Mathematical Biosciences and Engineering

MBE, 19(9): 9258–9290.
DOI: 10.3934/mbe.2022430
Received: 26 February 2022
Revised: 24 May 2022
Accepted: 12 June 2022
Published: 23 June 2022

http://www.aimspress.com/journal/MBE

Research article

An approach to solving optimal control problems of nonlinear systems by introducing detail-reward mechanism in deep reinforcement learning

Shixuan Yao[1], Xiaochen Liu[2,*], Yinghui Zhang[2], Ze Cui[3]

[1] School of Software Engineering, Dalian University of Foreign Languages, Dalian 116044, China
[2] School of Mechanical Engineering, Dalian Jiaotong University, Dalian 116028, China
[3] School of Control Science and Engineering, Dalian University of Technology, Dalian 116024, China

* **Correspondence:** Email:lxc_jason@163.com.

Abstract: In recent years, dynamic programming and reinforcement learning theory have been widely used to solve the nonlinear control system (NCS). Among them, many achievements have been made in the construction of network model and system stability analysis, but there is little research on establishing control strategy based on the detailed requirements of control process. Spurred by this trend, this paper proposes a detail-reward mechanism (DRM) by constructing the reward function composed of the individual detail evaluation functions in order to replace the utility function in the Hamilton-Jacobi-Bellman (HJB) equation. And this method is introduced into a wider range of deep reinforcement learning algorithms to solve optimization problems in NCS. After the mathematical description of the relevant characteristics of NCS, the stability of iterative control law is proved by Lyapunov function. With the inverted pendulum system as the experiment object, the dynamic environment is designed and the reward function is established by using the DRM. Finally, three deep reinforcement learning algorithm models are designed in the dynamic environment, which are based on Deep Q-Networks, policy gradient and actor-critic. The effects of different reward functions on the experimental accuracy are compared. The experimental results show that in NCS, using the DRM to replace the utility function in the HJB equation is more in line with the detailed requirements of the designer for the whole control process. By observing the characteristics of the system, designing the reward function and selecting the appropriate deep reinforcement learning algorithm model, the optimization problem of NCS can be solved.

Keywords: Hamilton-Jacobi-Bellman (HJB); nonlinear control system (NCS); detail-reward mechanism (DRM); reinforcement learning; Lyapunov function

1. Introduction

In the last two decades, the research and application of nonlinear systems have made considerable progress. Combined with the Lyapunov stability theory, there have been many research results in the control theory of nonlinear systems. Wu et al. [1] designed an adaptive quantitative control method for analyzing uncertain nonlinear strictly feedback systems by combining the backstepping technique and Lyapunov stability theory. Shatyrko et al. [2] studied the stabilization of Lur'e-type nonlinear indirect control systems with time-delay argument, and obtained the sufficient conditions for the absolute stability of the system by the Lyapunov method. Kai et al. [3] discussed a control method that can generate the desired limit-cycle-like behavior for a two-dimensional discrete-time nonlinear control system. Wei [4] discussed the boundedness of nonlinear nabla fractional-order systems, and by using the nabla Laplace transform, derived two stability criteria in the form of Lyapunov's theorem. Pole et al. [5] described progressively globally asymptotically stable nonlinear control systems through symbolic models, which enables one to utilize techniques from supervisory control and algorithms from game theory for controller synthesis purposes.

At the same time, the development of optimization theory of dynamic systems and the research and application of nonlinear control systems have opened up huge application space. Zhang et al. [6] introduced the control research of adaptive dynamic programming (ADP) in synchronous generator and generator excitation of multi-machine power system. Volckaert et al. [7] use iterative learning control (ILC) to solve the control problem of the cart-cable system with nonlinear characteristics. Gao et al. [8] used a global adaptive dynamic programming(GADP) method for cooperative adaptive cruise control (CACC) of connected vehicles with unknown nonlinear dynamics. Trélat. [9] conducted research on some common methods used to solve nonlinear optimal control problems in aerospace, such as Pontryagin's maximum principle and conjugate point theory.

At present, many scholars combine nonlinear system and optimal control theory to carry out research on optimal control of nonlinear system. The main methods to solve the optimal control problem are classical variational method, maximum value principle and dynamic programming [10–12]. One of the main difficulties of these classical optimal control theories is to determine the optimal control of a nonlinear system, so it is necessary to solve the Hamilton-Jacobi-Bellman (HJB) partial differential equations (PDEs) [13]. Due to the complexity of the nonlinear system and the various constraints of the real systems, nonlinear HJB equations and nonlinear two-point boundary value problems cannot be solved analytically. Therefore, it is of great significance to study some numerical solution methods or explicit optimal solutions approximate to the optimal solution to solve the optimal control problem.

With the development of various technologies in the field of artificial intelligence, many methods of combining artificial intelligence algorithms to solve nonlinear system problems emerge [14,15]. Another powerful methodology for solving optimal control problems is the Adaptive Dynamic Programming (ADP) algorithm, which was proposed by Werbos in 1991 [16]. Combining the reinforcement learning and the dynamic programming, ADP simulates the human learning through environmental feedback and is considered to be a method very close to the intelligence of the human

brain. This method effectively solves the problem of "curse of dimensionality" in dynamic programming [17]. In 1997, Prokhorov and Wunsch discussed the design of Heuristic Dynamic Programming (HDP), Dual Heuristic Programming (DHP) and Global Dual Heuristic Dynamic Programming (GDHP), and proposed the implementation method and training steps of ADP [18]. The essence of ADP is to use the function approximation structure to approximate the performance index function, control strategy in the dynamic programming equation, and obtain the optimal control and optimal performance index function under the condition of meeting the Behrman optimality principle [19–21]. In the ADP structure, there are generally three parts, namely the dynamic system, the critic performance index function and the control action. Each part can be replaced by a neural network [22]. The mathematical derivation process of ADP method is based on minimizing the performance index function or cost function in the infinite horizon, and the performance index function is the integration or summation of the utility function. In [23], combined with Bellman's optimization principle, the method of introducing discount factor into cost function was used to solve the optimal control problem. This method is closer to the role of discount rate in reinforcement learning, therefore, the model can better refer to the value return in the future during the process of training [24].

The research of the above literature is mainly aimed at the design and optimization of the network structure in ADP, and proposing ADP algorithms with different structures. In recent years, many scholars have done a lot of research on the stability analysis and algorithm convergence of the closed-loop system composed of the ADP iterative algorithm. Mu et al. [25] proposed an approximate optimal control strategy, obtained the approximate optimal tracking control law of uncertain nonlinear systems with predefined cost function by using ADP, and solved the tracking control problem of continuous-time nonlinear systems with unmatched uncertainty. Dong et al. [26] proposed a critic-only ADP algorithm to approximate the solution of the HJB equation and the corresponding optimal control strategy. This strategy solves the tracking control problem described by Euler-Lagrange equations with uncertain system parameters, furthermore, uniformly ultimately bounded stability is guaranteed via a Lyapunov-based stability analysis. Song et al. [27] proposed a value iteration (VI) algorithm based on ADP to solve the fixed-point tracking control problem, gave the convergence proof of VI algorithm, and proved that the iterative cost function can converge to the optimal value accurately. Liang et al. [28] constructed a partial-policy iterative adaptive dynamic programming (ADP) algorithm to solve the optimal control problem of nonlinear systems with discounted total rewards. Compared with the traditional strategy iterative ADP algorithm, the calculation amount in the iterative process is reduced, and the stability is improved. Fan et al. [29] proposed a novel control strategy based on robust adaptive dynamic programming (RADP) for optimal control of a class of output-constrained continuous-time unknown nonlinear systems. Yang et al. [30] proposed a self-learning robust optimal control scheme based on adaptive dynamic programming (ADP), using indicator functions and concurrent learning techniques to solve mismatched perturbed input affine continuous-time nonlinear systems.

