Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 561026, 9 pages
http://dx.doi.org/10.1155/2013/561026

Research Article

A Sarsa(λ) Algorithm Based on Double-Layer Fuzzy Reasoning

Quan Liu,[1,2] Xiang Mu,[1] Wei Huang,[1] Qiming Fu,[1] and Yonggang Zhang[2]

[1] *School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu 215006, China*
[2] *Key Laboratory of Symbol Computation and Knowledge Engineering, Jilin University, Ministry of Education, Changchun 130012, China*

Correspondence should be addressed to Quan Liu; quanliu@suda.edu.cn

Received 1 August 2013; Accepted 13 September 2013

Academic Editor: Zhiqiang Ma

Solving reinforcement learning problems in continuous space with function approximation is currently a research hotspot of machine learning. When dealing with the continuous space problems, the classic Q-iteration algorithms based on lookup table or function approximation converge slowly and are difficult to derive a continuous policy. To overcome the above weaknesses, we propose an algorithm named DFR-Sarsa(λ) based on double-layer fuzzy reasoning and prove its convergence. In this algorithm, the first reasoning layer uses fuzzy sets of state to compute continuous actions; the second reasoning layer uses fuzzy sets of action to compute the components of Q-value. Then, these two fuzzy layers are combined to compute the Q-value function of continuous action space. Besides, this algorithm utilizes the membership degrees of activation rules in the two fuzzy reasoning layers to update the eligibility traces. Applying DFR-Sarsa(λ) to the Mountain Car and Cart-pole Balancing problems, experimental results show that the algorithm not only can be used to get a continuous action policy, but also has a better convergence performance.

1. Introduction

Reinforcement learning is a kind of machine learning methods that gets the maximum cumulative rewards by interacting with the environment [1, 2]. If a reinforcement learning problem can be modeled as a Markov decision process (MDP), methods such as dynamic programming (DP), Monte Carlo (MC), and temporal difference (DP) can be used to get an optimal policy.

Classic reinforcement learning methods are generally used for dealing with discrete state and action space problems, where each of the state values or state action values is stored in a lookup table. This kind of methods can effectively solve simple tasks, but not for large, continuous space problems. At present, the most common approach to solve this problem is using function approximation methods to approximate the state value or action value function. The approximate function can generalize the learned experience from a state space subset to the entire state space. Besides, an agent can choose the best action sequence through the function approximation [3, 4]. A variety of function approximation methods are used to reinforcement learning problems at present. Sutton et al. proposed a gradient TD (GTD) learning algorithm [5], which combined TD algorithms with linear function approximation, and also introduced a new objective function related to Bellman errors. Sherstov and Stone proposed a linear function approximation algorithm based on online adaptive tile coding, in which the experimental results verified its effectiveness [6]. Heinen and Engel used incremental probabilistic neural network to approximate value function in reinforcement learning, which can be used to solve continuous state space problems well [7].

Reinforcement algorithms with the function approximation methods mentioned above usually have slow convergence and generally can only be used for getting discrete action policies [5–9]. By introducing prior knowledge, reinforcement learning algorithms based on fuzzy inference systems (FIS) not only can effectively accelerate the convergence rate, but also may get continuous action policies [10–12]. Horiuchi et al. put forward fuzzy interpolation-based Q-learning, which can solve the continuous space problems [13]. Glorennec and Jouffe combined FIS and Q-learning, using prior knowledge to make the global approximator, which can effectively speed up the convergence rate. However, the

algorithm cannot be used to get a continuous action policy [14]. Fuzzy Sarsa proposed by Tokarchuk et al. can effectively reduce the scale of state space and accelerate the convergence rate, but it easily causes "curse of dimensionality" when applied to multidimensional state-space problems [15]. Type-2 fuzzy Q-learning proposed by Hsu and Juang has strong robustness to noise, but its time complexity is relatively high, and meanwhile, it cannot guarantee convergence [12].

Though the classic Q-iteration algorithms based on only one fuzzy inference system can be used for solving continuous action space problems, there still exist reasons for the slow convergence: for each iteration step in the learning process, there might exist a state-action pair that corresponds to different Q-values due to the structure of FIS. If the next iteration step needs to use the Q-value of the mentioned state-action pair to update the value function, the algorithm will simply select a Q-value randomly, since there are no criteria on how to choose the best one from different Q-values, which will influence the learning speed. Because this situation may happen many times in the learning process, it will greatly slow down the convergence rate.

In allusion to the problem that classic Q-iteration algorithms based on the lookup table and fuzzy inference system converge slowly and cannot obtain continuous action policies as well, DFR-Sarsa(λ), which means Sarsa(λ) based on double-layer fuzzy reasoning, is proposed in this paper, and the convergence is proven theoretically. The algorithm has two-layer fuzzy reasoning. Firstly, it puts states as input of the first fuzzy reasoning layer and gets continuous actions as output. Secondly, the second fuzzy reasoning layer uses the obtained actions from the first layer as input and gets Q-value component of each activation rule of the first layer. Finally, through the combination of two-layer fuzzy reasoning, Q-values of the input states are obtained. What is more, a new eligibility trace based on gradient descent is defined, which is dependent on membership degrees of activation rule in two-layer fuzzy reasoning. Applying DFR-Sarsa(λ) and other algorithms to Mountain Car and Cart-pole Balancing problems, the results show that DFR-Sarsa(λ) not only can obtain a continuous action policy, but also has a better convergence performance.

2. Backgrounds

2.1. Markov Decision Process. In reinforcement learning framework, the process interacting with the environment can be modeled as an MDP [16], and the MDP can be described as a quadruple $M = \langle X, U, \rho, f \rangle$, where

(1) X is the state set and $x_t \in X$ is the state at time t;

(2) U is the action set and $u_t \in U$ is the action that the agent takes at time t;

(3) $\rho : X \times U \to \mathbb{R}^n$ is the reward function, that means, after the agent takes action u_t at time t, the current state transfers from x_t to x_{t+1}, and the agent receives an immediate reward $r(x_t, u_t, x_{t+1})$ at the same time. r_t represents a random reward generated from a distribution with mean $r(x_t, u_t, x_{t+1})$;

(4) $f : X \times U \times X \to [0, 1]$ is the state transition function, where $f(x, u, x')$ represents the probability of reaching x' after taking action u in state x.

