Neurocomputing 78 (2012) 3–13

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A three-network architecture for on-line learning and optimization based on adaptive dynamic programming

Haibo He [a,*], Zhen Ni [a], Jian Fu [b]

[a] Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881, USA
[b] School of Automation, Wuhan University of Technology, Wuhan, Hubei 430070, China

ARTICLE INFO

Available online 25 August 2011

Keywords:
Adaptive dynamic programming
Online learning and control
Actor-critic design
Three-network architecture
Multi-state optimization
Goal representation
Reinforcement learning

ABSTRACT

In this paper, we propose a novel adaptive dynamic programming (ADP) architecture with three networks, an action network, a critic network, and a reference network, to develop internal goal-representation for online learning and optimization. Unlike the traditional ADP design normally with an action network and a critic network, our approach integrates the third network, a reference network, into the actor-critic design framework to automatically and adaptively build an internal reinforcement signal to facilitate learning and optimization overtime to accomplish goals. We present the detailed design architecture and its associated learning algorithm to explain how effective learning and optimization can be achieved in this new ADP architecture. Furthermore, we test the performance of our architecture both on the cart-pole balancing task and the triple-link inverted pendulum balancing task, which are the popular benchmarks in the community to demonstrate its learning and control performance over time.

1. Introduction

Learning and optimization has been a long-term focus in the machine intelligence community to understand and develop principled methodologies to replicate certain level of the brain-like general-purpose intelligence [1,2]. Over the past decades, extensive efforts have been focused on different aspects of machine intelligence research, including memory-prediction theory, embodied intelligence (EI), reinforcement learning (RL), neural dynamic programming, and many others. Recently, strong evidences from multiple disciplinary research have supported that mathematically speaking biological brain can be considered as a whole system of an intelligent controller to learn and predict to adjust actions to achieve goals [1,3]. To this end, it is widely recognized that adaptive dynamic programming (ADP) could be a core methodology to accomplish the learning and optimization capabilities for intelligent systems to approximate optimal strategy of actions over time. From the application side, ADP has also demonstrated many successful applications across a wide range of domains, such as intelligent power grid, complex system control, chess gaming, and many others [4–18].

Generally speaking, the foundation for optimization over time in stochastic processes is the Bellman equation [19], closely tied with the Cardinal utility function concept by Von Neumann. Specifically, given a system with performance cost:

$$J[\boldsymbol{x}(t),t]=\sum_{t=i}^{\infty}\gamma^{t-i}U[\boldsymbol{x}(t),u(t),t] \tag{1}$$

where $\boldsymbol{x}(t)$ is the state vector of the system, $u(t)$ is the control action, U is the utility function, and γ is a discount factor. The objective of dynamic programming is to design control sequence $u(t)$ so the cost function J is minimized:

$$J^*(\boldsymbol{x}(t))=\min_{u(t)}\{U(\boldsymbol{x}(t),u(t))+\gamma J^*(\boldsymbol{x}(t+1))\} \tag{2}$$

Eq. (2) provides the foundation for implementing dynamic programming by working backward in time. For instance, universal approximators like neural networks with the backpropagation method are widely used in the community [20,21].

Existing adaptive critic design can be categorized into three major groups [5,22,23]: heuristic dynamic programming (HDP), dual heuristic dynamic programming (DHP), and globalized dual heuristic dynamic programming (GDHP). For instance, the HDP [1] was proposed with the objective of using a critic network to critique the action value in order to optimize the future cost function by using temporal differences between two consecutive estimates from the critic network [24]. This idea is essentially similar to the temporal-difference (TD) method discussed in the RL literature [25]. To overcome the limitations of scalability, DHP and GDHP were proposed [26], followed by many improvements and demonstrations of such methods [4,6,27]. The key idea of

* Corresponding author.
E-mail addresses: he@ele.uri.edu (H. He), ni@ele.uri.edu (Z. Ni), pigeon1387@gmail.com (J. Fu).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.05.031

DHP is to use a critic network to approximate the derivatives of the value function with respect to the states, while GDHP takes advantage of both HDP and DHP by using a critic network to approximate both the value function and its derivatives. Variations of all of these categories of ADP design have also been investigated in the community, such as the action dependent (AD) version of the aforementioned methods by taking action values as an additional input to the critic network [24].

In this paper, we aim to tackle one of the critical questions in the ADP design: how to effectively and efficiently represent the reinforcement signal to guide the intelligent system to achieve goals over time? In our approach, we propose to integrate a reference network to provide the internal reinforcement signal as a natural representation of the internal goal to facilitate machine learning. This reference network will interact with the critic network and action network in the classic ADP design to provide improved learning and optimization performance. Traditionally, in the existing ADP design, the normal way is to use a binary signal, such as either a "0" or a "−1," to represent "success" or "failure" of the system. In order to provide more informative reinforcement signals in many complex problems, different approaches have been proposed to use non-binary reinforcement signals to improve the learning performance, such as a three-value reinforcement signal (0, −0.4, and −1) for a pendulum swing up and balancing control task [24] and a quadratic reinforcement signal for the helicopter flight control [28]. Instead of hand-crafting such reinforcement signals, our objective in this paper is to seek an approach that can automatically and adaptively develop informative internal reinforcement signals to guide the intelligent system's behavior to achieve on-line learning, optimization, and control. We hope our proposed approach in this paper can provide useful suggestions to this important question regarding how to develop internal goal representations for machine intelligence research.

The rest of this paper is organized as follows. Section 2 presents the detailed three-network ADP architecture design and associated learning algorithm. The focus in this section is the reference network and critic network design, as well as their interactions to facilitate learning and optimization over time. In Section 3, detailed experimental setup and simulation analysis based on the cart-pole benchmark is presented to show the effectiveness of our approach. Following this, we present another case study in Section 4 with triple-linked inverted pendulum balancing benchmark to further demonstrate how the proposed approach performs under such a task. For both case studies, we have provided detailed comparative study of our approach with that of existing literature method. Finally, a conclusion and brief discussion regarding future research directions are discussed in Section 5.

2. A three-network ADP architecture

2.1. System architecture with internal goal representation

Fig. 1 shows the proposed ADP architecture with goal representation for learning and optimization over time. Compared to the existing ADP architectures, the key idea of our approach is to integrate another network, the *reference network*, to provide the internal reinforcement signal (internal goal representation) $s(t)$, to interact with the operation of the critic network. By introducing such a reference network to represent the system's internal goal, this architecture provides a new way to adaptively estimate the internal reinforcement signal instead of crafted by hand. This is the most important contribution of this work when compared to the existing ADP designs. From a mathematical point of view, this new architecture presents two major differences compared with that of the existing ADP designs. First, the critic network has one more additional input $s(t)$ from the reference network. Second, the optimization error function and learning in the reference network and critic network are different: The error function of the reference network is related to the primary reinforcement

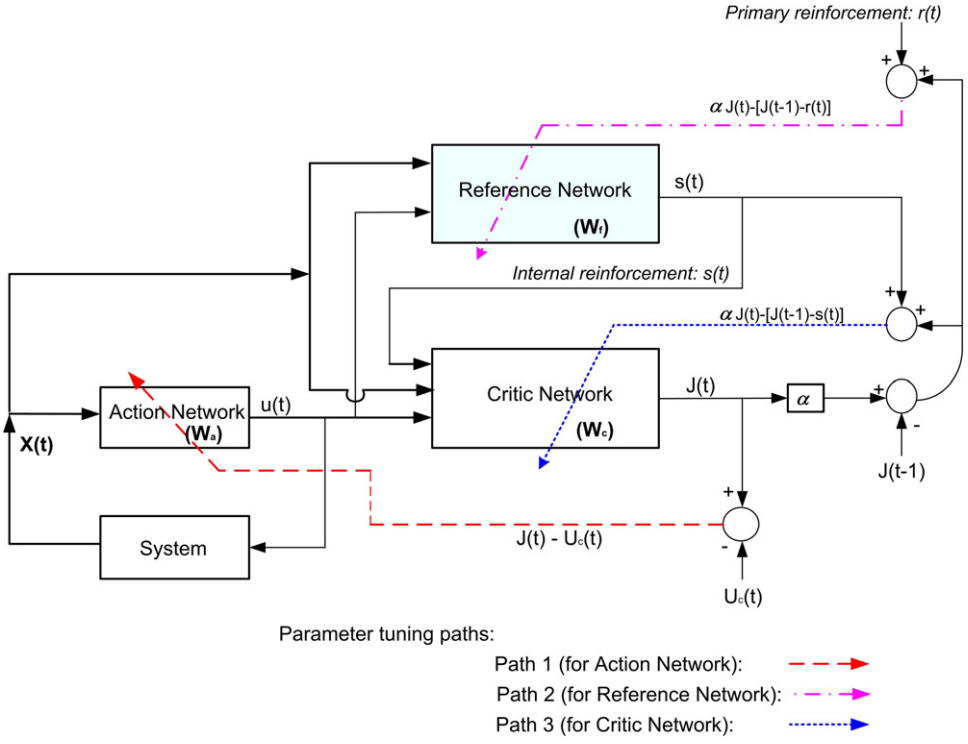


Fig. 1. The proposed ADP architecture with internal goal representation.