However, most ADP algorithms are either implemented offline by iterative methods or require prior knowledge of system dynamics. These ADP algorithms are intractable for real-time control applications since exact knowledge of nonlinear dynamic systems is often not available. Many scholars use reinforcement learning methods to solve the above problems. Liu et al. [31] proposed a robust adaptive control algorithm based on Reinforcement Learning (RL) to solve the control problem of continuous-time uncertain nonlinear systems with input constraints. The algorithm uses a

single evaluation network to obtain approximate optimal control, which ensures a certain stability of the uncertain nonlinear continuous-time system. Liu et al. [32] constructed two kinds of neural networks to achieve approximate optimal control for continuous-time nonlinear systems with unknown structure and control constraints. Among them, the cyclic neural network (RNN) is used to identify the system model of the system, and two feedforward neural networks (Feedforward NN) are used as the execution network and the evaluation network, respectively, for the approximation of the optimal control strategy and the optimal performance index function. Zhao et al. [33] proposed a RL-based cyclic fixed finite-time algorithm to solve the HJB equation concerning the optimal control of continuous nonlinear finite-time systems with uncertain system structure. Zhao et al. [34] proposed a new reinforcement learning strategy consisting of two parts: an online learning optimal control of a nominal system and a feedforward neural network (NN) compensation that handles uncertain input constraints. This strategy solves the optimal stability problem of unknown nonlinear systems constrained by uncertain inputs. Wang et al. [35] studied the approximate optimal control of continuous-time non-affine nonlinear systems using reinforcement learning, and used effective precompensation techniques for proper system transitions. Combined with the overall strategy iteration to alleviate the needs of system dynamics. J. W. Kim et al. [36] proposed a model-based RL approach that uses deep neural networks (DNNs) to iteratively learn the solutions of HJB and its associated equations, which can significantly improve the performance of learning policies in uncertain initial states and in the presence of state noise.

The algorithm based on dynamic programming and iterative strategy starts with the performance index function in the state space of the control system, explores the control effect of the control sequence, and obtains the optimal control law [37–40]. The above literature has achieved great results in the design of network structure and the convergence proof of policy algorithm. In the study of optimal control of nonlinear systems, more attention is paid to the control details of dynamic programming algorithms. Its purpose is to expect that the changes in the state space during the whole control process can meet the detailed requirements of the control system, and finally achieve the control goal. This requires that the utility function in the performance indicator function must be able to express the specific needs of users. Most of the above literatures use quadratic forms when defining utility functions, and they do not have a good description of the relationship between utility functions and system states. In addition, there are great limitations on the setting of parameters in utility functions. As for how to design a utility function that is closely related to the actual control objective in nonlinear systems, there are few literature, and this is also the research direction of this paper.

This paper proposes detailed regulation of DRM based on the control process. By designing the evaluation function of the control target, multiple evaluation functions are combined into a reward function to replace the utility function in the performance index function, and then the reward function is introduced into the deep reinforcement learning algorithms. We use three algorithms to verify the effectiveness of DRM, which are based on Deep Q-Networks, policy gradient and actor-critic frameworks respectively.

The remainder of this paper is organized as follows. Section 2 describes nonlinear system properties and introduces the limitations of utility functions in iterative algorithms. In Section 3, an alternative approach to the utility function is proposed combining Lyapunov stability and Q-learning. Taking the inverted pendulum system as the experiment object, Section 4 establishes the dynamic environment, constructs the reward function mechanism of the inverted pendulum system, analyzes

the deep reinforcement learning algorithm model and designs experiments to verify the effectiveness of the algorithm. Finally, Section 5 concludes this work.

2. Problem statement

Consider a continuous-time nonlinear system given by:

$$\dot{x}(t) = f[x(t), u(t), t], \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the system state vector, $u(t) \in \Omega \subset \mathbb{R}^m$ is the control input vector, and $f(\cdot)$ is a continuously differentiable vector function. The boundary conditions are that the initial condition $x(t_0) = x_0$ is fixed and the end condition $x(t_f)$ is free. The performance index function is defined as:

$$J(x(t), t) = \varphi(x(t_f), t_f) + \int_t^{t_f} L(x(\tau), u(\tau), \tau) d\tau, \tag{2}$$

where the control vector $u(t)$ is unconstrained and continuous. In the interval $[t, t_f]$, the scalar functions $\varphi(\cdot)$ and $L(\cdot)$ are continuous and twice differentiable. Function $J(x(t), t)$ is continuous and has continuous first and second order partial derivatives with respect to $x(t)$ and t. In the admissible control domain Ω, $u[\mathrm{t}, \mathrm{t_f}] \in \Omega$, so the optimal performance index function is defined as:

$$J^*(x(t), t) = \min_{u[t,t_f] \in \Omega} \left\{ \varphi(x(t_f), t_f) + \int_t^{t_f} L(x(\tau), u(\tau), \tau) d\tau \right\}. \tag{3}$$

Then the continuous time HJB equation can be:

$$\frac{\partial J^*(x(t),t)}{\partial t} = -\min_{u(t) \in \Omega} \left\{ L(x(t), u(t), t) + \left(\frac{\partial J^*(x(t),t)}{\partial x} \right)^T f[x(t), u(t), t] \right\}. \tag{4}$$

For optimal control law $u^*(\mathrm{t})$, Eq (4) can be written as:

$$\frac{\partial J^*(x(t),t)}{\partial t} = -L(x(t), u^*(t), t) - \left(\frac{\partial J^*(x(t),t)}{\partial x} \right)^T f[x(t), u^*(t), t]. \tag{5}$$

According to Euler's discretization [41], the discrete NCS function of the control system can be as follows:

$$x_{k+1} = F(x_k, u_k). \tag{6}$$

The nonlinear system is expressed by the function $F(x_k, u_k)$, where $x_k \in \Omega_x \subset \mathbb{R}^n$ is the system state vector, and $u_k \in \Omega_u \subset \mathbb{R}^m$ is the control input vector. Ω_x and Ω are defined as:

$$\begin{cases} \Omega_x = \{x_k \ \ | x_k \in \mathbb{R}^n \ \text{ and } \ \|x_k\| < \infty\} \\ \Omega_u = \{u_k \ \ | u_k \in \mathbb{R}^m \ \text{ and } \ \|u_k\| < \infty\}' \end{cases} \tag{7}$$

where $\|\cdot\|$ denotes the Euclidean norm. Let $\underline{u}_k = (u_k, u_{k+1}, \ldots)$ denote the control sequence from k to ∞. Let $U(x_k, u_k)$ be the utility function. The performance index function is defined as:

$$J(x_k, \underline{u}_k) = \sum_{i=k}^{\infty} U(x_i, u_i). \tag{8}$$

For a control sequence $\underline{u}_k$, the optimal performance index function can be defined as:

$$J^*(x_k) \doteq \min_{\underline{u}_k}\{J(x_k,\ \underline{u}_k)\}. \tag{9}$$

So we have:

$$J^*(x_k) = \min_{u_k}\{U(x_k,u_k) + J^*(x_{k+1})\}. \tag{10}$$

Then according to Bellman's principle of optimality [42], the discrete time HJB equation can be:

$$J^*(x_k) = U(x_k,u^*(x_k)) + J^*(F(x_k,u^*(x_k))). \tag{11}$$

The optimal single control law $u^*(x_k)$ can be expressed as:

$$u^*(x_k) = arg\ \min_{u_k}\{U(x_k,u(x_k)) + J^*(F(x_k,u(x_k)))\}. \tag{12}$$

The above is the basic principle of using dynamic programming to solve the optimal control of the two systems. The optimal control vector function and the optimal state trajectory can be solved through the HJB equation, but the process is often very complicated or difficult to achieve. In addition, when the discrete system is high-order, it is easy to cause the "curse of dimensionality" problem. A common alternative method is ADP. The definition of the performance index function in ADP is usually in the form of a quadratic performance index, similar to the following expression as follows:

$$J(x(t),u(t),t) = \frac{1}{2}\int_t^\infty [x^T(t)Q(t)x(t) + u^T(t)R(t)u(t)]\,dt, \tag{13}$$

where Q is a $l \times l$-dimensional nonnegative definite matrix, and R is a $m \times m$-dimensional positive definite matrix. The values of l and m depend on the dimensions of $x(t)$ and $u(t)$ respectively. The utility function is generally expressed in the following form:

$$U(x_k,u_k) = \left(x_k^T Q x_k + u_k^T R u_k\right). \tag{14}$$

In most cases, Q and R are set as identity matrix with suitable dimensions, but this is often not very consistent with the real control system and control objectives. For example, in a specific control system, the high-frequency control law may damage electronic components, and high-frequency control should be avoided. The mechanical characteristics of some systems require that the control law conforms to a functional flexible control as much as possible. Moreover, due to the specific operation environment, the state space under the output of the control law is required to satisfy the specific requirements. This requires that the utility function has a more flexible nonlinear form. As far as we know, there is very little literature about how to assign appropriate values to Q and R in order that the utility function satisfies the real control system requirements. In fact, it is difficult or even impossible to accurately determine the two matrices to match the real control system.

3. Construction of detail-reward mechanism

In order to solve the above problems, we try to modify the utility function so that it can not only satisfy the mathematical requirements of the best performance indicator, but also satisfy our

requirements for the control process. Firstly, define a new function $R(x_k, u_k)$ and use it as the mapping object of the utility function $U(x_k, u_k)$. Then, for a known NCS $f[x(t), u(t), t]$, the discrete method is used to establish the function $F(x_k, u_k)$, so as to establish the dynamic environment. Using the dynamic environment, the output vector x_{k+1} under the input vectors x_k and u_k can be solved, and the $R(x_k, u_k)$ matching with $U(x_k, u_k)$ can be output.

Thus, how to design the dynamic environment and how to design the output function $R(x_k, u_k)$ corresponding to $U(x_k, u_k)$ has become one of the key problems to solve in the nonlinear control systems. For mathematical convenience, we make the following conventions and constraints on $R(x_k, u_k)$:

1) The discrete nonlinear control sequence can be defined as a finite Markov decision process (MDP) $M = (S, A, R)$, where: $S = \{x_0, x_1, \ldots, x_N\}, x_i \in \Omega_x, N > 1$ is a finite set of states; $A = \{u_0, u_1, \ldots, u_N\}, u_i \in \Omega_u$ is a set of control actions; R is the reward distributions, and it can be written as $R: S \times A \times S \mapsto \mathbb{R}$, with $R(x_k, u_k, x_{k+1}) \in [0, r_{\max}]$ being the reward received upon taking u_k in state x_k transitioning to state x_{k+1}. The boundedness of the reward function of $R(x_k, u_k, x_{k+1})$ is to ensure that the performance index function composed of it is bounded. The intuitive explanation of the reward function is: the higher the compliance of the control effect, the closer the value is to r_{max}, and the lower the compliance of the control effect, the closer the value is to 0.

2) For a specific state space $\kappa \in \mathbb{R}^m$, it is the subspace of state space x, i.e., $\kappa \subset x$, and it is composed of specific components of state space x. In this state, there may be specific control requirements or the designer's objective evaluation of the state. We regard κ as an observation perspective of state space. Therefore, we define the evaluation function $e_\kappa(x_k, u_k) \in [0,1]$ under the perspective of κ. In this perspective, specific control can be done and the control effect (x_k, u_k) can be evaluated. As a component of the reward function, the evaluation function under a specific observation perspective is often helpful to guide the follow-up control to better approach the control goal. As shown in Figure 1, in a gradually stable MDP control trajectory, the control vector in state x_k is u_k, the reward value is r_k, and x_k transitions to the next state x_{k+1}. If there is a better control vector u'_k under the observation perspective κ_k a at this time step, the state is directly transitioned to x_{k+i}, bypassing the middle $i-1$ control processes, and the MDP trajectory becomes (x_k, u'_k, r'_k). Therefore, such a control law is suitable for increasing the reward value. According to the different observation perspectives, the evaluation function $e(x_k, u_k)$ related to the state-control pair (x_k, u_k) is formulated to evaluate the control validity. When $i > 0$, $e_i(\cdot)$ has a value range of $[0,1]$ and is continuously differentiable.

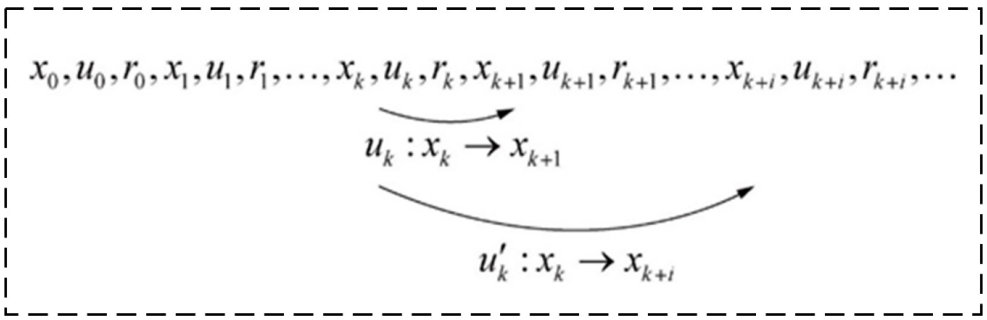


Figure 1. MDP control trajectory.

3) $R(\cdot)$ can be defined as:

$$R(x_k, u_k) \doteq \prod_{i=0}^{n-1} e_i(x_k, u_k), n > 1 \tag{15}$$

and

$$\begin{cases} e_0(x_k, u_k) = r_{max}, & x_k \in \Omega_x \ \ and \ \ u_x \in \Omega_u \\ e_0(x_k, u_k) = 0, & \text{otherwise} \end{cases}. \tag{16}$$

Here $r_{max} \geq 1$ means that the maximum value of the reward function is 1. When $i > 0$, $e_i(\cdot)$ should be continuous and twice differentiable in order to ensure that $R(\cdot)$ is also continuous and twice differentiable if $R(\cdot) > 0$. The evaluation function sequence is $(e_0(x_k, u_k), \ldots, e_{n-1}(x_k, u_k))$. Where n is the number of evaluation functions, i.e., the effectiveness of control is divided into n different observation perspectives for evaluation.