The policy $h(x, u)$ is a mapping from state space X to action space U, $h : X \to U$, which represents the probability that the agent selects action u in state x. $h(x, u)$ is used to solve the state value function (V-function) or action value function (Q-function). V-function satisfies (1)

$\forall x \in X$:

$$V^h(x) = \sum_{u \in U} h(x, u) \left(\rho(x, u) + \gamma \sum_{x' \in X} f\left(x, u, x'\right) V^h\left(x'\right) \right). \tag{1}$$

And Q-function satisfies (2)

$$\forall x \in X : Q^h(x, u) = \rho(x, u) + \gamma \sum_{x' \in X} f\left(x, u, x'\right) V^h\left(x'\right). \tag{2}$$

The objective of reinforcement learning is to get the optimal policy h^*. It satisfies for all $x \in X : V^{h^*}(x) \geq V^h(x)$. Under the optimal policy h^*, the optimal V-function and optimal Q-function satisfy (3) and (4), respectively:

$\forall x \in X$:

$$V^*(x) = \max_{u \in U} \left(\rho(x, u) + \gamma \sum_{x' \in X} f\left(x, u, x'\right) V^*\left(x'\right) \right), \tag{3}$$

$\forall x \in X,\ u \in h(x)$:

$$Q^*(x, u) = \rho(x, u) + \gamma \sum_{x' \in X} f\left(x, u, x'\right) \max_{u' \in U} Q^*\left(x', u'\right). \tag{4}$$

If f and ρ are known, DP is a good solution for getting optimal action policy. However if f and ρ are unknown, TD algorithms such as Q-learning or Sarsa can be the choice. Sarsa is an on-policy algorithm, and when the eligibility trace mechanism is introduced, it becomes a more efficient algorithm, which can effectively deal with temporal credit assignment. Besides, Sarsa(λ) can be combined with function approximation to solve continuous state space problems.

Definition 1 is a constraint on bounded MDP (mainly about state-space, action-space, reward, and value function). Attention should be given that all algorithms in this paper meet the definition.

Definition 1 (bounded MDP). X and U are known as finite sets; let Z represent the state-action set; that is, $Z : X \times U$, and then Z is also a finite set. Reward function ρ satisfies $0 \leq \rho(x, u) \leq C$. The bound factor of MDP is $\beta = 1/(1-\gamma)$, where γ is a discount factor. For all $x \in X$ and for all $(x, u) \in Z$, $0 \leq V(x) \leq \beta C$ and $0 \leq Q(x, u) \leq \beta C$ hold.

2.2. Fuzzy Inference System. FIS is a system that can handle fuzzy information. Typically, it mainly consists of a set of fuzzy rules whose design and interaction are crucial to the FIS's performance.

There are many types of fuzzy inference systems at present [17] in which a simple type of FIS named TSK-FIS is described as follows:

$$\text{Rule } R_r\text{: if } x_1 \text{ is } \chi_1^r \text{ AND} \cdots \text{AND } x_n \text{ is } \chi_n^r \text{ then } \mathbf{y} = g_r(\mathbf{x}), \tag{5}$$

where the first part is called antecedent and the second part is called consequent. R_r means rth rule in the rule base. $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ is an N-dimensional input variable. χ_i^r is the fuzzy set in the rth fuzzy rule which corresponds to the ith dimension of input variable. A membership function $\mu_{\chi_{r,i}}(x_i)$ is usually used to describe it. $\mathbf{y} = g_r(\mathbf{x})$ is a polynomial function with an input variable $\mathbf{x}$. If the input $\mathbf{x}$ is a vector, the output $\mathbf{y}$ is also a vector. When $g_r(\mathbf{x})$ is a constant, FIS is called zero-order FIS.

When the FIS has an exact input value $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, we can calculate the firing strength $\phi_r(\mathbf{x})$ of the rth rule (for T-norm product):

$$\phi_r(\mathbf{x}) = \mu_{\chi_{r,1}}(x_1) \cdot \mu_{\chi_{r,2}}(x_2) \cdot \cdots \cdot \mu_{\chi_{r,N}}(x_N), \quad r = 1, \ldots N_R. \tag{6}$$

$\phi_r(\mathbf{x})$ is used to calculate the output of FIS: set firing strength $\phi_r(\mathbf{x})$ as weight, multiply their corresponding consequent and sum up; then we can obtain the final output $Y(\mathbf{x})$ as follows:

$$Y(\mathbf{x}) = \frac{\sum\left(\phi_1(\mathbf{x})\,\mathbf{y}_1 + \phi_2(\mathbf{x})\,\mathbf{y}_2 + \cdots + \phi_{N_R}(\mathbf{x})\,\mathbf{y}_{N_R}\right)}{\sum\left(\phi_1(\mathbf{x}) + \phi_2(\mathbf{x}) + \cdots + \phi_{N_R}(\mathbf{x})\right)}. \tag{7}$$

TSK-FIS can be used for function approximation which approximates the objective function by updating the consequent of fuzzy rules. In general, the approximation error is measured by mean square error (MSE). When FIS gets an optimal approximation performance, the vector $\boldsymbol{\theta}$, which consists of all rules consequents, satisfies (8)

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N}\left(Y_i(\mathbf{x}) - \widehat{Y}_i(\mathbf{x})\right)^2, \tag{8}$$

where $Y_i(\mathbf{x})$ is the objective function and $\widehat{Y}_i(\mathbf{x})$ is its approximate function.

3. DFR-Sarsa(λ)

3.1. The Update of Q-Value. Under the framework of MDP, two-layer fuzzy inference structures are constructed to approximate Q-function. Figure 1 shows the framework using two-layer fuzzy reasoning to approximate Q-function, where the inputs of FIS1 are states; the outputs are continuous actions obtained by FIS1 through fuzzy reasoning; the inputs of FIS2 are continuous actions obtained from FIS1; the outputs are the components of Q-value of the continuous actions. Then, the two-layer FISs are combined to get the approximating Q-function of continuous action $C(\mathbf{x})$.

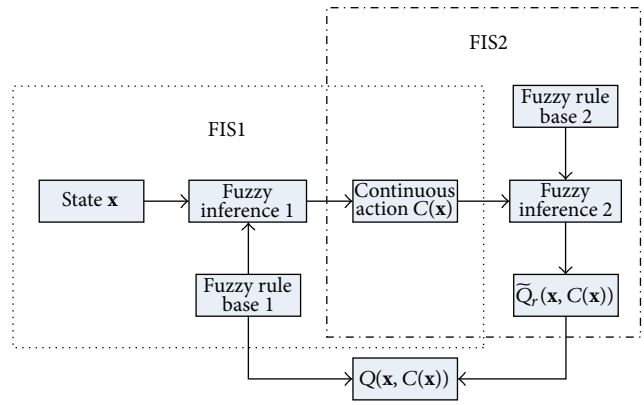

Figure 1: Framework of approximating the Q-value functions by using double-layer fuzzy reasoning.