signal $r(t)$, whereas for the critic network, it is related to the internal reinforcement signal $s(t)$. Both of these characteristics will change the parameter tuning and adaptation in such a network, which we will discuss in further detail in this section. We would also like to note that another characteristic of our method is that it shares the advantage of no requirement of a system model to predict the future system state, as proposed in the ADP architecture in [24]. This means similar to the architecture in [24], our approach in this work also stores the previous cost-to-go value to obtain the temporal difference for training at any time instance, which enables the online learning, association, and optimization over time.

2.2. Learning and adaptation in the proposed architecture

From Fig. 1 one can see there are three paths to tune the parameters of the three types of networks. The action network in this architecture is similar to the classic ADP approach to indirectly backpropagate the error between the desired ultimate objective U_c and the J function from the critic network [24,29]. Therefore, the error function $E_a(t)$ used to update the parameters in the action network can be defined as (path 1 in Fig. 1):

$$e_a(t)=J(t)-U_c(t); \quad E_a(t)=\tfrac{1}{2}e_a^2(t) \tag{3}$$

The key of this architecture relies on the learning and adapting process for the reference network and critic network. As the primary reinforcement signal $r(t)$ is presented to the reference network, the secondary reinforcement signal $s(t)$ is adapted to provide a more informative internal reinforcement representation to the critic network, which in turn is used to provide a better approximation of the $J(t)$. In this way, the primary reinforcement signal $r(t)$ is in a higher hierarchical level and can be a simple binary signal to represent "good" or "bad", or "success" or "failure", while the secondary reinforcement signal $s(t)$ can be a more informative continuous values for improved learning and generalization performance. Therefore, the error function $E_f(t)$ used to update the parameters in the reference network can be defined as (path 2 in Fig. 1):

$$e_f(t)=\alpha J(t)-[J(t-1)-r(t)]; \quad E_f(t)=\tfrac{1}{2}e_f^2(t) \tag{4}$$

Once the reference network outputs the $s(t)$ signal, it will be used as an input to the critic network, and also used to define the error function to adjust the parameters of the critic network (path 3 in Fig. 1):

$$e_c(t)=\alpha J(t)-[J(t-1)-s(t)]; \quad E_c(t)=\tfrac{1}{2}e_c^2(t) \tag{5}$$

In this architecture, the chain backpropagation rule is the key to training and adaptation of the parameters of all three networks (action network, critic network, and reference network) [20]. Fig. 2 shows the three backpropagation paths used to adapt the parameters in the three networks.

In this figure, the optimization error functions for the action network E_a, reference network E_f, and critic network E_c are defined in Eqs. (3)–(5), respectively. Therefore, chain backpropagation can be calculated through the three data paths as highlighted in Fig. 2. Briefly speaking, the high level conceptual calculation on this can be summarized as follows:

Path 1: For action network:

$$\frac{\partial E_a(t)}{\partial w_a(t)}=\frac{\partial E_a(t)}{\partial J(t)}\frac{\partial J(t)}{\partial u(t)}\frac{\partial u(t)}{\partial w_a(t)} \tag{6}$$

Path 2: For reference network:

$$\frac{\partial E_f(t)}{\partial w_f(t)}=\frac{\partial E_f(t)}{\partial J(t)}\frac{\partial J(t)}{\partial s(t)}\frac{\partial s(t)}{\partial w_f(t)} \tag{7}$$

Path 3: For critic network:

$$\frac{\partial E_c(t)}{\partial w_c(t)}=\frac{\partial E_c(t)}{\partial J(t)}\frac{\partial J(t)}{\partial w_c(t)} \tag{8}$$

2.3. Adaptation and learning in the three networks

In order to focus on the learning principle, in this paper we assume neural networks with a three-layer nonlinear architecture in all three networks. Meanwhile, since the weight tuning process in the action network is similar to that as presented in [24], in this paper we will only focus on the adaptation and learning in the reference network and critic network for clear presentation. Interested reader can refer to [24] for the detailed learning process in the action network. We would also like to note that the learning principles discussed in this section can also be generalized to any arbitrary function approximators by properly applying backpropagation rule, which will be reported in future research studies. We now proceed to discuss the detailed design strategy and learning process in the proposed ADP architecture.

2.3.1. Reference network learning and adaptation

Fig. 3(a) shows the reference network used in this design with a three-layer nonlinear architecture (with one hidden layer). To calculate the backpropagation, we first need to define the

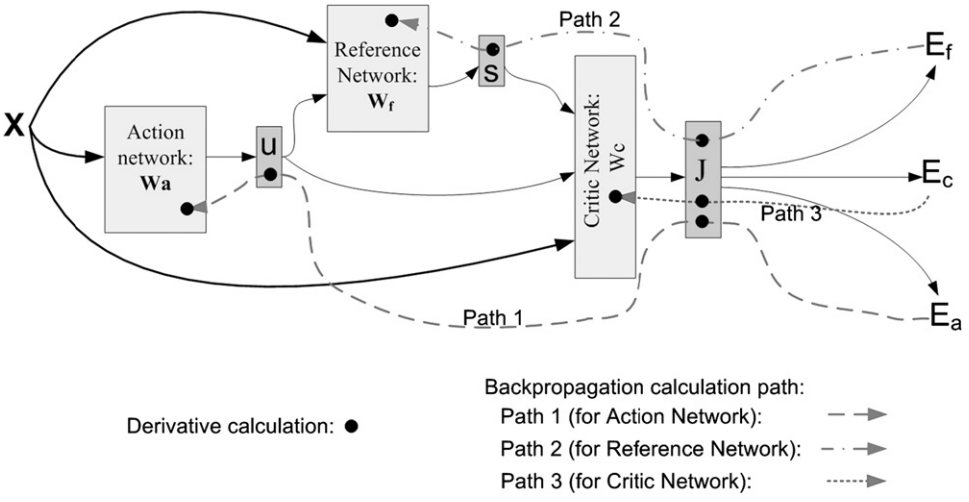


Fig. 2. Parameters adaptation and tuning based on backpropagation.

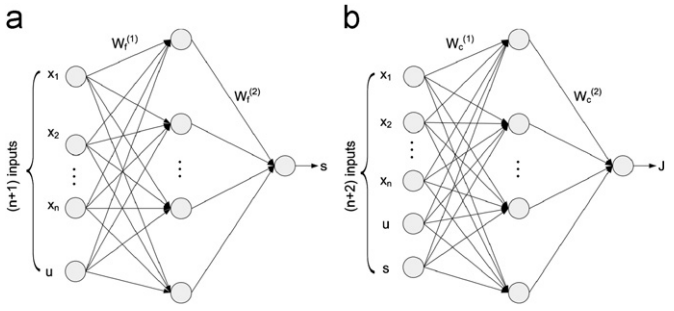


Fig. 3. Design of the three networks with nonlinear neural network: (a) Reference network design; (b) critic network design.

reference network output $s(t)$ as follows:

$$s(t)=\frac{1-\exp^{-k(t)}}{1+\exp^{-k(t)}} \tag{9}$$

$$k(t)=\sum_{i=1}^{N_h} w_{f_i}^{(2)}(t)y_i(t) \tag{10}$$

$$y_i(t)=\frac{1-\exp^{-z_i(t)}}{1+\exp^{-z_i(t)}},\quad i=1,\ldots,N_h \tag{11}$$

$$z_i(t)=\sum_{j=1}^{n+1} w_{f_{i,j}}^{(1)}(t)x_j(t),\quad i=1,\ldots,N_h \tag{12}$$

where z_i is the ith hidden node input of the reference network and y_i is the corresponding output of the hidden node, k is the input to the output node of the reference network before the sigmoid function, N_h is the number of hidden neurons of the reference network, and $(n+1)$ is the total number of inputs to the reference network including the action value $u(t)$ from the action network.