A dynamic environment is constructed by the above formula, as shown in Figure 2. The input parameters are state-control pair (x_k, u_k), and the output parameters are the next state x_{k+1} and $R(x_k, u_k)$ of the system. If $R(\cdot)$ is regarded as the reward value of the dynamic environment, the transformation from the utility function of the NCS to the reward is completed, i.e., the utility function is substituted by the reward value. The description effect may be that the smaller the value of the utility function, the greater the reward value, but the value range of the reward is the interval from 0 to $\mathrm{r_{max}}$.

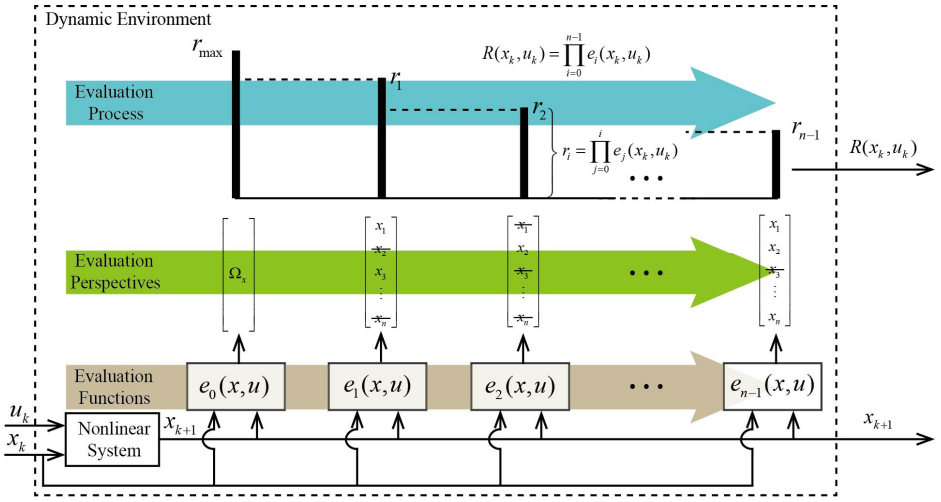


Figure 2. Dynamic environment.

In order to satisfy the Eqs (8)–(11), we can express the relationship between $U(x_k, u_k)$ and $R(x_k, u_k)$ with the following equation:

$$U(x_k, u_k) = e_0(x_k, u_k) - R(x_k, u_k). \tag{17}$$

It is worth mentioning that the n evaluation functions and their respective observation perspectives should be designed consistent with the control objectives. The design of observation perspectives can mainly focus on two points: the first is to monitor the effect of the control law in a specific state during the control process, the second is that this design is conducive to the system to achieve an optimal control effect. The observation perspective of the evaluation function should not

be repeated, and there should be no conflict in control methods between the evaluation functions, which means that the high score of the control law under one observation perspective is easier to guide the control law to obtain high scores under other observation perspectives. However, the high score of the control law under a certain observation perspective may not immediately lead to high scores under other observation perspectives, but this will cause the control law in the current state to get high scores with great probability under more observation perspectives in the next time steps.

According to the definition of expected discounted return in reinforcement learning theory, let $R_k = R(x_k, u_k)$ and bring it into the following equation:

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k, \tag{18}$$

where γ is the discount rate, $0 \leq \gamma \leq 1$, R_k is the reward of step k, $R_k \in \Re \subset \mathbb{R}$, $\Re$ is reward sets, and T is a final time step.

The advantages of using reward function $R(x_k, u_k)$ instead of utility function $U(x_k, u_k)$ are as follows:

1) The reward function reflects the feasible control objectives from various observation perspectives. It may be a nonlinear function instead of the linear expression of the utility function, which can better satisfy the detailed control objectives.

2) The reward function $R(x_k, u_k)$ is composed of several evaluation functions from their respective observation perspectives. Each evaluation function $e(x_k, u_k)$ is specific, simple and easy to calculate, which avoids the problem that it is difficult to specify the weight matrix Q and R in the utility function of $U(x_k, u_k)$.

3) For a specific real control platform, the reward function is easier to realize. According to the requirements of control objectives and control details, it is easier to adjust each evaluation function, and the adjustment of one evaluation function will not affect other evaluation functions.

4) After adding the reward function in the dynamic environment, we can try to use more skillful reinforcement learning algorithms to solve the nonlinear control problem.

In Eq (15), the reward function takes the form of successive multiplications instead of successive additions or others. The advantages of doing this are as following:

1) If the reward function takes the continuous addition form of the evaluation functions, when changing one of the evaluation functions, the weight of each evaluation function must also be considered, which will make the problem very complex, and its rationality needs to be carefully verified in the experiment. In the form of continuous multiplication, if one of the evaluation functions is changed, it only changes the calculation method and the possible gain of the function, and there is no need to change the weights of other evaluation functions. In fact, the output value of each function can be understood as its own weight.

2) Any evaluation function can play the role of "one vote veto" by using continuous multiplication. For example, in the experiment of controlling the robot to walk, no matter how high the score given by the evaluation function that responsible for evaluating the robot's posture, as long as it touches the boundary, it should be given a zero score. According to Eq (15), when the evaluation function responsible for boundary detection gives a zero score, the score of the reward function is zero. But the continuous addition form of the evaluation functions will not bring such an effect.

In order to comply with Eqs (3)–(5) and (11), we try to modify Eq (8) as:

$$J(x_k, \underline{u}_k) = \sum_{i=k}^{\infty} U(x_i, u_i)$$

$$= \sum_{i=k}^{\infty} \left(e_0(x_i, u_i) - R(x_i, u_i)\right)$$

$$= \sum_{i=k}^{\infty} (r_{max} - R(x_i, u_i)). \tag{19}$$

Let

$$\Psi(x_k, \underline{u}_k) = \sum_{i=k}^{\infty} R(x_i, u_i), \tag{20}$$

and we have:

$$J(x_k, \underline{u}_k) = \sum_{i=k}^{\infty} r_{max} - \Psi(x_k, \underline{u}_k). \tag{21}$$

Therefore, as long as $\underline{u}_k$ conforms to the system under (6), it satisfies the optimality equation:

$$\Psi^*(x_k) = \max_{\underline{u}_k} \{\Psi(x_k, \underline{u}_k)\}. \tag{22}$$

Hence $\Psi^*(x_k)$ can be obtained as:

$$\Psi^*(x_k) = \max_{u_k} \{R(x_k, u_k) + \Psi^*(x_{k+1})\}. \tag{23}$$

So the optimal performance index function has another form:

$$J^*(x_k) = \min_{\underline{u}_k} \{J(x_k, \underline{u}_k)\}$$

$$= \sum_{i=k}^{\infty} r_{max} - \Psi^*(x_k). \tag{24}$$

Recalling Eq (10), we get:

$$J^*(x_k) = \min_{u_k} \{U(x_k, u_k) + J^*(x_{k+1})\}$$

$$= r_{max} - R(x_k, u^*(x_k)) + J^*(F(x_k, u^*(x_k))), \tag{25}$$

and the optimal single control law is:

$$u^*(x_k) = \arg \max_{u_k} \Psi^*(x_k). \tag{26}$$

Comparing Eqs (26) and (14), the problem of solving the optimal performance index function is transformed into the problem of maximizing the return function. Further, combined with Eq (18), we can try to use a wider range of reinforcement learning algorithms to solve the optimization problem of nonlinear systems.

For the convenience of analysis, the stability proof is made on the basis of the following assumptions 1 and 2.

