega_{cm}^2\phi_{cm}^2. \tag{33}$$

If

$$\|x(k)\| > \frac{D_{1m}}{\sqrt{(1-\beta)\lambda_{\min}(Q)}} \tag{34}$$

the first difference $\triangle L(k) \leq 0$. This demonstrates that the closed-loop system state and the weight estimation errors are UUB in this case.

Case 2 (A learning event is observed at time instant k with an event index δ_k): In this case, the weights of the critic network and the action network will be updated using (14) and (17). With the same Lyapunov function candidate (27) and a similar derivation as [41, Th.4.3], we have

$$\begin{aligned}\triangle L(k) &\leqslant -\left(\frac{1}{2} - \frac{4}{\gamma}\right)\|\xi_c(k)\|^2 - \lambda_{\min}(Q)\|x(k)\|^2 + D_{2m}^2\\ D_{2m}^2 &= \left(12 + \frac{4}{\gamma}\right)\omega_{cm}^2\phi_{cm}^2 + \frac{4}{\gamma}\omega_{cm}^2 C_m^2\omega_{am}^2\phi_{am}^2 + 8r_m^2\end{aligned} \tag{35}$$

where C_m and r_m are the upper bounds of $C(k)$ and $r(k)$ in (18) and (7), respectively, and $\gamma > 8$ is a weighting factor. If

$$\|x(k)\| > \frac{D_{2m}}{\sqrt{\lambda_{\min}(Q)}} \quad \text{or} \quad \|\xi_c(k)\| > \frac{D_{2m}}{\sqrt{\left(\frac{1}{2} - \frac{4}{\gamma}\right)}} \tag{36}$$

the first difference $\triangle L(k) \leq 0$.

Summarizing both cases we conclude that the closed-loop system state and the errors between the optimal network weights ω_c^*, ω_a^* and their respective estimates $\hat{\omega}_c(k)$ and $\hat{\omega}_a(k)$ are UUB. In this way, $\hat{\omega}_a(k+1)$ and the designed controller is stabilizing. This completes the proof. ■

Remark 1: Theorem 1 presents a sufficient condition to guarantee the stability of an event-driven dHDP-controlled system with learning realized in the critic and action neural networks. Without any special constraints on the nonlinear system, we have obtained a system boundedness result under a few mild assumptions. However, notice that even though the actor and critic neural networks are universally approximating according to the universal approximation theorem, it is unavoidable that the neural network weights are locally convergent and thus the designed controller is locally stabilizing. As there exist unavoidable approximation errors in both the cost-to-go function and the input control, Theorem 2 below examines how an approximate solution of the Bellman optimality equation can be achieved within a finite approximation error.

Theorem 2: Under the same conditions as in Theorem 1, the Bellman optimality equation is approximated within a finite approximation error. Meanwhile, the adopted control law $\hat{u}(k)$ is uniformly convergent to a finite neighborhood of the optimal control $u^*(k)$.

Proof: From the approximate cost-to-go in (10) for the Bellman equation (8) and the optimal cost-to-go expressed in approximation form (21) for the Bellman optimality (9), we have

$$\begin{aligned}\|\hat{V}(x(k)) - V^*(x(k))\| &= \big\|\hat{\omega}_c^T(k)\phi_c(k) - \omega_c^{*T}(k)\phi_c(k) - \epsilon_c\big\|\\ &\leqslant \|\tilde{\omega}_c(k)\|\phi_{cm} + \epsilon_c \leqslant \hat{\epsilon}_c.\end{aligned} \tag{37}$$

Similarly, from (11) and (22), we have

$$\|\hat{u}(k) - u^*(k)\| \leqslant \|\tilde{\omega}_a(k)\|\phi_{am} + \epsilon_a \leqslant \hat{\epsilon}_a. \tag{38}$$

This comes directly as $\|\tilde{\omega}_c(k)\|$ and $\|\tilde{\omega}_a(k)\|$ are both UUB as shown in Theorem 1. This demonstrates that the Bellman optimality is achieved within finite approximation errors. ■

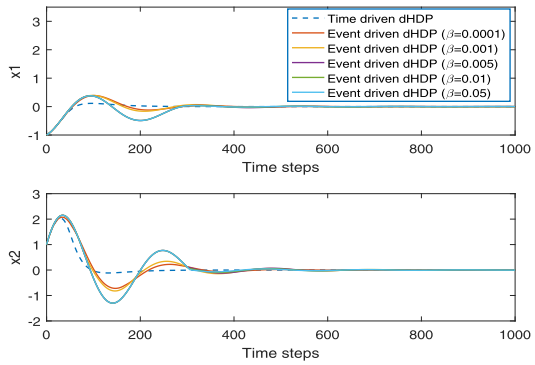

Fig. 2. State trajectories with event-driven dHDP controller (different β) and time-driven dHDP controller ($\beta = 0$).