The main structure of the two-layer FIS is described as follows.

(1) The rule of FIS1 is given as follows:

$$\begin{aligned}\text{Rule } R_r\text{: if } x_1 \text{ is } \chi_1^r \text{ AND} \cdots \text{AND } x_n \text{ is } \chi_n^r\\ \text{then } y = u_{r,1} \text{ with } q_{r,1} = \theta_{r,1}\\ \text{or } y = u_{r,2} \text{ with } q_{r,2} = \theta_{r,2}\\ \vdots\\ \text{or } y = u_{r,M} \text{ with } q_{r,M} = \theta_{r,M},\end{aligned} \tag{9}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ is the state and $u_{r,j}$ is the jth discrete action in the rth fuzzy rule. The action space is divided into M discrete actions. $q_{r,j}$ is a component of Q-value corresponding to the jth discrete action in the rth fuzzy rule. When the state is $\mathbf{x}$, the firing strength of the rth rule is

$$\varphi_r(\mathbf{x}) = \mu_{\chi_{r,1}}(x_1) \cdot \mu_{\chi_{r,2}}(x_2) \cdot \cdots \cdot \mu_{\chi_{r,N}}(x_N). \tag{10}$$

If $\varphi_r(\mathbf{x}) > 0$, we call the rth rule "the activation rule."

In the activation rule R_r, we select an action from M discrete actions by ε-greedy action selection policy according to the value $q_{r,j}$. The selected action is called activation action, denoted by $\widetilde{u}_r$. Therefore, by multiplying activation actions selected from FIS1 to its firing strength $\varphi_r(\mathbf{x})$ and summing them up, we get the continuous action $C(\mathbf{x})$ as follows:

$$C(\mathbf{x}) = \frac{\sum_{r=1}^{N_R} \varphi_r(\mathbf{x})\,\widetilde{u}_r}{\sum_{r=1}^{N_R} \varphi_r(\mathbf{x})}. \tag{11}$$

We call $C(\mathbf{x})$ a continuous action because the change of $C(\mathbf{x})$ is smooth with state $\mathbf{x}$, which does not mean that any action in action space can be selected in state $\mathbf{x}$. To simplify (11), regularize the firing strength $\varphi_r(\mathbf{x})$ as follows:

$$\phi_r(\mathbf{x}) = \frac{\varphi_r(\mathbf{x})}{\sum_{r=1}^{N_R} \varphi_r(\mathbf{x})}; \tag{12}$$

so (11) can be written as

$$C(\mathbf{x}) = \sum_{r=1}^{N_R} \phi_r(\mathbf{x})\, \widetilde{u}_r. \tag{13}$$

(2) The rule of FIS2 is given as follows:

$$\begin{aligned} &\widetilde{R}_{r,1}\text{: if } u \text{ is } \nu_{r,1} \text{ then } q_{r,1} = \theta_{r,1} \\ &\widetilde{R}_{r,2}\text{: if } u \text{ is } \nu_{r,2} \text{ then } q_{r,2} = \theta_{r,2} \\ &\qquad\vdots \\ &\widetilde{R}_{r,M}\text{: if } u \text{ is } \nu_{r,M} \text{ then } q_{r,M} = \theta_{r,M}. \end{aligned} \tag{14}$$

The construction of $\widetilde{R}_{r,j}$ depends on FIS1. The core of the fuzzy set $\nu_{r,j}$ is the jth action of the rth rule in FIS1, and its membership function is described as $\sigma_{\nu_{i,j}}(u)$; the value $q_{r,j}$ from the consequent part of the rule equals the value $q_{r,j}$ in FIS1.

Set the continuous action $C(\mathbf{x})$ obtained from FIS1 as the input of FIS2; it can activate $N_{\widetilde{R}}$ rules of FIS2. Through fuzzy reasoning of FIS2, we can get the Q-value component of the rth rule in FIS1 as follows:

$$\widetilde{Q}_r(\mathbf{x}, C(\mathbf{x})) = \frac{\sum_{j=1}^{M} \sigma_{\nu_{r,j}}(C(\mathbf{x}))\, \theta_{r,j}}{\sum_{j=1}^{M} \sigma_{\nu_{r,j}}(C(\mathbf{x}))}. \tag{15}$$

In the same way of getting (12), regularize the membership function $\sigma_{\nu_{r,j}}(C(\mathbf{x}))$ in (15); we get

$$\mu_{\nu_{r,j}}(C(\mathbf{x})) = \frac{\sigma_{\nu_{r,j}}(C(\mathbf{x}))}{\sum_{j=1}^{M} \sigma_{\nu_{r,j}}(C(\mathbf{x}))}; \tag{16}$$

then (15) can be written as

$$\widetilde{Q}_r(\mathbf{x}, C(\mathbf{x})) = \sum_{j=1}^{M} \mu_{\nu_{r,j}}(C(\mathbf{x}))\, \theta_{r,j}. \tag{17}$$

From (17), we can get $\widetilde{Q}_r(\mathbf{x}, C(\mathbf{x}))$, the Q-value component obtained by the activation rule R_r of FIS1. So when taking continuous action $C(\mathbf{x})$, the Q-value of all activation rules in FIS1 is given as follows:

$$\begin{aligned} Q(\mathbf{x}, C(\mathbf{x})) &= \sum_{r=1}^{N_R} \phi_r(\mathbf{x})\, \widetilde{Q}_r(\mathbf{x}, C(\mathbf{x})) \\ &= \sum_{r=1}^{N_R} \sum_{j=1}^{M} \phi_r(\mathbf{x})\, \mu_{\nu_{r,j}}(C(\mathbf{x}))\, \theta_{r,j}. \end{aligned} \tag{18}$$

From (18) we can see that Q-value depends on fuzzy sets of the two-layer FIS and their shared consequent variables $\theta_{r,j}$. Since fuzzy sets are set according to prior knowledge in advance, they are no longer changed in the algorithm. In order to get convergent Q-value, the FISs require updating $\theta_{r,j}$ until convergence.