To apply the backpropagation rule, one can refer to Fig. 2 and Eqs. (4) and (7). Specifically, since the output $s(t)$ is an input to the critic network, backpropagation can be applied here through the chain rule (path 2) to adapt the parameters W_f. This procedure is illustrated as follows.

(1) $\Delta w_f^{(2)}$: Reference network weight adjustment for the hidden to the output layer:

$$\Delta w_{f_i}^{(2)}=\eta_f(t)\left[-\frac{\partial E_f(t)}{\partial w_{f_i}^{(2)}(t)}\right] \tag{13}$$

$$\begin{aligned}\frac{\partial E_f(t)}{\partial w_{f_i}^{(2)}(t)}&=\frac{\partial E_f(t)}{\partial J(t)}\frac{\partial J(t)}{\partial s(t)}\frac{\partial s(t)}{\partial k(t)}\frac{\partial k(t)}{\partial w_{f_i}^{(2)}(t)}\\&=\alpha e_f(t)\cdot\sum_{i=1}^{N_h}\left[w_{c_i}^{(2)}(t)\frac{1}{2}(1-p_i^2(t))w_{c_{i,n+2}}^{(1)}(t)\right]\\&\quad\cdot\frac{1}{2}(1-(s(t))^2)\cdot y_i(t)\end{aligned} \tag{14}$$

(2) $\Delta w_f^{(1)}$: Reference network weight adjustments for the input to the hidden layer:

$$\Delta w_{f_{i,j}}^{(1)}=\eta_f(t)\left[-\frac{\partial E_f(t)}{\partial w_{f_{i,j}}^{(1)}(t)}\right] \tag{15}$$

$$\begin{aligned}\frac{\partial E_f(t)}{\partial w_{f_{i,j}}^{(1)}(t)}&=\frac{\partial E_f(t)}{\partial J(t)}\frac{\partial J(t)}{\partial s(t)}\frac{\partial s(t)}{\partial k(t)}\frac{\partial k(t)}{\partial y_i(t)}\frac{\partial y_i(t)}{\partial z_i(t)}\frac{\partial z_i(t)}{\partial w_{f_{i,\ j}}^{(1)}(t)}\\&=\alpha e_f(t)\cdot\sum_{i=1}^{N_h}\left[w_{c_i}^{(2)}(t)\frac{1}{2}(1-p_i^2(t))w_{c_{i,n+2}}^{(1)}(t)\right]\\&\quad\cdot\frac{1}{2}(1-(s(t))^2)\cdot w_{f_i}^{(2)}(t)\cdot\frac{1}{2}(1-y_i^2(t))\cdot x_j(t)\end{aligned} \tag{16}$$

Once the reference network provides the secondary reinforcement signal $s(t)$ to the critic network, one can adapt the parameters in the critic network.

2.3.2. Critic network learning and adaptation

Fig. 3(b) shows the critic network used in our current design with a three-layer nonlinear architecture (with one hidden layer). To calculate the backpropagation, we first need to define the critic network output $J(t)$ as follows:

$$J(t)=\sum_{i=1}^{N_h} w_{c_i}^{(2)}(t)p_i(t) \tag{17}$$

$$p_i(t)=\frac{1-\exp^{-q_i(t)}}{1+\exp^{-q_i(t)}},\quad i=1,\ldots,N_h \tag{18}$$

$$q_i(t)=\sum_{j=1}^{n+2} w_{c_{i,j}}^{(1)}(t)x_j(t),\quad i=1,\ldots,N_h \tag{19}$$

where q_i and p_i are the input and output of the ith hidden node of the critic network, respectively, and $(n+2)$ is the total number of inputs to the critic network including the action value $u(t)$ from the action network and the secondary reinforcement signal $s(t)$ from the reference network.

By applying chain backpropagation rule (path 3), the procedure of adapting parameters in the critic network is summarized as follows.

(1) $\Delta w_c^{(2)}$: Critic network weight adjustments for the hidden to the output layer:

$$\Delta w_{c_i}^{(2)}=\eta_c(t)\left[-\frac{\partial E_c(t)}{\partial w_{c_i}^{(2)}(t)}\right] \tag{20}$$

$$\frac{\partial E_c(t)}{\partial w_{c_i}^{(2)}(t)}=\frac{\partial E_c(t)}{\partial J(t)}\frac{\partial J(t)}{\partial w_{c_i}^{(2)}(t)}=\alpha e_c(t)\cdot p_i(t) \tag{21}$$

(2) $\Delta w_c^{(1)}$: Critic network weight adjustments for the input to the hidden layer:

$$\Delta w_{c_{i,j}}^{(1)}=\eta_c(t)\left[-\frac{\partial E_c(t)}{\partial w_{c_{i,j}}^{(1)}(t)}\right] \tag{22}$$

$$\begin{aligned}\frac{\partial E_c(t)}{\partial w_{c_{i,j}}^{(1)}(t)}&=\frac{\partial E_c(t)}{\partial J(t)}\frac{\partial J(t)}{\partial p_i(t)}\frac{\partial p_i(t)}{\partial q_i(t)}\frac{\partial q_i(t)}{\partial w_{c_{i,j}}^{(1)}(t)}\\&=\alpha e_c(t)\cdot w_{c_i}^{(2)}(t)\cdot\frac{1}{2}(1-p_i^2(t))\cdot x_j(t)\end{aligned} \tag{23}$$

3. Simulation analysis I: case study with cart-pole balancing benchmark

3.1. Description of the system model

The proposed three-network architecture has been implemented on a cart-pole balancing problem, which is the same as that in [24]. Our ultimate goal here is to control the force applied on the cart to move it either left or right to keep the balance of the single pole mounted on the cart. The system function of the model is described as following:

$$\partial^2\theta/\partial t^2=\frac{g\sin\theta+\frac{\cos\theta[-F-ml\dot\theta^2\sin\theta+\mu_c\,\mathrm{sgn}(\dot x)]}{m_c+m}-\frac{\mu_p\dot\theta}{ml}}{l\left(\frac{4}{3}-\frac{m\cos^2\theta}{m_c+m}\right)} \tag{24}$$

$$\partial^2 x/\partial t^2=\frac{F+ml[\dot\theta^2\sin\theta-\ddot\theta\cos\theta]-\mu_c\,\mathrm{sgn}(\dot x)}{m_c+m} \tag{25}$$

where the acceleration $g=9.8$ m/s^2, the mass of the cart $m_c=1.0$ kg, the mass of the pole $m=0.1$ kg, half-pole length $l=0.5$ m, the coefficient of friction of the cart $\mu=0.0005$ and the coefficient of friction of the pole $\mu_p=0.000002$. The force F applied to the cart is either 10 N or -10 N, and the sgn function in Eq. (25) is defined as following:

$$\mathrm{sgn}(x)=\begin{cases}1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases} \tag{26}$$

The state vector in this system model is

$$Q=[x\ \ \theta\ \ \dot{x}\ \ \dot{\theta}] \tag{27}$$

In our current study, we adopted the same criteria as those in [24] to evaluate the performance of our control approach. That is to say a pole is considered fallen when the angular is outside the range of $[-12^\circ, 12^\circ]$ or the cart if beyond the range of $[-2.4$ m, 2.4 m]. Note that in our current study, the F force applied to the cart is a binary value (i.e., either 10 N or -10 N) while the control action $u(t)$ fed to the critic network is a continuous value.