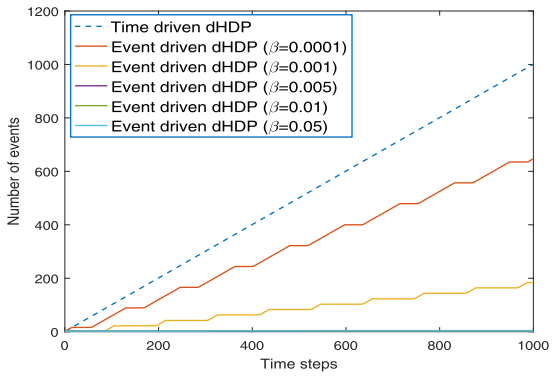

Fig. 3. Numbers of learning events of event-driven dHDP (different β) and time-driven dHDP ($\beta = 0$).

IV. Illustrative Example

In this section, we use a numerical example to demonstrate the theoretical results in this work and differences between event-driven dHDP and its time-driven counterpart. The unknown system dynamics are assumed to be generated from the following equation:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9996x_1(k) + 0.0099x_2(k) \\ -0.0887x_1(k) + 0.99x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1u(k) \end{bmatrix}.$$

Now we try to stabilize this system by the proposed event-driven dHDP algorithm. For both the critic neural network and the action neural network, we set the number of hidden layer nodes to $N_{hc} = N_{ha} = 6$. From Theorem 1, the learning rates of these two neural networks should satisfy (26). In this simulation, we choose $l_a = l_c = 0.1$. The initial condition of the system state is set as $x(0) = [-1, 1]^T$. The RL signal is $r(k) = x^T(k)I_m x(k) + u^T(k)(0.1I_n)u(k)$, where I_m and I_n are identity matrices. The initial weights of both the critic and the action neural network are set randomly within $[-0.4, 0.4]$. For the learning event condition (25), we choose different β values where $\beta = 0$ corresponds to time-driven dHDP as in [32].

Dynamic system state trajectories under different event-driven learning conditions (i.e., different β values) are shown in Fig. 2. As expected, little difference is noticed between the event-driven dHDP and the time-driven dHDP when β is small. Also as expected, the controlled system state trajectories deviate further from those controlled by the time-driven dHDP as β increases. Fig. 3 illustrates reduced numbers of learning events and consequently degraded learning control responses as β increases.

Next, we illustrate how event-driven dHDP may be used as an effective tool to prevent converged critic and action network weights from drifting too far from the learned controller due to inevitable noise in the state measurements.

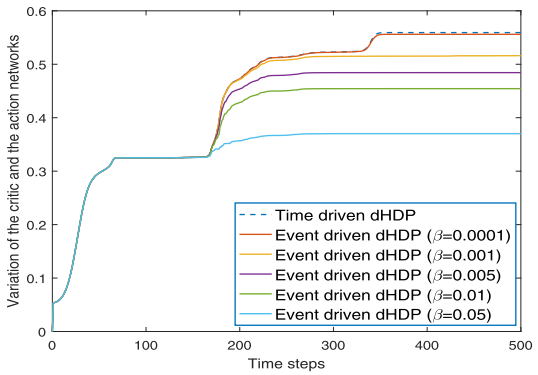

Fig. 4. Average accumulated variation for every weight under event-driven dHDP controller (different β) and time-driven dHDP controller ($\beta = 0$) with external noise.