Assumption 1: The discrete NCS represented by Eq (6) is controllable, and the system state $x_k = 0$ is an equilibrium state of the discrete NCS under the condition of $u_k = 0$, i.e., $F(0,0) = 0$.

Assumption 2: The feedback control $u_k = u(x_k)$ satisfies $u_k = u(x_k) = 0$ for $x_k = 0$, $\forall x_i \in \Omega_i \subset \mathbb{R}^n$, $u(x_k)$ is continuous and $u(x_k)$ stabilizes the discrete NCS on Ω_i, $J(x_i)$ is finite.

Assumption 1 ensures that the object we use the deep reinforcement learning method to solve is

a controllable NCS, and determines that the equilibrium point of the system is $x_k = 0$, and the output of the system function $F(\cdot)$ is 0. Assumption 2 ensures that the control law is admissible and the output is 0 at the equilibrium point, so as to meet the requirements of a controllable NCS. In the process of control sequence, the performance index function $J(x_i)$ is a bounded function. In this way, we have made restrictive assumptions about the system object and control law.

According to the definition of $e(x_k, u_k)$, $e(x_k, u_k) \geq 0$, then $R(x_k, u_k)$ is a positive definite function for any x_k and u_k. Let $a_0(x_k)$ be an arbitrary admissible control law. According to Q-learning [43], for $i = 0$, let $Q_0(x_k, u_k)$ be the initial iterative Q function constructed by $a_0(x_k)$, i.e.,

$$Q_0(x_k, a_0(x_k)) = \sum_{j=0}^{\infty} R(x_{k+j}, a_0(x_{k+j})). \tag{27}$$

Thus, initial iterative Q function satisfies the following optimality equation:

$$Q_0(x_k, u_k) = R(x_k, u_k) + Q_0(x_{k+1}, a_0(x_{k+1})). \tag{28}$$

Then, we get the iterative control law as follows:

$$a_1(x_k) = \arg\max_{u_k} Q_0(x_k, u_k). \tag{29}$$

Let $Q_i(x_k, u_k)$ be the iterative Q function constructed by $a_i(x_k)$ for $i = 1,2,\ldots$, then we can get the following generalized optimality equation:

$$Q_i(x_k, u_k) = R(x_k, u_k) + Q_i(x_{k+1}, a_i(x_{k+1})). \tag{30}$$

Therefore, the iterative control law is updated by:

$$a_{i+1}(x_k) = \arg\max_{u_k} Q_i(x_k, u_k). \tag{31}$$

Because the reward function $R(x_k, u_k)$ is a positive definite function of x_k and u_k, and under Assumption 1 and 2, the iterative function $Q_i(x_k, u_k), i = 0,1,\ldots$, is positive definite for x_k and u_k. According to Eq (30), substitute $a_i(x_k)$ for u_k, and let $V_i(x_k) = Q_i(x_k, a_i(x_k))$ for $i = 0,1,\ldots$, we can get:

$$\begin{aligned} V_i(x_{k+1}) - V_i(x_k) &= Q_i(x_{k+1}, a_i(x_{k+1})) - Q_i(x_k, a_i(x_k)) \\ &= -R(x_k, a_i(x_k)) < 0. \end{aligned} \tag{32}$$

For $i = 0,1,\ldots$, the function $V_i(x_k)$ is positive definite for x_k, so $V_i(x_k)$ is a Lyapunov function. Thus, $a_i(x_k)$ is a stable control law.

In the above, it is difficult to give a specific expression form for the control objective of the utility function under the actual control system. We can try to reconstruct the reward function $R(x_k, u_k)$ to replace the utility function with the help of DRM. Therefore, the key step of solving nonlinear optimization problem is transformed into the problem of how to construct a reinforcement learning dynamic environment and optimize the reward function $R(x_k, u_k)$. According to Eqs (15) and (16), the determination of evaluation function is one of the important tasks of DRM. Ng et al. [44] proposed the reward shaping theory, if a potential energy-based function is added to the reward function, the optimal strategy remains unchanged, realizing the optimal strategy of modifying the reward without affecting the Markov Decision Process. The reward shaping method is used in

multi-agent reinforcement learning, and it is effective in robots collaborative-competitive operations and multi-objective tasks [45–49]. This approach give us some inspiration, especially in the construction of the potential energy function and how to avoid reward conflicts.

4. Experimental schemes based on reinforcement learning

Using the mathematical models above, the problem of obtaining a control method for a nonlinear system can be transformed into a problem of training one or more agents' control strategies, establishing a value function and obtaining the maximum reward. In this section, by choosing the inverted pendulum system as the experiment object, we will analyze its characteristics and establish the dynamic environment, design the reward function through different control objectives, then use three deep reinforcement learning algorithm models to conduct experiments. the experiments result is discussed finally.

4.1. Description of the inverted pendulum system model

Figure 3 is a schematic diagram of the structure of the inverted pendulum system. Our ultimate goal is to control the force exerted to the cart and make it move left or right to maintain the balance of the single pole mounted on the cart. The typical parameters of inverted pendulum-cart system setup are selected as: mass of the cart (m_c): 1.34 kg, mass of the pendulum (m_p): 0.09 kg, length of the swing pole (l): 0.40 m, length of the cart rail (l_r): 0.40 m, friction coefficient of the cart & pole rotation is assumed negligible. The acceleration due to gravity g = 9.81 m/s^2.

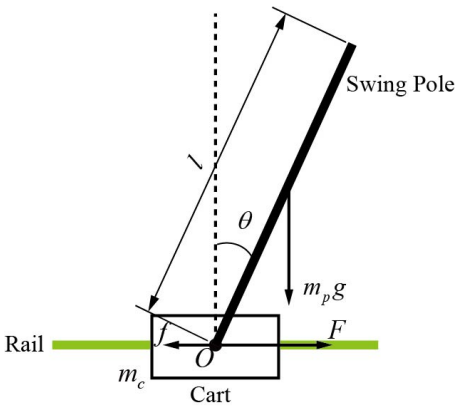


Figure 3. Schematic diagram of the structure of the inverted pendulum system.

In [50], the deep reinforcement learning algorithm model is used to realize the balance of the cart-pole in the inverted pendulum system, but the relevant control theory in the NCS is not involved, and the design of the reward function is not introduced in detail. The system function of the inverted pendulum system model in this paper is expressed as follows [51,52]:

$$\ddot{\theta} = \frac{gm\sin\theta - \cos\theta\left(F + m_p l\dot{\theta}^2\sin\theta\right)}{lm\left(4m/3 - m_p\cos^2\theta\right)}, \tag{33}$$

$$\ddot{\chi} = \frac{F + m_p l\left(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right)}{m}, \tag{34}$$

where $m = m_c + m_p$, $\ddot{\chi}$ is the moving speed of the cart on the rail, $\ddot{\theta}$ is the swing angular speed of the swing pole. The nonlinear system equations (33) and (34) of inverted pendulum dynamic system represent the control input affine system. These two nonlinear equations are represented in the following standard state space form:

$$\dot{x}(t) = f\big(x(t)\big) + g\big(x(t)\big)u(t). \tag{35}$$

4.2. Reward function design

For Eq (35), Let $x(t) = (\chi, \dot{\chi}, \theta, \dot{\theta})^T$, and set θ_{max} as the maximum limit of the swing pole deviation from the balance angle, Let χ_{max} be the maximum value that the cart deviates from the origin of the guide rail, $\dot{\chi}_{max}$ the maximum speed of the cart, and $\dot{\theta}_{max}$ the maximum rotation speed of the swing pole. The definition rules of the system state space and evaluation function are as following:

$$\Omega_x = \begin{bmatrix} -\frac{l_r}{2} \le \chi \le \frac{l_r}{2} \\ -\dot{\chi}_{max} \le \dot{\chi} \le \dot{\chi}_{max} \\ -\theta_{max} \le \theta \le \theta_{max} \\ -\dot{\theta}_{max} \le \dot{\theta} \le \dot{\theta}_{max} \end{bmatrix}, \tag{36}$$

$$e_0(x,u) = \begin{cases} 1, x \in \Omega_x \\ 0, x \notin \Omega_x \end{cases}. \tag{37}$$