In order to minimize the approximation error of FIS, that is, parameter vector $\boldsymbol{\theta}$ meets (8), the algorithm uses gradient descent method to update the parameter vector $\boldsymbol{\theta}$ as follows:

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \frac{1}{2}\alpha \nabla_{\boldsymbol{\theta}_t} \left[r_{t+1} + \gamma Q_t(\mathbf{x}_{t+1}, u_{t+1}) - Q_t(\mathbf{x}_t, u_t) \right]^2 \\ &= \boldsymbol{\theta}_t + \alpha \left[r_{t+1} + \gamma Q_t(\mathbf{x}_{t+1}, u_{t+1}) - Q_t(\mathbf{x}_t, u_t) \right] \nabla_{\boldsymbol{\theta}_t} Q_t(\mathbf{x}_t, u_t), \end{aligned} \tag{19}$$

where the bracket part in (19) is the TD error. Set $\delta = r_{t+1} + \gamma Q_t(\mathbf{x}_{t+1}, u_{t+1}) - Q_t(\mathbf{x}_t, u_t)$; combining the backward TD(λ) algorithm [1], we get

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\delta\mathbf{e}_t, \tag{20}$$

where α is a step-size parameter and $\mathbf{e}_t$ is the eligibility trace vector at time t, which corresponds to parameter vector $\boldsymbol{\theta}_t$. It is updated as follows:

$$\mathbf{e}_t(r, j) \longleftarrow \begin{cases} \gamma\lambda \mathbf{e}_{t-1}(r, j) + \nabla_{\boldsymbol{\theta}_t} Q_t(\mathbf{x}_t, u_t), & \text{if } \nabla_{\boldsymbol{\theta}_t} Q_t(\mathbf{x}_t, u_t) \neq \vec{0}, \\ \gamma\lambda \mathbf{e}_{t-1}(r, j), & \text{else.} \end{cases} \tag{21}$$

$\mathbf{e}_t$ of (21) is a kind of accumulating trace [1], where γ is the discount factor and λ is the decay factor. $\nabla_{\boldsymbol{\theta}_t} Q_t(\mathbf{x}_t, u_t)$ represents the gradient vector obtained by the partial derivative of Q-function on each dimension of parameter vector at time t [1]. According to (18), we get the gradient value of each dimension in $\boldsymbol{\theta}_t$ at time t as follows:

$$\nabla_{\theta^t_{r,j}} Q_t(\mathbf{x}, u) = \nabla_{\theta^t_{r,j}} \sum_{i=1}^{N_R} \sum_{j=1}^{M} \phi_r(\mathbf{x})\, \mu_{\nu_{r,j}}(u)\, \theta_{r,j} = \phi_r(\mathbf{x})\, \mu_{\nu_{r,j}}(u), \quad \text{where } r = 1, \ldots, N_R,\ j = 1, \ldots, M; \tag{22}$$

then (21) can be further expressed as

$$\mathbf{e}_t(r, j) \longleftarrow \begin{cases} \gamma\lambda \mathbf{e}_{t-1}(r, j) + \phi_r(\mathbf{x})\, \mu_{\nu_{r,j}}(u), & \text{if } \phi_r(\mathbf{x}) \neq 0 \text{ and } \mu_{\nu_{r,j}}(u) \neq 0, \\ \gamma\lambda \mathbf{e}_{t-1}(r, j), & \text{else.} \end{cases} \tag{23}$$

3.2. The Learning Process of DFR-Sarsa(λ). In this section, DFR-Sarsa(λ) is proposed based on the algorithm Sarsa in literature [1] and the content of MDP in Section 2.1. DFR-Sarsa(λ) not only can solve reinforcement learning problems with continuous state and discrete action space, but also can solve problems with continuous state and continuous action space. Algorithm 1 describes the general process of DFR-Sarsa(λ).

(1) Initialize parameter vector $\boldsymbol{\theta} = \vec{0}$, eligibility trace vector $\mathbf{e} = \vec{0}$, discount factor γ, step-size parameter α
(2) Repeat(for every episode):
(3) $x \leftarrow$ initial state
(4) According to (10), compute $\phi_r(\mathbf{x})$, $r = 1, \ldots, N_R$
(5) According to ε-greedy policy, select activation action $\widetilde{u}_r$, $r = 1, \ldots, N_R$
(6) According to (13), select action u when state is x
(7) According to (16), compute $\mu_{\nu_{r,j}}(u)$, $r = 1, \ldots, N_R$, $j = 1, \ldots, M$
(8) According to (17) and (18), compute Q_u
(9) Repeat(for each step of episode)
(10) Update eligibility trace: $\mathbf{e}(r, j) \leftarrow \gamma\lambda\mathbf{e}(r, j) + \phi_r(\mathbf{x})\mu_{\nu_{r,j}}(u)$, $r = 1, \ldots, N_R$, $j = 1, \ldots, M$
(11) Take action u, receive next state x' and reward r
(12) $\delta \leftarrow r - Q_u$
(13) According to ε-greedy policy, select activation action $\widetilde{u}_r$, $r = 1, \ldots, N_R$
(14) According to (13), select action u' when state is x'
(15) According to (16), compute $\mu_{\nu_{r,j}}(u')$, $r = 1, \ldots, N_R$, $j = 1, \ldots, M$
(16) According to (10), compute $\phi_r(\mathbf{x}')$, $r = 1, \ldots, N_R$
(17) According to (17) and (18), compute $Q_{u'}$
(18) $\delta \leftarrow \delta + \gamma Q_{u'}$
(19) $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\mathbf{e}$
(20) $u \leftarrow u'$
(21) Until x' is the terminal state
(22) Until preset episode number or other terminal condition meets

Algorithm 1: DFR-Sarsa(λ).

3.3. Convergence Analysis. In the literature [18, 19], the convergence of on-policy TD(λ) using linear function approximation is analyzed in detail. When this kind of algorithm meets some assumptions and lemmas, it converges with probability 1. Since DFR-Sarsa(λ) is exactly such an on-policy TD(λ) algorithm, it can be proved to be convergent when it satisfies some assumptions and lemmas in literature [18]. And this paper will not take too much details for its convergence proof.

Assumption 2. The state transition function and reward function of MDP follow stable distributions.

Lemma 3. *The Markov chain that DFR-Sarsa(λ) depends on is irreducible and aperiodic, and the reward and value function are bounded.*

Proof. Firstly, we prove its irreducibility. According to the property of Markov process, if any two states of a Markov process can be transferred from each other, it is irreducible [20]. DFR-Sarsa(λ) is used for solving reinforcement learning problems that satisfy MDP framework, and the MDP meets Definition 1. Thus for any state x in the MDP, there must exist an f that meets $f(x, u, x') \geq 0$, which indicates that state x can be visited infinitely. Therefore, each state can be transferred to any other state. So the Markov chain of DFR-Sarsa(λ) is irreducible.