This simulation entails 500 time steps, the first 100 of which are for training the critic and action weights. During this period, the weights are updated at each time instant k while the system dynamics are not subject to any noise. From time step 101 to 300, we introduce random Gaussian white noise $\omega_k \sim N(\omega_0, 1)$ with $\omega_0 = 0.1$ into the system. Then from time step 301 to 500, the random Gaussian white noise is removed or the system dynamics evolve under noise free condition. We use event-driven dHDP with different β values and consider the accumulated amount of variation in the critic and action network weights as defined below:

$$\eta(k) = \frac{\sum_{i=a,c}\sum_{j=0}^{k}\|\hat{\omega}_i(j+1) - \hat{\omega}_i(j)\|^2}{N_{hc} + N_{ha}} \tag{39}$$

where $\|\cdot\|$ denotes the 2-norm.

The first 100 time steps in Fig. 4 shows actual learning in the critic and action networks and such learning converges at about time step 80. From steps 101 to 300, we notice from small to large weight variations corresponding to small to large β values due to the added white Gaussian noise, and such variation is the largest when $\beta = 0$ or when learning takes place continuously. Therefore, we can see how learning event condition (25) helps protect the critic and action networks from drifting away due to random fluctuation in system dynamics.

V. Conclusion

We have proposed a new event-driven RL control method, that is event-driven dHDP based on the time-driven dHDP for a general discrete time nonlinear system. Specifically, we introduced a learning event criterion that is directly related to system states such that a learned dHDP controller could be less affected by random fluctuations in the system dynamics. We proved the stability of the closed-loop system under event-driven dHDP control using Lyapunov functions as well as the approximate optimality of the dHDP control solution. We demonstrated a significant reduction of weight updating times for event-driven learning from time-driven learning without sacrificing too much system performance.

References

[1] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychol. Learn. Motivat.*, vol. 24, pp. 109–165, Dec. 1989.

[2] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions," *Psychol. Rev.*, vol. 97, no. 2, pp. 285–308, 1990.

[3] G. Carpenter and S. Grossberg, "Adaptive resonance theory: Stable self-organization of neural recognition codes in response to arbitrary lists of input patterns," in *Proc. 8th Annu. Conf. Cogn. Sci. Soc.*, 1986, pp. 45–62.

[4] G. Hinton, J. McClelland, and D. Rumelhart, *Distributed Representations*. Pittsburgh, PA, USA: Carnegie Mellon Univ., 1984.

[5] G. Hinton and D. Plaut, "Using fast weights to deblur old memories," in *Proc. 9th Annu. Conf. Cognit. Sci. Soc.*, 1987, pp. 177–186.

[6] R. Sutton, "Two problems with back propagation and other steepest descent learning procedures for networks," in *Proc. 8th Annu. Conf. Cognit. Sci. Soc.*, 1986, pp. 823–831.

[7] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[8] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3987–3995.

[9] K. James *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[10] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, Sep. 1992.

[11] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.

[12] X. Zhong and H. He, "An event-triggered ADP control approach for continuous-time system with unknown internal states," *IEEE Trans. Cybern.*, vol. 47, no. 3, pp. 683–694, Mar. 2017.

[13] X. Yang, H. He, and D. Liu, "Event-triggered optimal neuro-controller design with reinforcement learning for unknown nonlinear systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 9, pp. 1866–1878, Sep. 2019.

[14] L. Dong, X. Zhong, C. Sun, and H. He, "Event-triggered adaptive dynamic programming for continuous-time systems with control constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1941–1952, Aug. 2017.

[15] Y. Zhu, D. Zhao, H. He, and J. Ji, "Event-triggered optimal control for partially unknown constrained-input systems via adaptive dynamic programming," *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4101–4109, May 2017.

[16] H. Zhang, K. Zhang, G. Xiao, and H. Jiang, "Robust optimal control scheme for unknown constrained-input nonlinear systems via a plug-n-play event-sampled critic-only algorithm," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 9, pp. 3169–3180, Sep. 2020.

[17] Q. Zhang, D. Zhao, and D. Wang, "Event-based robust control for uncertain nonlinear systems using adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 37–50, Jan. 2018.

[18] D. Wang, C. Mu, H. He, and D. Liu, "Event-driven adaptive robust control of nonlinear systems with uncertainties through NDP strategy," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1358–1370, Jul. 2017.

[19] D. Wang, C. Mu, D. Liu, and H. Ma, "On mixed data and event driven design for adaptive-critic-based nonlinear H_∞ control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 993–1005, Apr. 2018.