3.2. Simulation results

In order to evaluate the statistic performance of our proposed approach, we set 100 runs to this task with different initial state conditions. Specifically, the angular and angular velocity of the pole in each of these initial states are uniformly generated within $[-0.1^\circ, 0.1^\circ]$ and $[-0.5,0.5]180/pi$ rad/s, respectively, while the position and velocity of the cart are both 0. We hope these different initial conditions will provide a comprehensive understanding of our approach.

The parameters in our simulation are summarized in Table 1 and the notations are defined as following:

$l_c(0)$: initial learning rate of the critic network;
$l_a(0)$: initial learning rate of the action network;
$l_r(0)$: initial learning rate of the reference network;
$l_c(k)$: learning rate of the critic network which is decreased by 0.05 every five time steps until it reach $l_c(f)$ and stay thereafter;
$l_a(k)$: learning rate of the action network which is decreased by 0.05 every five time steps until it reach $l_a(f)$ and stay thereafter;
$l_r(k)$: learning rate of the reference network which is decreased by 0.05 every five time steps until it reach $l_r(f)$ and stay thereafter;
N_c: internal cycle of the critic network;
N_a: internal cycle of the action network;
N_r: internal cycle of the reference network;
T_c: internal training error threshold for the critic network;
T_a: internal training error threshold for the action network;
T_r: internal training error threshold for the reference network.

For comparative study, here we compare our results with those of ADP structure presented in [24] with the same parameter setting. These results are summarized in Table 2. For fair comparison, we also added the same type of noises according to [24] in our simulation. From these results one can see, our approach can provide competitive results, especially under the uniform noise and Gaussian noise on sensors. We would also like to note that under noise free condition and uniform noise on actuator (both 5% and 10%), the approach in [24] can provide better results. This might indicate our approach is more robust and can work effectively under relatively large level of noises. We are currently conducting large-scale experiments with different benchmarks under various types of noises to investigate this issue, and corresponding results will be presented later.

Table 1
Summary of the parameters used in cart-pole balancing task.

Para.	$l_c(0)$	$l_a(0)$	$l_r(0)$	$l_c(f)$	$l_a(f)$	$l_r(f)$	
Value	0.3	0.3	0.3	0.001	0.001	0.001	
Para.	N_c	N_a	N_r	T_c	T_a	T_r	α
Value	80	100	50	0.05	0.005	0.05	0.95

Table 2
Performance evaluation on case I: cart-pole balancing task. The 2nd and the 3rd columns are with our proposed method, while the 4th and the 5th columns are the results from [24].

Noise type	Success rate (%)	# of trial	Success rate (%)	# of trial
Noise free	100	13.7	100	6
Uniform 5% a.[a]	100	16.1	100	8
Uniform 10% a.	100	20.6	100	14
Uniform 5% s.[b]	100	12.6	100	32
Uniform 10% s.	100	14.4	100	54
Gaussian $\sigma^2(0.1)$ s.	100	15.0	100	164
Gaussian $\sigma^2(0.2)$ s.	100	21.3	100	193

[a] a.: actuators are subject to the noise.
[b] s.: sensors are subject to the noise.

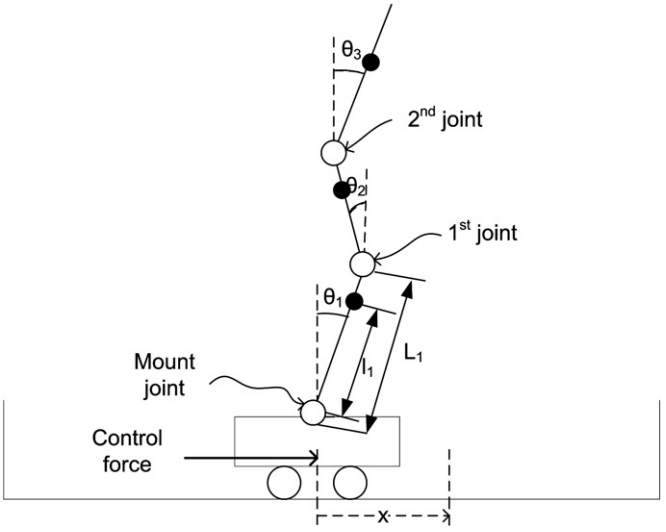


Fig. 4. Definition of notation used in the system equations for the triple link inverted pendulum benchmark.

4. Simulation analysis II: case study with triple-link inverted pendulum balancing benchmark

4.1. Description of the system model

The three-network approach mentioned above is now tested on triple-link inverted pendulum, which is unstable with multi-variables and exhibits non-negligible nonlinearities. This kind of pendulum is frequently used to evaluate the performance of new control strategies. Here we consider the same system model as in [24,30]. Fig. 4 shows a schematic diagram depicting the notations used.

Table 3
System constants for the triple link inverted pendulum problem.

Constant	Value	Constant	Value
A_1	$M+m_1+m_2+m_3$	A_2	$m_1l_1+(m_2+m_3)L_1$
A_3	$m_2l_2+m_3L_2$	A_4	m_3l_3
A_5	C_c	A_6	$-m_1l_1-(m_2+m_3)L_1$
A_7	$-(m_2l_2+m_3L_2)$	A_8	$-m_3l_3$
A_9	$m_1l_1+(m_2+m_3)L_1$	A_{10}	$I_1+m_2l_1^2+(m_2+m_3)L_1^2$
A_{11}	$(m_2l_2+m_3L_2)L_1$	A_{12}	$m_3l_3L_1$
A_{13}	C_1+C_2	A_{14}	$(m_2l_2+m_3L_2)L1$
A_{15}	$-C_2$	A_{16}	$m_3l_3L_1$
A_{17}	$-g(m_1l_1+m_2L_1+m_3L_1)$	A_{18}	$m_2l_2+m_3L_2$
A_{19}	$(m_2l_2+m_3L_2)L_1$	A_{20}	$I_2+m_3L_2^2+m_2l_2^2$
A_{21}	$m_3l_3L_2$	A_{22}	$-(m_2l_2+m_3L_2)L_1$
A_{23}	$-C_2$	A_{24}	C_2+C_3
A_{25}	$m_3l_3L_2$	A_{26}	$-C_3$
A_{27}	$-g(m_2l_2+m_3L_2)$	A_{28}	m_3l_3
A_{29}	$m_3l_3L_1$	A_{30}	$m_3l_3L_2$
A_{31}	$I_3+m_3l_3^2$	A_{32}	C_3
A_{33}	$-m_3l_3L_1$	A_{34}	$-gm_3l_3$
A_{35}	$-m_3l_3L_2$	A_{36}	$-C_3$
A_{37}	1.3	A_{38}	0.506
A_{39}	0.219	A_{40}	0.568

The nonlinear dynamic equations of this system can be expressed as

$$F(q)\frac{d^2q}{dt^2}=-G\left(q,\frac{dq}{dt}\right)\frac{dq}{dt}-H(q)+L(q,u) \tag{28}$$

where

$$F(q)=\begin{bmatrix} \mathrm{A}_1 & \mathrm{A}_2\cos(\theta_1) & \mathrm{A}_3\cos(\theta_2) & \mathrm{A}_4\cos(\theta_3)\\ \mathrm{A}_9\cos(\theta_1) & \mathrm{A}_{10} & \mathrm{A}_{11}\cos(\theta_1-\theta_2) & \mathrm{A}_{12}\cos(\theta_1-\theta_3)\\ \mathrm{A}_{18}\cos(\theta_2) & \mathrm{A}_{19}\cos(\theta_1-\theta_2) & \mathrm{A}_{20} & \mathrm{A}_{21}\cos(\theta_2-\theta_3)\\ \mathrm{A}_{28}\cos(\theta_3) & \mathrm{A}_{29}\cos(\theta_1-\theta_3) & \mathrm{A}_{30}\cos(\theta_2-\theta_3) & \mathrm{A}_{31} \end{bmatrix} \tag{29}$$