Secondly, we prove that it is aperiodic. For the irreducible Markov chain, if one of the states in Markov chain is proved aperiodic, the entire Markov chain can be proved aperiodic. In addition, if a state of the Markov Chain has the property of autoregression, the state can be proven aperiodic [20]. For state x of the MDP, there must exist a state transition satisfying $f(x, u, x) > 0$, which indicates that state x is autoregressive. From the above analysis, we can conclude that the MDP is aperiodic. Therefore, the Markov chain that DFR-Sarsa(λ) depends on is aperiodic.

Finally, we prove that its reward and value function are bounded. Literature [1] shows that value function is a discounted accumulating reward function, which satisfies the equation $Q(x, u) = \sum_{i=0}^{\infty} \gamma^i \rho(x, u)$, $\gamma \in (0, 1)$. By Definition 1, we know that the reward function ρ is bounded, and it satisfies $0 \leq \rho(x, u) \leq C$, where C is a constant. Hence

$$Q(x, u) = \sum_{i=0}^{\infty} \gamma^i \rho(x, u) < \sum_{i=0}^{\infty} \gamma^i C = \lim_{i \to \infty} \frac{\left(1 - \gamma^i\right)}{1 - \gamma} C = \frac{C}{1 - \gamma}. \tag{24}$$

By Inequation (24), we can conclude that value function $Q(x, u)$ is bounded.

In summary, Lemma 3 is proved. □

Condition 1. For each membership function i, there exists a unique state x_i that $\mu_i(x_i) > \mu_i(x)$, for all $x \neq x_i$, while the other membership functions in state x_i are 0; that is, $\mu_{i'}(x_i) = 0$, for all $i' \neq i$.

Lemma 4. *The basis functions of DFR-Sarsa(λ) are bounded, and the basis function vector is linearly independent.*

Proof. Firstly, we prove the basis functions are bounded. From $\phi_r(\mathbf{x}) \in [0, 1]$ and $\mu_{\nu_{r,j}}(C(\mathbf{x})) \in [0, 1]$, we get

$$\left\| \phi_r(\mathbf{x}) \mu_{\nu_{r,j}}(C(\mathbf{x})) \right\|_{\infty} \leq 1, \tag{25}$$

where $\|\cdot\|_\infty$ represents infinite norm. Since the basis function of DFR-Sarsa(λ) is known as $\phi_r(\mathbf{x})\mu_{\nu_{r,j}}(C(\mathbf{x}))$ from (25), we get that the basis functions of DFR-Sarsa(λ) are bounded.

Secondly, we prove the basis function vector is linearly independent. In order to make the basis function vector linearly independent, let the basis functions meet Condition 1 [21], where the function form is shown in Figure 4. From literature [21] we know that, when Condition 1 is met, the basis function vector is linearly independent.

The requirement in Condition 1 can be relaxed appropriately by making the membership degree of $\mu_{i'}(x_i)$ at state x_i a small value, for example, a Gaussian membership function with smaller standard deviation. Applying the membership function to DFR-Sarsa(λ), experimental results show that DFR-Sarsa(λ) is convergent, though the convergence still cannot be given theoretically.

In summary, Lemma 4 is proved. □

Lemma 5. *Step-size parameter α of DFR-Sarsa(λ) satisfies* (26)

$$\sum_{t=0}^{\infty}\alpha_t = \infty, \qquad \sum_{t=0}^{\infty}\alpha_t^2 < \infty. \tag{26}$$

Proof. Set step-size parameter of DFR-Sarsa(λ) $\alpha = 1/(t+1)$, where t is the time step. By Newton power series expansion, we get

$$\sum_{t=0}^{\infty}\alpha_t = \sum_{t=0}^{\infty}\left(1 + \frac{1}{2} + \cdots + \frac{1}{t}\right) = \ln(t+1) + r, \tag{27}$$

where $r = 0.577218$ is Euler's constant. Because $\ln t$ is an increasing function, it satisfies $\sum_{t=0}^{\infty}\alpha_t = \infty$ when $t \to \infty$.

Cosider

$$\sum_{t=0}^{\infty}\alpha_t^2 = \sum_{t=0}^{\infty}\left(1^2 + \left(\frac{1}{2}\right)^2 + \cdots + \left(\frac{1}{t}\right)^2\right) < \frac{2t-1}{t} = 2 - \frac{1}{t}; \tag{28}$$

the inequality part in Inequation (28) can be proven by induction; thus $\sum_{t=0}^{\infty}\alpha_t^2 < \infty$ is met when $t \to \infty$.

By (27) and Inequation (28), we get that the step-size parameter of DFR-Sarsa(λ) satisfies (26); thus we proved Lemma 5. □

Theorem 6. *Under the condition of Assumption 2, if DFR-Sarsa(λ) satisfies Lemma 3 to Lemma 5, the algorithm converges with probability 1.*

Proof. Literature [18] gives the related conclusion that, under the condition of Assumption 2, when on-policy TD(λ) algorithms with linear function approximation meet certain conditions (Lemma 3 to Lemma 5), the algorithms converge with probability 1. DFR-Sarsa(λ) is just such an algorithm and it meets Assumption 2 and Lemma 3 to Lemma 5. So we get that DFR-Sarsa(λ) converges with probability 1. □

4. Experiments

In order to verify DFR-Sarsa(λ)'s performance about the convergence rate, iteration steps after convergence, and the effectiveness of continuous action policy, we take two problems as experimental benchmarks: Mountain Car and Cart-pole Balancing. These two problems are classic episodic tasks with continuous state and action spaces in reinforcement learning, which are shown in Figures 2 and 3, respectively.

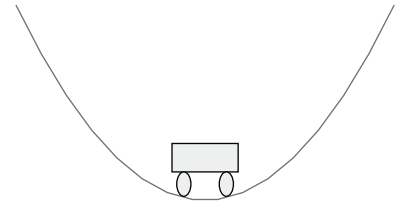

Figure 2: Mountain Car.

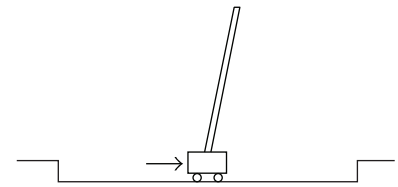

Figure 3: Cart-pole Balancing.