The range of the output value of the evaluation function e_0 is [0,1]. In order to describe the expected position of the cart on the rail and guide the cart to run at the equilibrium position of the rail with greater probability, we make an evaluation function as following:

$$e_1(x,u) = 1 - \mathrm{sgn}(\chi)\frac{2\chi}{l_r}. \tag{38}$$

From the perspective of control, the cart "has to" drive the swing pole into a better state space at some positions of the rail. In this way, we can establish an evaluation function to better guide the system into the equilibrium state. If it is said that the desired balance angle is zero in the equilibrium state, then at different positions of the rail, we assign position-related dynamic balance angles to guide the movement of the cart. The position-related balance angle is given by:

$$\theta_0 = -\frac{2\chi\theta_{max}}{l_r}. \tag{39}$$

Then the evaluation function is established as:

$$e_2(x,u) = \begin{cases} 1 - \frac{|\theta - \theta_0|}{\theta_{max}}, & |\theta - \theta_0| \le \theta_{max} \\ 0, & |\theta - \theta_0| > \theta_{max} \quad or \quad \theta > \theta_{max} \quad or \quad \theta < -\theta_{max} \end{cases}. \tag{40}$$

Notice that the output range of $e_2(\cdot)$ is [0,1], and θ_{max} is the maximum allowable deflection angle of the swing pole.

In the balancing process of inverted pendulum system, not only the system is required to reach the equilibrium state in a limited time, but also the energy consumed should be considered. It is

expected that the energy consumed will reach the minimum in the process of state transition under two adjacent time steps. Therefore, we make an evaluation function as follows:

$$e_3(x,u) = \eta(x,u)\varphi(x), \tag{41}$$

where, $\eta(x,u)$ is the power consumption function of the system under the action of control output u in x state; $\varphi(x)$ is the coefficient function in x state. Without considering the external disturbance of the system such as air resistance, the force power consumption function can be expressed by the following formula:

$$\eta(x_k,u_k) = W\left(x_{k+1}^{u=F}\right) + \varsigma\left(x_{k+1}^{u=F}\right) - W\left(x_{k+1}^{u=0}\right) - \varsigma\left(x_{k+1}^{u=0}\right), \tag{42}$$

here, $W(x_{k+1}^{u=0})$ is defined as the energy loss after the system transitions to the next state when there is no action output control in state x_k; in the same case of no action output, $\varsigma(x_{k+1}^{u=0})$ is the energy loss caused by system friction. $W(x_{k+1}^{u=F})$ is defined as the energy loss after the system transitions to the next state under the action of output control $u = F$ in state x_k, and $\varsigma(x_{k+1}^{u=F})$ is the energy loss caused by system friction after the system transitions to the next state under the action of output control $u = F$ in state x_k.

In the system represented by Eqs (33) and (34), friction is not considered, so $\varsigma(x_{k+1}^{u=0}) = \varsigma(x_{k+1}^{u=F}) = 0$. Therefore, in state x_k, the system energy is:

$$W(x_k) = T_c(x_k) + T_p(x_k) + V_p(x_k), \tag{43}$$

where $T_c(x_k)$ is the kinetic energy of the cart, $T_p(x_k)$ is the kinetic energy of the swing pole, and $V_p(x_k)$ is the potential energy of the swing pole. According to the inverted pendulum model, we can get:

$$T_c(x_k) = \frac{1}{2}m_c\dot{\chi}_{x_k}^2, \tag{44}$$

$$\begin{aligned} T_p(x_k) &= \frac{1}{2}m_p\left(\left(\frac{d(\theta_{x_k}+l\sin\theta_{x_k}/2)}{dt}\right)^2 + \left(\frac{d(l\cos\theta_{x_k}/2)}{dt}\right)^2\right) + \frac{1}{6}m_p l^2\dot{\theta}_{x_k}^2 \\ &= \frac{1}{2}m_p\dot{\chi}_{x_k}^2 + \frac{1}{2}m_p l\dot{\chi}_{x_k}\dot{\theta}_{x_k}\cos\dot{\theta}_{x_k}^2 + \frac{7}{24}m_p l^2\dot{\theta}_{x_k}^2, \end{aligned} \tag{45}$$

$$V_p(x_k) = \frac{1}{2}m_p g l\cos\theta_{x_k}, \tag{46}$$

where θ_{x_k} is the swing pole angle in state x_k. According to the above formulas, calculate $W(x_{k+1}^{u=0})$ and $W(x_{k+1}^{u=F})$ respectively, and then calculate $\eta(x_k,u_k)$. Note the coefficient function $\varphi(x_k)$, which reflects the user's specific allocation method of energy in combination with the current state. For example, it is required to minimize the single-step energy, or establish the minimum theoretical value of energy according to the current state, and then require the single-step energy to approach the value as much as possible, and ensure that the final evaluation function is between 0 and 1.

Note that in the control process of the system, $e_0(x,u)$ gives the maximum value of evaluation 1, $e_1(x,u)$ is the evaluation function of cart displacement χ in the state vector,

$e_2(x,u)$ is the evaluation function of χ and θ in the state vector, and $e_3(x,u)$ is the evaluation function of θ, $\dot{\chi}$ and $\dot{\theta}$ in the state vector. The final evaluation effect is expressed as:

$$R(x_k,u_k)=\prod_{i=0}^{3}e_i(x_k,u_k). \tag{47}$$

$R(x_k,u_k)$ is a nonlinear description that acts as a utility function, which is different from the description method of Eq (8). In other words, $R(x_k,u_k)$ reflects the user's specific control needs in a more subtle way. It is commendable that users can add, modify the details of specific control requirements via the evaluation function, and the modification of a certain evaluation function will not affect the roles of other evaluation functions. In the dynamic environment, the relationship between the evaluation functions and the reward function is that the reward function is like composed of multiple "referees", and these "referees" are the evaluation functions. Each referee gives a score ranging from 0 to 1, and then the reward function summarizes the scores of each referee by means of continuous multiplication. The scoring standard for each referee is based on their different viewing angles, and the viewing angles of each referee cannot be the same. In fact, the viewing angles are their observation perspectives as shown in Figure 2. As mentioned above, no evaluation function can violate the control law of the system, which means that the referee is required to guide and encourage the effect of "performer", in other words, the performer (i.e., the control law) follows the performance guidance effect of a referee's high score, which is easier to achieve the final effect of control. This final effect is the best one close to r_{max} recognized by each referee.