$$G\left(q,\frac{dq}{dt}\right)=\begin{bmatrix} \mathrm{A}_5 & \mathrm{A}_6\sin(\theta_1)\dot{\theta_1} & \mathrm{A}_7\sin(\theta_2)\dot{\theta_2} & \mathrm{A}_8\sin(\theta_3)\dot{\theta_3}\\ 0 & \mathrm{A}_{13} & \mathrm{A}_{14}\sin(\theta_1-\theta_2)\dot{\theta_2}+\mathrm{A}_{15} & \mathrm{A}_{16}\sin(\theta_1-\theta_3)\dot{\theta_3}\\ 0 & \mathrm{A}_{22}\sin(\theta_1-\theta_2)\dot{\theta_2}+\mathrm{A}_{23} & \mathrm{A}_{24} & \mathrm{A}_{25}\sin(\theta_2-\theta_3)\dot{\theta}_3+\mathrm{A}_{26}\\ 0 & \mathrm{A}_{33}\sin(\theta_1-\theta_3)\dot{\theta_1} & \mathrm{A}_{35}\sin(\theta_2-\theta_3)\dot{\theta_2}+\mathrm{A}_{36} & \mathrm{A}_{32} \end{bmatrix} \tag{30}$$

$$q=\begin{bmatrix} x\\ \theta_1\\ \theta_2\\ \theta_3 \end{bmatrix} \tag{31}$$

$$H(q)=\begin{bmatrix} 0\\ \mathrm{A}_{17}\sin(\theta_1)\\ \mathrm{A}_{27}\sin(\theta_2)\\ \mathrm{A}_{34}\sin(\theta_3) \end{bmatrix} \tag{32}$$

$$L(q,u)=\begin{bmatrix} K_su-\mathrm{sgn}(x)\mu_x\mathrm{A}_{37}\\ -\mathrm{sgn}(\theta_1)\mu_1\mathrm{A}_{38}\\ -\mathrm{sgn}(\theta_2)\mu_2\mathrm{A}_{39}\\ -\mathrm{sgn}(\theta_3)\mu_3\mathrm{A}_{40} \end{bmatrix} \tag{33}$$

Note that all the μ here is the Coulomb friction coefficient for links and is not linearizable. In this simulation, $\mu_x=0.07$, $\mu_1=\mu_2=\mu_3=0.003$, and A_i are all available in Table 3. The parameters in Table 3 are defined in the following:

L_1 : 0.43 m, total length of the 1st link;
L_2 : 0.33 m, total length of the 2nd link;
L_3 : 0.13 m, total length of the 3rd link;
l_1 : 0.37 m, length from mount joint to the center of gravity of 1st link;
l_2 : 0.3 m, length from 1st joint to the center of gravity of 2nd link;
l_3 : 0.05 m, length from 2nd joint to the center of gravity of 3rd link;
m_1 : 0.4506 kg, mass of the 1st link;
m_2 : 0.219 kg, mass of the 2nd link;
m_3 : 0.0568 kg, mass of the 3rd link;
M: 1.014 kg, mass of the whole cart;
g : 9.8 m/s^2, acceleration of gravity;
I_1 : 0.0042 kg m^2, mass moment of inertia of the 1st link about its center of gravity;
I_2 : 0.0012 kg m^2, mass moment of inertia of the 2nd link about its center of gravity;
I_3 : 0.00010609 kg m^2, mass moment of inertia of the 3rd link about its center of gravity;
C_c : 5.5 Nm s, dynamic friction coefficient between the cart and the track;
C_1 : 0.00026875 Nm s, dynamic friction coefficient for the 1st link;
C_2 : 0.00026875 Nm s, dynamic friction coefficient for the 2nd link;
C_3 : 0.00026875 Nm s, dynamic friction coefficient for the 3rd link.

In this case, the only control unit u (in voltage), generated by the action network, is converted into force by an analog amplifier (with gain $K_s=24.7125$ N/V) to the DC servo motor. Each link here only rotates in a vertical plane, and the sample time interval is chosen to be 5 ms. In order to better show the performance of the proposed three network algorithm with Runge–Kutta methods, the system equations are transformed into the state-space forms as follows:

$$\dot{Q}(t)=f(Q(t),u(t)) \tag{34}$$

$$f(Q(t),u(t))=\begin{bmatrix} \mathbf{0}_{4\times4} & \mathbf{I}_{4\times4}\\ \mathbf{0}_{4\times4} & -\mathrm{F}^{-1}(Q(t))G(Q(t)) \end{bmatrix}+\begin{bmatrix} \mathbf{0}_{4\times1}\\ -\mathrm{F}^{-1}(Q(t))[H(Q(t))-L(Q(t),u(t))] \end{bmatrix} \tag{35}$$

and

$$Q=[x\ \ \theta_1\ \ \theta_2\ \ \theta_3\ \ \dot{x}\ \ \dot{\theta_1}\ \ \dot{\theta_2}\ \ \dot{\theta_3}] \tag{36}$$

Specific physical meanings for these eight state variables are illustrated in Fig. 4. They are

(a) x, position of the cart on the track;
(b) θ_1, vertical angle of the 1st link joint to the cart;
(c) θ_2, vertical angle of the 2nd link joint to the 1st link;
(d) θ_3, vertical angle of the 3rd link joint to the 2nd link;
(e) $\dot{x}$, cart velocity;
(f) $\dot{\theta_1}$, angular velocity of θ_1;
(g) $\dot{\theta_2}$, angular velocity of θ_2;
(h) $\dot{\theta_3}$, angular velocity of θ_3.

4.2. Experiment setup

In this experimental setup, the constraints for the triple-linked inverted pendulum are (1) the cart track extends 1.0 m to both sides from the center point; (2) the voltage applied to the motor should be within [−30 V, 30 V]; (3) each link angle should be within the range of [−20°, 20°] with respect to the vertical axis. Here, condition (2) is guaranteed by using a sigmoid function. While for the other two conditions, if either one fails or both fail, the system will be provided with an external reinforcement signal $r=-1$ at the moment of failure, otherwise $r=0$ all the time. Based on this external reinforcement signal, our three-network ADP approach will automatically and adaptively develop an internal reinforcement signal to facilitate the learning and optimization process over time. For performance assessment, we also adopted the same criteria as in [24] for this triple-linked inverted pendulum in this case study.

One hundred runs in our current study are conducted. Similarly as cart-pole problem, for different runs, here we will use different initial starting states. Specifically, we set the three angles and angle velocity of the triple links to be uniformly within the range of [−1°, 1°] and [−0.50, 0.50]180/*pi* rad/s, respectively. As for x and $\dot{x}$, their initial states are set to zero. The critic network is chosen as a 10–20–1 multi-layer perceptron (MLP) (i.e., ten input neurons, twenty hidden layer neurons, and one output neuron) structure, the action neural network is chosen as 8–14–1 MLP structure and the reference network is set as 9–14–1 MLP structure. The learning parameters such as learning rate, internal cycle, and internal training error threshold for the action network, reference network, and critic network are presented in Table 4.

Table 4
Summary of the parameters used in triple-link inverted pendulum balancing task.

Para.	$l_c(0)$	$l_a(0)$	$l_r(0)$	$l_c(f)$	$l_a(f)$	$l_r(f)$	
Value	0.3	0.3	0.3	0.001	0.001	0.001	
Para.	N_c	N_a	N_r	T_c	T_a	T_r	α
Value	80	100	50	0.05	0.005	0.05	0.95

Table 5
Performance evaluation on case II: triple-link balancing task. The 2nd and the 3rd columns are with our proposed method, while the 4th and the 5th columns are the results from [24].

Noise type	Success rate (%)	# of trial	Success rate (%)	# of trial
Noise free	99	571.4	97	1194
Uniform 5% a.	99	596.9	92	1239
Uniform 10% a.	99	673.1	84	1852
Uniform 5% s.	99	620.1	89	1317
Uniform 10% s.	99	657.9	80	1712
Gaussian $\sigma^2(0.1)$ s.	80	1170.4	85	1508
Gaussian $\sigma^2(0.2)$ s.	50	1372.2	76	1993

4.3. Simulation results

Summary of the simulation results with different noise conditions is presented in Table 5. Comparing with those in [24], we can see the proposed three network ADP architecture in this paper can achieve competitive results in this case as well. To observe how the proposed approach performs under this task, here we present a snapshot of the statistics of different runs under noise free condition. Fig. 5(a) shows the number of required trials for each successful run, and Fig. 5(b) shows the corresponding histogram information. Again, we would like to note that our results presented here are based on different initial conditions as specified in Section 4.2 (i.e., the three angles and angle velocities of the triple links are set to be uniformly within the range of [−1°, 1°] and [−0.50, 0.50]180/*pi* rad/s, respectively, and the initial states of x and $\dot{x}$ are set to zero).