4.1. Mountain Car. Mountain Car is a representative problem with continuous state space, as shown in Figure 2. Suppose the underpowered car cannot accelerate up directly to reach the top of the right side. So, it has to move around more than once to get there. Modeling the task as an MDP, in which the state represented a two-dimensional variable: location and speed; that is, $x = [y, v]$. The action is the force that drives the car to move horizontally, which is bounded in $[-1, 1]$. In this problem, the system dynamics are described as follows:

$$\begin{aligned} v_{t+1} &= \text{bound}\left[v_t + 0.001\,u_t + g\cos\left(3y_t\right)\right], \\ y_{t+1} &= \text{bound}\left[y_t + 1\right], \end{aligned} \tag{29}$$

where $\text{bound}(v_t) \in [-0.07, +0.07]$, $\text{bound}(y_t) \in [-1.5, +0.5]$, and $g = 0.0025$ is a constant related to gravity. In addition, time step is 0.1 s and the reward function is as follows:

$$r_t = \begin{cases} -1, & y < 0.5, \\ 0, & y \geq 0.5. \end{cases} \tag{30}$$

Equation (30) is a punishment reward function, where r_t means the reward received at time t.

In the simulation, the number of episodes is set to 1000. The maximum time step in each episode is also set to 1000. The initial state of the car is $y = -0.5$, $u = 0$. When the car arrives to the destination ($y = 0.5$) or the time steps exceed 1000, we finish this episode and begin a new one. The experiment will end after 1000 episodes.

In order to show the effectiveness of DFR-Sarsa(λ), we compare the algorithm with Fuzzy Sarsa proposed by Tokarchuk et al. [15], GD-Sarsa(λ) proposed by Sutton et al. [3], and Fuzzy $Q(\lambda)$ proposed by Zajdel [22]. Additionally, the effect of eligibility trace on the convergence performance is also tested.

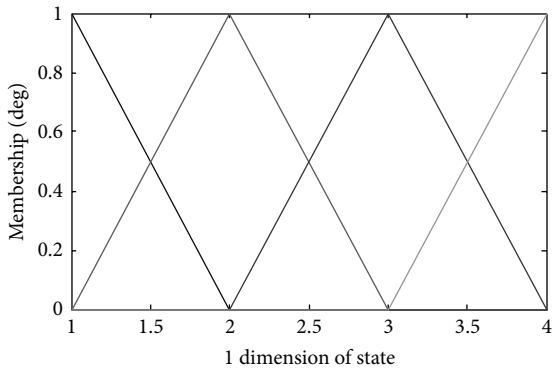

Figure 4: Triangular membership functions (except the domains of states are different, the form of membership functions of position, and velocity is described in Figure 4. Besides, the form of membership functions in Section 4.2 is also the same).

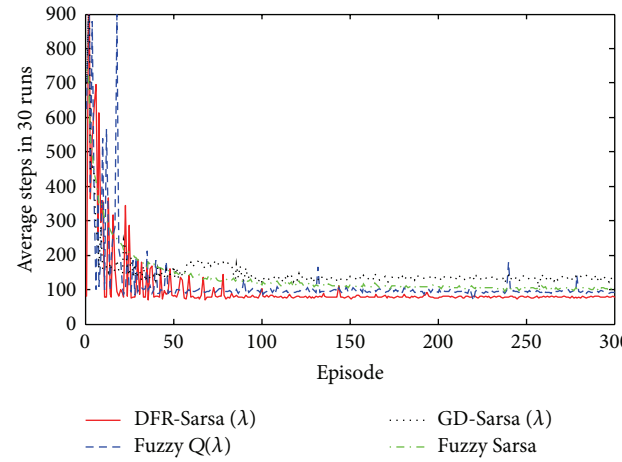

Figure 5: Comparisons on convergent efficiency of the four algorithms.

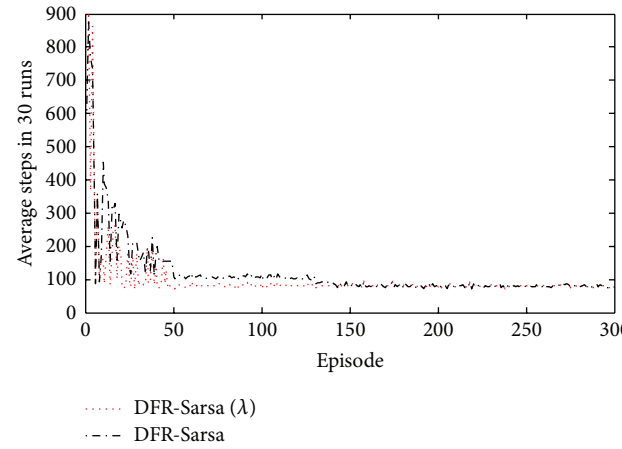

Figure 6: The effect of eligibility traces on convergent efficiency of DFR-Sarsa(λ).

At present, there is no proper way to select parameters that make the four algorithms have their best performance, respectively. In order to make the comparison more reasonable, the parameters that exist in all of the four algorithms will be set at the same value, while the parameters that do not exist in all of the four algorithms will be set at the value from where it firstly comes.

We first set the parameters of DFR-Sarsa(λ): 20 triangular fuzzy sets whose cores are equidistant are used to partition each state variable, which results in 400 fuzzy rules. Similarly, use eight triangular fuzzy sets whose cores are equidistant to partition the continuous action space, where the number of fuzzy rules is 8. Set the other parameters $\varepsilon = 0.001$, $\alpha = 0.9$, $\lambda = 0.9$, and $\gamma = 1.0$. The form of fuzzy partition in Fuzzy Sarsa is the same as in DFR-Sarsa(λ). Other parameters are set to $\varepsilon = 0.001$, $\alpha = 0.9$, and $\gamma = 0.9$. GD-Sarsa(λ) uses 10 tilings of 9×9 to divide state space, where the parameters are set as the best experimental parameters given in literature [1]: $\varepsilon = 0.001$, $\alpha = 0.14$, $\lambda = 0.3$, and $\gamma = 1.0$. The form of fuzzy partition in Fuzzy $Q(\lambda)$ Sarsa is also the same as in DFR-Sarsa(λ). Other parameters are set in accordance with literature [22] to $\varepsilon = 0.005$, $\alpha = 0.1$, $\lambda = 0.1$, and $\gamma = 0.995$.