4.3. Deep reinforcement learning model analysis

In this paper, $R(\cdot)$ is regarded as the reward value of the dynamic environment, and the deep reinforcement learning algorithm is designed to interact with the dynamic environment from three perspectives of state-value, policy and "Actor-Critic". During the interaction process, the agent's execution strategy is optimized through the reinforcement learning algorithm to achieve the maximum reward value. Among them, the deep reinforcement learning algorithm based on state-value mainly takes Deep Q-Learning (DQN) as an example, the policy-based algorithm mainly takes the policy-gradient reinforcement algorithm as an example, and the "Actor-Critic" framework mainly takes Deep Deterministic Policy Gradient as an example. Three deep reinforcement learning models are analyzed and described below.

4.3.1. Deep Q-learning

DQN is a value-based iteration reinforcement learning algorithm [53]. When the state and action space generated during the interaction between the agent and the environment is discrete and the dimension is not high, the Q-Learning method can be used to establish the state-action value Q table. Update the Q-values in the table according to the Bellman optimality equation. In the inverted pendulum system, the states of the cart and the swing pole are high-dimensional continuous. Using the Q-Learning method to build a table is very difficult, and it often causes the "curse of dimensionality". When the states and actions are high-dimensional and continuous, the construction table can be transformed into a function fitting problem, and the Q value is generated by fitting the function instead of the table, so that similar states can obtain similar output actions. In the function

fitting of high-dimensional input, the advantage of deep neural network for complex feature extraction can be used to map the input to the output through neural network. Therefore, the essence of the DQN method is to combine the deep neural network and Q-Learning, map the state action value Q through the network structure, and update the agent action strategy according to the Bellman optimal equation. Algorithm 1 shows the DQN algorithm flow.

Algorithm1: Deep Q-Learning

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights w

Initialize target action-value function $\hat{Q}$ with weights $w^- = w$

1. For episode=1, M do
2. Initialize sequence $s_1 = \{s_1\}$ and preprocessed sequence $\emptyset_1 = \emptyset(s_1)$
3. For t=1, T do
4. With probability ε select a random action a_t
5. Otherwise select $a_t = argmax_a Q(\emptyset(s_t), a; w)$
6. After the agent performs action a_t,it gets immediate reward r_t and next state $s_{t+1}\{\chi_{t+1}, \theta_{t+1}, \dot{\chi}_{t+1}, \dot{\theta}_{t+1}\}$
7. Set $\emptyset_t = \emptyset(s_t)$, $\emptyset_{t+1} = \emptyset(s_{t+1})$,and store transition $(\emptyset_t, \emptyset_{t+1}, a_t, r_t)$ in D
8. Sample random minibatch of transitions $(\emptyset_j, \emptyset_{j+1}, a_j, r_{j+1})$ from D
9. Set $y_j = \begin{cases} r_{j+1} & if\ eposide \text{ terminates at step } j+1 \\ r_{j+1} + \gamma max_{a'} \hat{Q}(\emptyset_{j+1}, a'; w^-) & otherwise \end{cases}$
10. Perform a gradient descent step on $(y_j - Q(\emptyset_j, a_j; w))^2$ with respect to the network parameters w
11. After each step C is executed, the parameter w of the current action-value network Q is assigned to the parameter $w^- of\ the$ action-value target network $\hat{Q}$, $Q^- = Q$
12. end for
13. end for

In algorithm 1, $s_t\{\chi_t, \theta_t, \dot{\chi}_t, \dot{\theta}_t\}$ is the state space of the inverted pendulum system at time t. Among them, χ_t is the displacement of the cart, θ_t is the angle of the swing pole, $\dot{\chi}_t$ is the speed of the cart, and $\dot{\theta}_t$ is the angular speed of the swing pole. r_t is the reward value obtained during the interaction with the dynamic environment at time t.

DQN records the tuple $\{s_t, a_t, r_{t+1}, s_{t+1}\}$ generated by the agent in each state and stores it in the experience replay. The deep neural network randomly selects the tuple in the experience replay as the training label to update the network parameters. Randomly selecting previous experience in the experience replay will make the network more efficient, solving the problem of correlation and non-static distribution to a certain extent [54].

DQN adopts two network structures with the same structure but different parameters, namely

action-value network (Q-network) and target action-value network (target Q-network). When the action-value network is used to approximate the Q value, the update of the Q value is prone to oscillation, resulting in unstable learning behavior of the agent. The target action-value network reduces the correlation between the predicted Q value and the estimated Q value to a certain extent to improve the stability of the algorithm. In the network parameter update process, the parameters used by the Q-network are the latest parameters of each update, and the Q-network parameters are copied to the target Q-network after multiple iterations.

The agent maps the estimated Q value through Q-network at s_t, uses the epsilon-greedy [55] strategy to select the Q value under the corresponding action, takes the action a_t corresponding to the Q value to interact with the environment, and records the generated tuple $\{s_t, a_t, r_{t+1}, s_{t+1}\}$ and stores it in the experience replay. The target Q-network maps s_{t+1} as input to the predicted Q value, selects the sum of the maximum predicted Q value and the immediate reward r_{t+1} as the TD target[56], uses the mean square error between the TD target and the Q value as a loss function for updating network parameters. The loss function between target action-value function and action-value function in DQN is as follows:

$$Target_{Q=r_{t+1}} + \gamma max_{a'} \hat{Q}_{w^-}(s_{t+1}, a'), \tag{48}$$

$$L(\theta) = E_{\{s_t, a_t, r_{t+1}, s_{t+1}\} \epsilon N}\left[\left(Target_Q - Q_w(s_t, a_t)\right)^2\right]. \tag{49}$$

4.3.2. Reinforce: Monte Carlo Policy Gradient

In the DQN series of reinforcement learning algorithms, we mainly approximate the value function, learn the action value function based on the value, and then adopt a similar greedy strategy to select the agent output action according to the estimated action value function. But the DQN series of reinforcement learning algorithms have the following problems: 1) Algorithms based on value functions can only process discrete actions, but cannot process continuous actions output by the agent. 2) The algorithm based on the value function cannot solve the stochastic policy problem. The optimal strategy corresponding to the DQN series of methods is a deterministic strategy, and the action corresponding to the maximum value is taken as the optimal strategy. Aiming at the shortcomings of value-based reinforcement learning algorithms, a policy-based reinforcement learning algorithm, policy gradient, is introduced [57].

The following is the state value function gradient formula in Monte Carlo sampling, where $\Pr(s \to x, k, \pi)$ is the probability of transitioning from state s to state x in k steps under policy π:

$$\nabla v_\pi(s) = \nabla[\textstyle\sum_a \pi(a|s) q_\pi(s,a)]$$

$$= \textstyle\sum_a [\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a)]$$

$$= \textstyle\sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)\,(r + v_\pi(s'))\right]$$

$$= \textstyle\sum_a [\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a)\, \nabla v_\pi(s')]$$

$$= \textstyle\sum_{x \in S} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a). \tag{50}$$

$\nabla v_\pi(s)$ in Eq (50) is converted as follows:

$$\nabla J(w) = \nabla v_\pi(s_0)$$

$$= \sum_s \sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

$$= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

$$= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

$$= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

$$\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

$$= E_\pi[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,w)]. \quad (51)$$

$\nabla J(w)$ is the gradient of the policy parameters, $\mu(s)$ is the policy distribution under the policy π, and the learning goal of the policy parameters is to maximize the value of $J(w)$, so the update of the policy parameters is similar to solving the gradient of $J(w)$,which is:

$$w_{t+1}$$

$$= w_t + \alpha \nabla J(w_t)$$

$$= w_t + \alpha E_\pi[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,w)]$$

$$= w_t + \alpha \sum_a \hat{q}\left(S_t, a, w`\right) \nabla \pi(a|S_t,w) \ . \quad (52)$$