To further analyze how the proposed ADP structure can accomplish the control task, Fig. 6(a)–(h) shows a typical trajectory on the task for all the state variables under noise free condition, namely the position x of the cart (Fig. 6(a)), the first, second, and third joint angle of the triple link pendulum (Fig. 6(b)–(d)), the velocity of the cart Fig. 6(e), and the angular velocity of the first, second, and third joint angle of the triple link pendulum

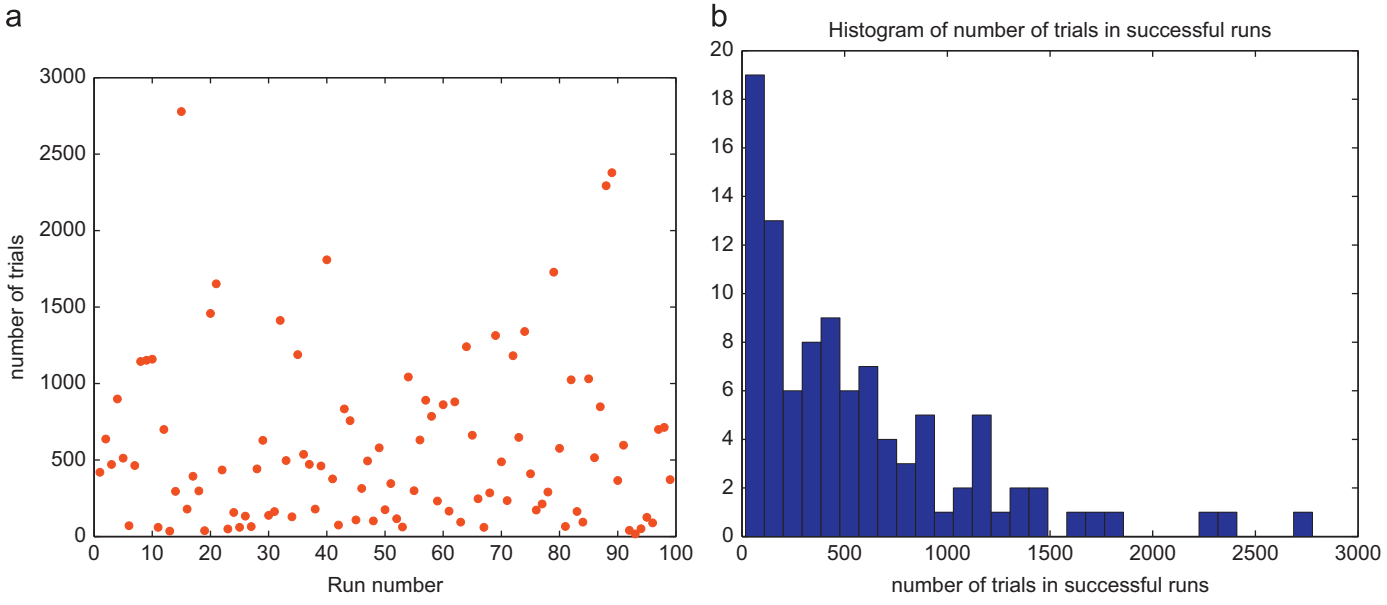


Fig. 5. Statistics of the successful runs. (a) Number of required trials for the successful runs. (b) Histogram the statistics.

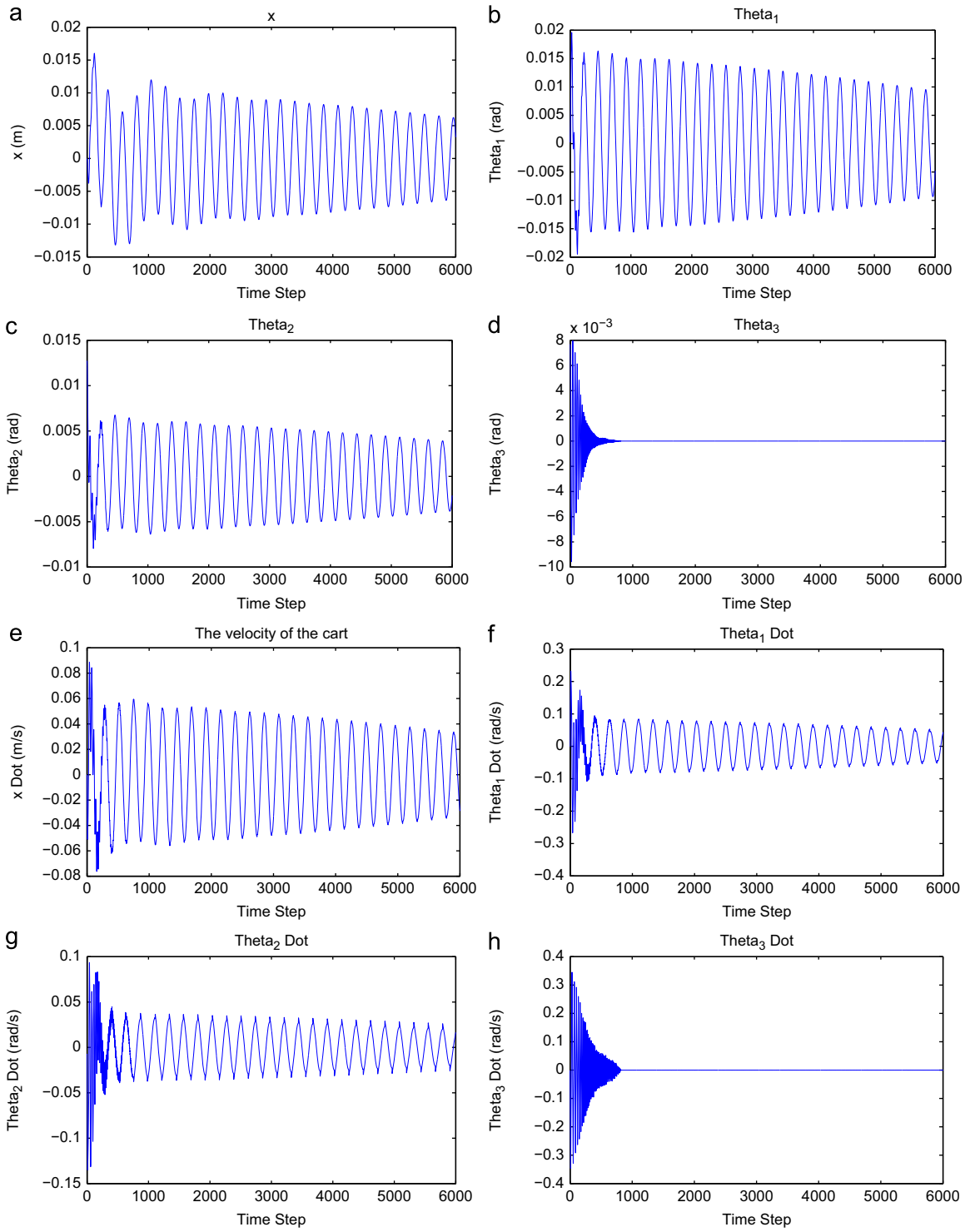


Fig. 6. Typical trajectory on the triple-link inverted pendulum balancing task. (a) The position x of the cart. (b) The 1st joint angle of the triple link pendulum. (c) The 2nd joint angle of the triple link pendulum. (d) The 3rd joint angle of the triple link pendulum. (e) The velocity of the cart. (f) The angular velocity of the 1st joint angle of the triple link pendulum. (g) The angular velocity of the 2nd joint angle of the triple link pendulum. (h) The angular velocity of the 3rd joint angle of the triple link pendulum.

(Fig. 6(f)–(h)). The corresponding histogram information for the position x and three joint angle information are also shown in Fig. 7(a)–(d). All these results clearly indicate that the proposed ADP approach can effectively control the system to achieve desired states during the online learning process.