DFR-Sarsa(λ), Fuzzy Sarsa, GD-Sarsa(λ), and Fuzzy $Q(\lambda)$ are applied to Mountain Car. Figure 5 shows the average result in 30 independent simulation experiments. The x-coordinate indicates the number of episodes, and y-coordinate represents the average time steps the car drives from the initial state to the target. As can be seen from Figure 5, the convergence performance of DFR-Sarsa(λ) is better than those of the other three algorithms.

The detailed performance of the four algorithms is shown in Table 1 (the benchmark time is the average time of a single iteration of DFR-Sarsa(λ)).

In order to test the effectiveness of the proposed eligibility trace, DFR-Sarsa(λ) with eligibility trace and DFR-Sarsa without eligibility trace are both applied in Mountain Car. Figure 6 shows the convergence performance of these two algorithms. It can be seen that these two algorithms converge in the same average time steps, but the convergence speed of DFR-Sarsa(λ) is better than that of DFR-Sarsa.

Table 1: Performance comparison of the four algorithms in Mountain Car problem.

Algorithm	DFR-Sarsa (λ)	Fuzzy Q (λ)	GD-Sarsa (λ)	Fuzzy Sarsa
Average episodes	86	98	118	103
Average steps	79	101	134	112
Average time within an iterative step	100%	80%	30%	65%

4.2. Cart-Pole Balancing. Figure 3 shows a Cart-pole Balancing system, in which the cart can move left or right on the horizontal plane. A pole is hinged to the cart, which can rotate freely within a certain angle. The task is to move the cart horizontally to keep the pole standing in a certain range $[-\pi/2, \pi/2]$. Similarly, modeling the task as an MDP, the state is a two-dimensional variable, which is represented by the vertical angle of pole θ, and the angular velocity of the pole $\dot{\theta}$; that is, $x = [\theta, \dot{\theta}]$. These two state variables satisfy $\theta \in [-\pi/2, \pi/2]$ (rad) and $\dot{\theta} \in [-16\pi, 16\pi]$ (rad/s). The action

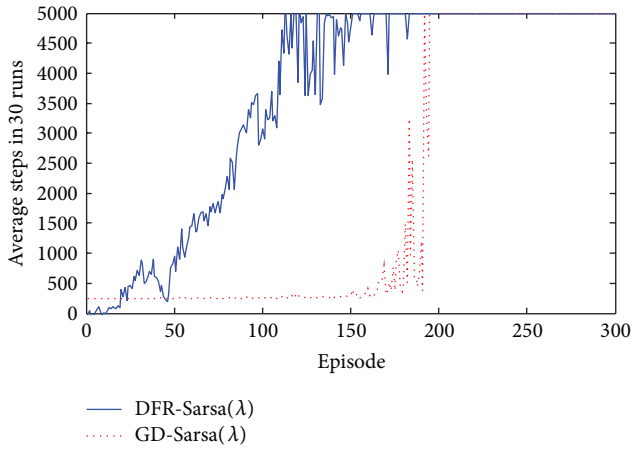

FIGURE 7: Comparisons on convergent efficiency of the two algorithms.

is the force exerted on the cart, which ranges from −50 N to 50 N. In addition, the force is added by a noise force which is uniformly distributed in [−10 N, 10 N]. The system dynamics are described as

$$\ddot{\theta} = \frac{g \sin(\theta) - aml\left(\dot{\theta}\right)^2 \sin(2\theta)/2 - \alpha \cos(\theta) u}{4l/3 - aml \cos^2(\theta)}, \quad (31)$$

where $g = 9.8\,\text{m/s}^2$ is acceleration of gravity, $m = 2.0\,\text{kg}$ is the mass of pole, $M = 8.0\,\text{kg}$ is the mass of cart, $l = 0.5\,\text{m}$ is the length of pole, and constant $\alpha = 1/(m + M)$. The change of reward depends on the change of state. At each time step (0.1 s), when the angle of the pole with vertical direction is no more than $\pi/2$, reward 0 is received. While the angle is more than $\pi/2$, the reward is –1, and the episode ends.

The parameter setting in this example is similar to the settings in Section 3.1, so we only give the difference here: 12 equidistant triangular fuzzy sets are used to partition the continuous action space, which leads to 144 fuzzy rules.

DFR-Sarsa(λ) and GD-Sarsa(λ) are executed on 30 independent simulations on Cart-pole Balancing; the results are shown in Figure 7, where the x-coordinate represents the number of episodes; the y-coordinate represents the average time steps. As can be seen from Figure 7, the convergence performance of DFR-Sarsa(λ) is also better than GD-Sarsa(λ).

The detailed performance of the two algorithms is shown in Table 2 (the benchmark time is the average time of a single iteration of DFR-Sarsa(λ)).

Figure 8 shows the results of GD-Sarsa(λ) and DFR-Sarsa(λ) on Cart-pole Balancing task, respectively. We have known that GD-Sarsa(λ) is based on discrete action policies, while DFR-Sarsa(λ) is based on continuous action policies. From Figure 8 we can see that the continuous action policy obtained by DFR-Sarsa(λ) can make the pole's angle change in only a small angle, while discrete action policy obtained by GD-Sarsa(λ) makes the pole's angle change in a large range. This fact indicates that policies obtained by DFR-Sarsa(λ) are much more stable than that of GD-Sarsa(λ). Thus, DFR-Sarsa(λ) is more suitable for applications which require more stable policies.

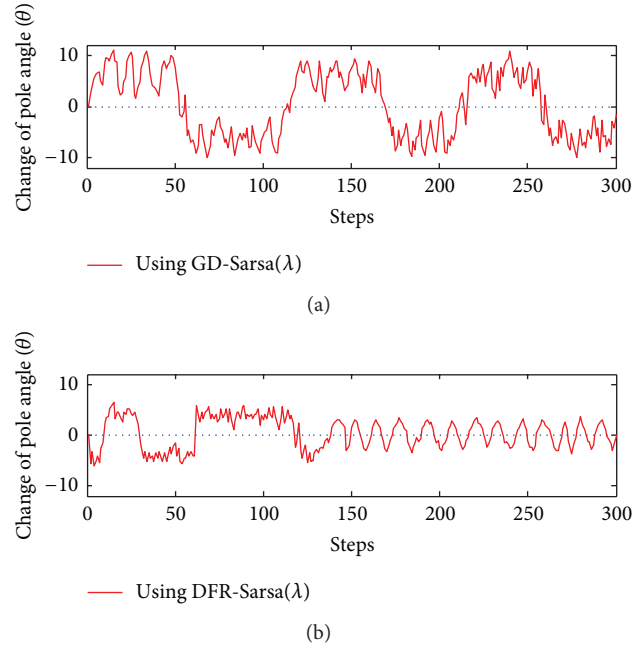

FIGURE 8: Changes of the pole angle θ when using the two mentioned algorithms, respectively.