[20] Q. Zhang, D. Zhao, and Y. Zhu, "Event-triggered H_∞ control for continuous-time nonlinear system via concurrent learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1071–1081, Jul. 2017.

[21] X. Yang and H. He, "Adaptive critic designs for event-triggered robust control of nonlinear systems with unknown dynamics," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 2255–2267, Jun. 2019.

[22] B. Luo, Y. Yang, D. Liu, and H.-N. Wu, "Event-triggered optimal control with performance guarantees using adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 1, pp. 76–88, Jan. 2020.

[23] A. Sahoo, H. Xu, and S. Jagannathan, "Near optimal event-triggered control of nonlinear discrete-time systems using neurodynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1801–1815, Sep. 2016.

[24] M. Ha, D. Wang, and D. Liu, "Event-triggered adaptive critic control design for discrete-time constrained nonlinear systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 9, pp. 3158–3168, Sep. 2020.

[25] L. Dong, X. Zhong, C. Sun, and H. He, "Adaptive event-triggered control based on heuristic dynamic programming for nonlinear discrete-time systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 7, pp. 1594–1605, Jul. 2017.

[26] W. Gao and Z.-P. Jiang, "Learning-based adaptive optimal tracking control of strict-feedback nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2614–2624, Jun. 2018.

[27] Y. Jiang and Z.-P. Jiang, "Global adaptive dynamic programming for continuous-time nonlinear systems," *IEEE Trans. Automat. Control*, vol. 60, no. 11, pp. 2917–2929, Nov. 2015.

[28] T. Bian, Y. Jiang, and Z.-P. Jiang, "Adaptive dynamic programming and optimal control of nonlinear nonaffine systems," *Automatica*, vol. 50, no. 10, pp. 2624–2632, Oct. 2014.

[29] Y. Jiang, J. Fan, T. Chai, and F. L. Lewis, "Dual-rate operational optimal control for flotation industrial process with unknown operational model," *IEEE Trans. Ind. Electron.*, vol. 66, no. 6, pp. 4587–4599, Jun. 2019.

[30] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 943–949, Aug. 2008.

[31] C. Mu, D. Wang, and H. He, "Novel iterative neural dynamic programming for data-based approximate optimal control design," *Automatica*, vol. 81, pp. 240–252, Jul. 2017.

[32] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, Mar. 2001.

[33] R. Enns and J. Si, "Helicopter trimming and tracking control using direct neural dynamic programming," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 929–939, Jul. 2003.

[34] R. Enns and J. Si, "Apache helicopter stabilization using neural dynamic programming," *J. Guid., Control, Dyn.*, vol. 25, no. 1, pp. 19–25, Jan. 2002.

[35] R. Enns and J. Si, "Helicopter flight-control reconfiguration for main rotor actuator failures," *J. Guid., Control, Dyn.*, vol. 26, no. 4, pp. 572–584, Jul. 2003.

[36] C. Lu, J. Si, and X. Xie, "Direct heuristic dynamic programming for damping oscillations in a large power system," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 1008–1013, Aug. 2008.

[37] Y. Wen, J. Si, X. Gao, S. Huang, and H. H. Huang, "A new powered lower limb prosthesis control framework based on adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 9, pp. 2215–2220, Sep. 2017.

[38] Y. Wen, J. Si, A. Brandt, X. Gao, and H. H. Huang, "Online reinforcement learning control for the personalization of a robotic knee prosthesis," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2346–2356, Jun. 2020.

[39] Y. Zhang, S. Li, K. Nolan, and D. Zanotto, "Adaptive assist-as-needed control based on actor-critic reinforcement learning," in *Proc. Int. Conf. IROS*, 2019, pp. 4066–4071.

[40] X. Gao, J. Si, Y. Wen, M. Li, and H. Huang, "Knowledge-guided reinforcement learning control for robotic lower limb prosthesis," in *Proc. Int. Conf. ICRA*, 2020, pp. 754–760.

[41] F. Liu, J. Sun, J. Si, W. Guo, and S. Mei, "A boundedness result for the direct heuristic dynamic programming," *Neural Netw.*, vol. 32, pp. 229–235, Aug. 2012.