Furthermore, since the main objective of the reference network is to provide an internal reinforcement signal to facilitate the learning and optimization in the ADP structure, we further analyze how the J value and control action u looks like in this case. Fig. 8(a) shows a snapshot of the convergence of the J value during the learning process, and Fig. 8(b) shows another snapshot of the control action u during a typical successful run. Both figures also clearly demonstrate that our proposed approach can effectively accomplish the control performance in this case.

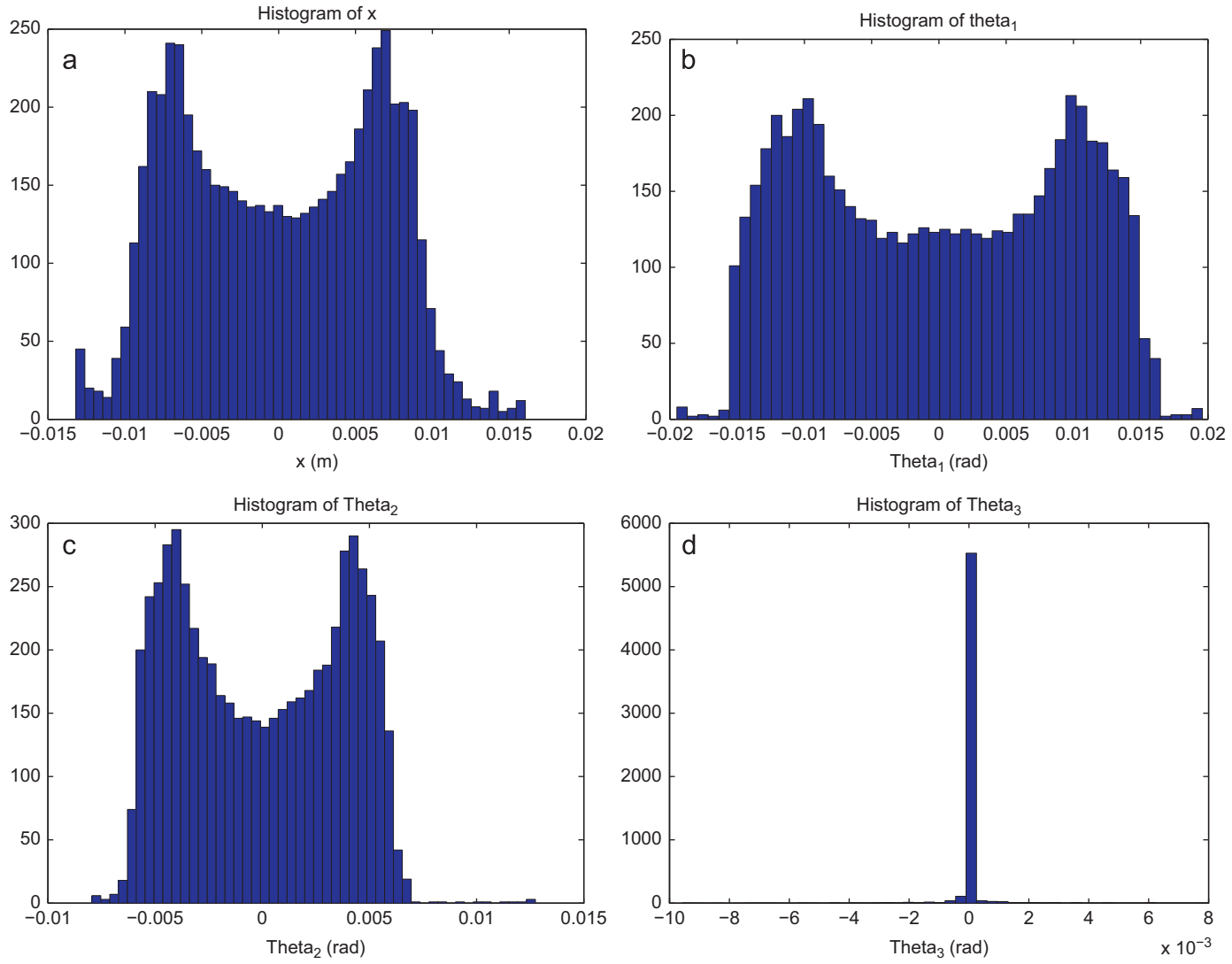


Fig. 7. Typical histogram on the triple-link inverted pendulum balancing task. (a) The histogram of position x of the cart. (b) The histogram of the 1st joint angle of the triple link pendulum. (c) The histogram of the 2nd joint angle of the triple link pendulum. (d) The histogram of the 3rd joint angle of the triple link pendulum.

5. Conclusion and future directions

In this paper, we present a three-network ADP architecture with an action network, a critic network, and a reference network for adaptive learning, control, and optimization. The key idea of our approach is the introduction of a new reference network in the ADP design to develop internal goal-representation to facilitate learning and optimization. This provides an effective way to adaptively and automatically build the internal goal representations for the intelligent systems to achieve the goals. A detailed design architecture and learning algorithm is presented, followed by detailed simulation analysis on two benchmark tasks (i.e., balancing a cart-pole model and a triple-link inverted pendulum model) to demonstrate the effectiveness of our approach.

As a new ADP architecture with the potential powerful capability of learning and optimization, there are several interesting and important future research topics to explore along this direction. First, in this work we mainly focus on the ADP architecture design, it would be important to study the theoretical aspect of this approach such as convergence and stability of such a structure. Such analytical analysis and theoretical proof will be important to fully understand the internal mechanism and foundation of this approach. Second, the learning algorithm we developed in this work is based on the backpropagation method, it will be interesting to investigate other weights turning and adaptation techniques such as Levenberg–Marquardt (LM) algorithm to see its performance [18,31]. Third, our current approach in this paper is based on the action dependent HDP category of ADP design, and it would be useful to investigate the integration of this approach with other types of ADP design, such as DHP and GDHP design to explore its capabilities to handle large-scale complex optimization problems. Also, on a strongly related aspect, our approach in this paper is built on the online actor-critic framework without model network. It is well known that model network plays an important role in many of the complex adaptive control and optimization problems [1], therefore, the integration of our approach with the model network design will be another important future research direction. Finally, our case study in this paper is limited to two balancing tasks (e.g., cart-pole and triple-link inverted pendulum), it would be useful to test the application of our approach to different benchmarks and practical industrial control applications to fully justify its effectiveness across different domains. Our group is currently investigating all these issues and will report corresponding results in the future. Motivated by our results in this work, we hope the proposed three-network ADP architecture will not only advance

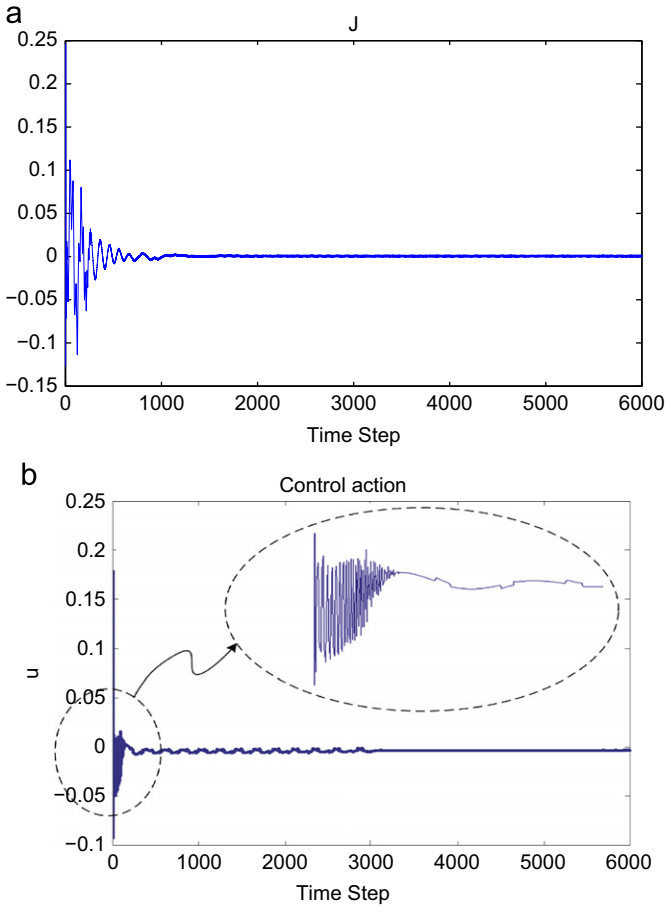


Fig. 8. Typical trajectory of cost-to-go and control action signal on the triple-link inverted pendulum balancing task. (a) The cost-to-go signal on the task. (b) The control action signal on the task.