TABLE 2: Performance comparison of the two algorithms in Cart-pole Balancing problem.

Algorithm	Episodes		Average time within an iterative step
	Minimum episodes	Average episodes	
DFR-Sarsa(λ)	135	155	100%
GD-Sarsa(λ)	179	204	46%

5. Conclusions

In allusion to the problem that classic reinforcement learning algorithms based on lookup table or function approximation converge slowly and are difficult to obtain continuous action policies, this paper presents an algorithm with eligibility trace based on double-layer fuzzy reasoning—DFR-Sarsa(λ). Firstly, the algorithm constructs two fuzzy reasoning layers to approximate Q-function, which are associated with state, action, and Q-value. Then, it uses gradient descent method to update eligibility trace and the consequent of fuzzy rules in the two FISs. Applying the proposed algorithm and other three similar relatively new algorithms to Mountain Car and Cart-pole Balancing system, experimental results show that, compared with reinforcement learning algorithms using only one fuzzy inference system, our algorithm requires fewer steps to convergence, though it increases the time complexity; compared with algorithms based on lookup table or some other function approximation methods, DFR-Sarsa(λ) has better convergence performance and can obtain a continuous action policy.

The performance of DFR-Sarsa(λ) relies on the two-layer fuzzy inference systems, while the performance of fuzzy

inference system mainly depends on the fuzzy sets and fuzzy rules. In this paper, the type of fuzzy sets and the number of rules are given as prior knowledge, and they are no longer changed during the learning process. In order to achieve a much better convergence performance, we will focus on using appropriate optimization algorithms to optimize the membership functions and adjust the fuzzy rules adaptively.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61070223, 61103045, 61070122, 61272005, and 61170314), the Natural Science Foundation of Jiangsu Province (BK2012616), the Foundation of Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University (93K172012K04).

References

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

[2] Q. Yan, Q. Liu, and D. Hu, "A hierarchical reinforcement learning method based on heuristic reward function," *Journal of Computer Research and Development*, vol. 48, no. 12, pp. 2352–2358, 2011.

[3] R. S. Sutton, D. McAllester, S. Singh et al., "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, pp. 1057–1063, Denver, Colo, USA, 1999.

[4] H. R. Maei and R. S. Sutton, "GQ(λ): a general gradient algorithm for temporal-difference prediction learning with eligibility traces," in *Proceedings of the 3rd Conference on Artificial General Intelligence (AGI '10)*, pp. 91–96, Atlanta, Ga, USA, March 2010.

[5] R. S. Sutton, C. Szepesvári, and H. R. Maei, "A convergent O(n) algorithm for off-policy temporal-difference learning with linear function approximation," in *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NIPS '08)*, pp. 1609–1616, Vancouver, Canada, December 2008.

[6] A. A. Sherstov and P. Stone, "Function approximation via tile coding: automating parameter choice," in *Proceedings of the 5th Symposium on Abstraction, Reformulation and Approximation*, pp. 194–205, New York, NY, USA, 2005.

[7] M. R. Heinen and P. M. Engel, "An incremental probabilistic neural network for regression and reinforcement learning tasks," in *Proceedings of the 20th International Conference on Artificial Neural Networks*, pp. 170–179, Berlin, Germany, 2010.

[8] M. Geist and O. Pietquin, "Parametric value function approximation," in *Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL '11)*, pp. 9–16, New York, NY, USA, April 2011.

[9] J. Pazis and M. G. Lagoudakis, "Learning continuous-action control policies," in *Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL '09)*, pp. 169–176, Nashville, Tenn, USA, April 2009.

[10] A. Bonarini, A. Lazaric, F. Montrone, and M. Restelli, "Reinforcement distribution in fuzzy Q-learning," *Fuzzy Sets and Systems*, vol. 160, no. 10, pp. 1420–1443, 2009.

[11] P. Yves Glorennec, "Fuzzy Q-learning and dynamical fuzzy Q-learning," in *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, pp. 474–479, Orlando, Fla, USA, June 1994.

[12] C. H. Hsu and C. F. Juang, "Self-organizing interval type-2 fuzzy Q-learning for reinforcement fuzzy control," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '11)*, pp. 2033–2038, Anchorage, Alaska, USA, October 2011.

[13] T. Horiuchi, A. Fujino, O. Katai, and T. Sawaragi, "Fuzzy interpolation-based Q-learning with continuous states and actions," in *Proceedings of the 5th IEEE International Conference on Fuzzy System*, pp. 594–600, Los Angeles, Calif, USA, September 1996.

[14] P. Y. Glorennec and L. Jouffe, "Fuzzy Q-learning," in *Proceedings of the 6th IEEE International Conference on Fussy Systems*, pp. 659–662, Cambridge, Mass, USA, July 1997.

[15] L. Tokarchuk, J. Bigham, and L. Cuthbert, "Fuzzy sarsa: an approach to linear function approximation in reinforcement learning," in *Proceedings of the International Conference on Artificial Intelligence and Machine Learning*, pp. 1021–1026, Berlin, Germany, 2005.

[16] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, *Simulation-Based Algorithms for Markov Decision Processes*, Springer, New York, NY, USA, 2007.

[17] O. Castillo and P. Melin, *Type-2 Fuzzy Logic: Theory and Applications*, Springer, New York, NY, USA, 2008.

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

[19] P. Dayan, "The convergence of TD(λ) for general λ," *Machine Learning*, vol. 8, no. 3-4, pp. 341–362, 1992.

[20] L. Cihua, *Stochastic Process*, Huazhong University of Science and Technology Press, Wuhan, China, 2008.

[21] B. Lucian, B. Robert, S. Bart De, and E. Damien, *Reinforcement Learning and Dynamic Programming Using Function Approximation*, CRC Press, Boca Raton, Fla, USA, 2010.

[22] R. Zajdel, "Fuzzy Q(λ)-learning algorithm," in *Proceedings of the 10th International Conference on Artificial Intelligence and Soft Computing*, pp. 256–263, Berlin, Germany, 2010.