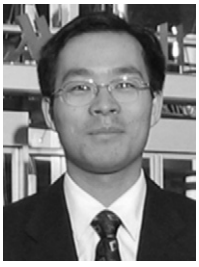
the fundamental ADP research for machine intelligence development, but it can also provide potential new techniques and tools for critical engineering applications across different domains.

Acknowledgment

This work was supported by the National Science Foundation under Grant CAREER ECCS 1053717.

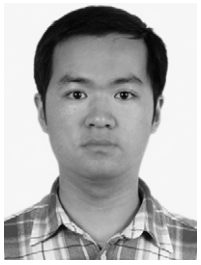
References

[1] P.J. Werbos, Intelligence in the brain: a theory of how it works and how to build it, Neural Netw. (2009) 200–212.
[2] H. He, Self-Adaptive Systems for Machine Intelligence, Wiley, 2011.
[3] P.J. Werbos, Using ADP to understand and replicate brain intelligence: the next level design, in: IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, 2007, pp. 209–216.
[4] J. Si, A.G. Barto, W.B. Powell, D.C. Wunsch, Handbook of Learning and Approximate Dynamic Programming, IEEE Press, 2004.
[5] D.V. Prokhorov, D.C. Wunsch, Adaptive critic designs, IEEE Trans. Neural Netw. 8 (5) (1997) 997–1007.
[6] D.A. White, D.A. Sofge, Handbook of Intelligent Control, Van Nostrand, New York, 1992.
[7] W.B. Powell, Approximate Dynamic Programming: Solving the Curses of Dimensionality, Wiley-Interscience, 2007.
[8] D. Liu, N. Jin, Adaptive dynamic programming for discrete-time systems with infinite horizon and epsilon-error bound in the performance cost, in: Proceedings of the IEEE International Conference on Neural Networks, 2009.
[9] F.Y. Wang, H. Zhang, D. Liu, Adaptive dynamic programming: an introduction, IEEE Comput. Intel. Mag. 4 (2) (2009) 39–47.
[10] K. Vamvoudakis, F.L. Lewis, Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem, in: Proceedings of the IEEE International Conference on Neural Networks, 2009.
[11] S.N. Balakrishnan, J. Ding, F.L. Lewis, Issues on stability of ADP feedback controllers for dynamical systems, IEEE Trans. Syst. Man Cybern., Part B 38 (4) (2008) 913–917 Special Issue on ADP/RL invited survey paper.
[12] A. Al-Tamimi, M. Abu-Khalaf, F.L. Lewis, Adaptive critic designs for discrete-time zero-sum games with application to h-infinity control, IEEE Trans. Syst. Man Cybern. Part B 37 (1) (2007) 240–247.
[13] G.K. Venayagamoorthy, R.G. Harley, Handbook of learning and approximate dynamic programming, in: Application of Approximate Dynamic Programming in Power System Control, IEEE Press, 2004, pp. 479–515.
[14] S. Ray, G.K. Venayagamoorthy, B. Chaudhuri, R. Majumder, Comparison of adaptive critics and classical approaches based wide area controllers for a power system, IEEE Trans. Syst. Man Cybern. Part B 38 (4) (2008) 1002–1007.
[15] H.G. Zhang, Y.H. Luo, D. Liu, Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints, IEEE Trans. Neural Netw. 20 (9) (2009) 1490–1503.
[16] F.Y. Wang, N. Jin, D. Liu, Q. Wei, Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound, IEEE Trans. Neural Netw. 22 (1) (2011) 24–36.
[17] D. Liu, Y. Zhang, H.G. Zhang, A self-learning call admission control scheme for CDMA cellular networks, IEEE Trans. Neural Netw. 16 (5) (2005) 1219–1228.
[18] H. He, J. Fu, X. Zhou, Adaptive learning and control for MIMO system based on adaptive dynamic programming, IEEE Trans. Neural Netw. 22 (7) (2011) 1133–1148.
[19] R.E. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.
[20] P.J. Werbos, Backpropagation through time: what it does and how to do it, Proc/ IEEE, vol. 78, 1990, pp. 1550–1560.
[21] P.J. Werbos, Backpropagation: basics and new developments, in: The Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge, MA, 1995, pp. 134–139.
[22] S. Ferrari, R.F. Stengel, Model-based adaptive critic designs, in: Handbook of Learning and Approximate Dynamic Programming, IEEE Press, 2004.
[23] P.J. Werbos, Neuralcontrol and supervised learning, in: Handbook of Intelligent Control, Van Nostrand, New York, 1992.
[24] J. Si, Y.T. Wang, On-line learning control by association and reinforcement, IEEE Trans. Neural Netw. 12 (2) (2001) 264–276.
[25] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.
[26] P.J. Werbos, Applications of advances in nonlinear sensitivity analysis, in: System Modeling and Optimization, 1981.
[27] P.J. Werbos, Stable adaptive control using new critic designs," [online], available: 〈http://arxiv.org as adap-org/9810001〉, 2008.
[28] R. Enns, J. Si, Helicopter flight control using direct neural dynamic programming, in: Handbook of Learning and Approximate Dynamic Programming, IEEE Press, 2004, pp. 535–559.
[29] J. Si, D. Liu, Direct neural dynamic programming, in: Handbook of Learning and Approximate Dynamic Programming, IEEE Press, 2004, pp. 125–151.
[30] K.D. Eltohamy, C.-Y. Kuo, Nonlinear optimal control of a triple link inverted pendulum with single control input, Int. J. Contr. 69 (2) (1998) 239–256.
[31] Z. Ni, H. He, D.V. Prokhorov, J. Fu, An online actor-critic learning approach with Levenberg–Marquardt algorithm, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN'11), 2011.

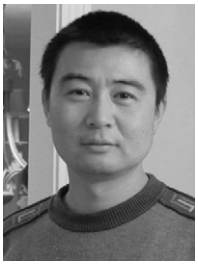


Haibo He received his B.S. and M.S. degrees in Electrical Engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 1999 and 2002, respectively, and the Ph.D. degree in Electrical Engineering from Ohio University, Athens, in 2006. From 2006 to 2009, he was an assistant professor in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, New Jersey. He is currently an assistant professor in the Department of Electrical, Computer, and Biomedical Engineering at the University of Rhode Island, Kingston, Rhode Island.

His research interests include self-adaptive intelligent systems, machine learning and data mining, computational intelligence and applications. He has published one book entitled "Self-Adaptive Systems for Machine Intelligence" (Wiley) in 2011 and edited six conference proceedings as Lecture Notes in Computer Science (LNCS) (Springer). He has served regularly on the organizing committees and program committees of numerous international conferences in his field. He has served as a guest editor for several international journals, including IEEE Computational Intelligence Magazine, Journal of Experimental & Theoretical Artificial Intelligence (Taylor & Francis Group), Applied Mathematics and Computation (Elsevier), Soft Computing (Springer), among others. He has delivered several invited talks at various universities and organizations. Currently, he is an Associate Editor of the IEEE Transactions on Neural Network and IEEE Transactions on Smart Grid, and an Editorial Board member of Cognitive Computation (Springer) and Evolving Systems (Springer). He received the National Science Foundation's CAREER Award in 2011.



Zhen Ni received his B.S. degree in department of control science and engineering from Huazhong University of Science and Technology, China, in 2010. He is currently a Ph.D. student in Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, RI, USA. His major research interests include adaptive dynamic programming, reinforcement learning, and computational intelligence.



Jian Fu received the B.S. degree from Wuhan Science and Technology University (WSTU) in 1994, the M.S. degree from Huazhong University of Science and Technology (HUST) in 1999, and the Ph.D. degree from University of Science and Technology Beijing (USTB) in 2006, all in Electrical Engineering. He is currently an Associate Professor at the School of Automation, Wuhan University of Technology (WHUT), Wuhan, China. His major research interests include advanced control theory and automation, optimization, computational intelligence and applications.