In the above equation, $w`$ and w are the parameters in the action value network and the policy network, respectively. In the Monte Carlo Policy Gradient, the sampling A_t is used to replace a under the policy π. Therefore, $\nabla J(w)$ can be defined as:

$$\nabla J(w) = E_\pi[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,w)]$$

$$= E_\pi\left[\sum_a \pi(a|S_t,w)\, q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,w)}{\pi(a|S_t,w)}\right]$$

$$= E_\pi\left[q_\pi(S_t,A_t) \frac{\nabla \pi(A_t|S_t,w)}{\pi(A_t|S_t,w)}\right]$$

$$= E_\pi\left[G_t \frac{\nabla \pi(A_t|S_t,w)}{\pi(A_t|S_t,w)}\right]. \quad (53)$$

G_t represents the reward obtained in the S_t state, $G_t \frac{\nabla \pi(A_t|S_t,w)}{\pi(A_t|S_t,w)}$ can be calculated by sampling, the expected value is equal to the gradient. Introducing the above gradient into the formula $w_{t+1} = w_t + \alpha \nabla J(w_t)$, we get:

$$w_{t+1} \doteq w_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,w_t)}{\pi(A_t|S_t,w_t)}$$

$$= w_t + \alpha G_t \nabla ln\pi(A_t|S_t,w_t). \quad (54)$$

Algorithm 2 shows the process of the Monte Carlo Policy Gradient reinforce algorithm. In algorithm 2, $s = \{\chi, \theta, \dot{\chi}, \dot{\theta}\}$ represents the state space in the inverted pendulum system, and r represents the immediate reward obtained when the state is transferred, which is equivalent to $R(x_k, u_k)$.

Algorithm2: REINFORCE:Monte-Carlo Policy-Gradient Control

Input: a differentiable policy parameterization $\pi(a|s,w)$

Set the algorithm parameter α, indicating the network update range: $\alpha > 0$

Initialize policy network with random weights w

1.Use π_w to collect agents to generate trajectories

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2 \dots, a_H, r_{H+1}, s_{H+1})$$

2.Estimate the return generated by trajectory τ: $R(\tau) = (G_0, G_1, \dots, G_H)$

3.Where G_k represents the expected return when s_k:

$$G_k \leftarrow \sum_{t=k+1}^{H+1} \gamma^{t-k-1} R_k$$

4.Use trajectory τ to estimate gradient $\nabla_w J(w)$

$$\nabla_w J(w) \leftarrow \sum_{t=0}^{H} \nabla_w ln\pi_w(a_t|s_t) G_t$$

5.Update parameter w according to gradient ascent algorithm

$$w \leftarrow w + \alpha \nabla_w J(w)$$

6. Loop steps 1 to 5 to update network parameters

4.3.3. Deep deterministic policy gradient

According to the output action characteristics of the agent in reinforcement learning, strategies can be divided into deterministic strategies and random strategies. Deterministic policy means that the action of the agent output through the policy in a certain state is deterministic. The random strategy refers to the probability that the agent outputs multiple actions through the strategy in a certain state, and the agent chooses to execute the action according to the probability. In the DQN series of methods, the greedy strategy is used to select the action corresponding to the maximum Q value, which belongs to the greedy deterministic strategy. When the action set output by the agent is continuous-valued or discrete-valued with very high dimensionality, we use a stochastic strategy to study the probability of action output, which increases the amount of computation. The Deep Deterministic Policy Gradient (DDPG) algorithm is often used to solve the action selection problem in a high-dimensional continuous state space [58]. The algorithm is tuned on the basis of the Actor-Critic framework, making full use of DQN and policy gradients. The DDPG algorithm uses a training network and a target network in the design of the network structure. Through the target

network, the correlation between the predicted value and the estimated value can be reduced to a certain extent, and the stability of the algorithm can be improved. Training the network with samples from the experience replay minimizes the correlation between samples.

The DDPG algorithm mainly includes the following parts: ① The network includes two parts, Actor and Critic, in which Actor and Critic are composed of training network and target network respectively. ② Introduce the experience replay to store the data of the interaction between the agent and the environment, and use soft update to update the parameters of the target network to ensure the stability of the training network [59]. ③ Add random noise to the Actor network output to ensure that the agent has a certain exploration ability when selecting actions [60]. Algorithm 3 shows the process of the DDPG algorithm.

Algorithm3: DDPG algorithm

Randomly initialize:
critic $Q(s,a|w^Q)$ and actor $\mu(s,a|w^\mu)$ with weights $w^Q$and w^u

Initialize target network Q' and μ' with weights $w^{Q'} \leftarrow w^Q$, $w^{\mu'} \leftarrow w^\mu$

Initialize replay buffer D

1. For episode=1, M do
2. Initialize a random process $\mathcal{N}$ for action exploration
3. Receive initial observation state $s\{\chi,\theta,\dot{\chi},\dot{\theta}\}$
4. For t=1, T do
5. Selectn action $a_t = \mu(s_t \mid w^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
6. Execute action a_t and observe reward r_{t+1} and observe new state s_{t+1}
7. Store transition$(s_t, a_t, r_{t+1}, s_{t+1}, \delta_t)$ in D
8. Sample a random minibatch of N transitions $(s_i, a_i, r_{i+1}, s_{i+1})$ from D
9. Set $y_i = r_{i+1} + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid w^{\mu'}) \mid w^{Q'})$
10. Update critic by minibatch the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i \mid w^Q))^2$
11. Update the actor policy using the sampled gradient:

$$\nabla_{w^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|w^Q)|_{s=s_i,a=\mu(s_i)} \nabla_{w^\mu}\mu(s|w^\mu)|_s$$

12. Update the target network:

$$\omega' \leftarrow \tau\omega + (1-\tau)\omega'$$
$$w' \leftarrow \tau w + (1-\tau)w'$$

13. end for
14. end for

In algorithm 3, $\mathcal{N}$ is random noise, $\mu(s|w^\mu)$ is the Actor network, $Q(s,a|w^Q)$ is the Critic network, $\mu'(s|w^{\mu'})$ is the target Actor network and $Q'(s,a|w^{Q'})$ is the target critic network. The agent obtains the execution action a_t through the network μ at s_t, and interacts with the environment after adding noise to the action to generate an immediate reward r_{t+1}, and the state is updated to s_{t+1}. The agent stores the tuple $\{s_t, a_t, r_{t+1}, s_{t+1}\}$ in the experience replay. When the experience replay data reaches a certain amount, it samples data from the experience replay to train

the network Q. Q-Network needs to evaluate the actions made by μ. Therefore, Q-Network uses the deterministic policy gradient method to optimize the network parameters of μ, and the network parameter replication process adopts a soft update method. The agent keeps repeating the above process until the policy network converges.

The target action value function in the DDPG algorithm can be expressed as:

$$y_i = r_{i+1} + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid w^{\mu'}\right) \mid w^{Q'}\right). \tag{55}$$

The loss function of the Critic network is:

$$L(w) = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i \mid w^Q))^2. \tag{56}$$

Using the deterministic sampling strategy gradient to update the policy network parameter can be expression as:

$$\nabla_{w^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|w^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{w^\mu}\mu(s|w^\mu)|_s. \tag{57}$$

4.4. Comparison and analysis of experimental results

This section describes and analyzes the experimental results. We use three reinforcement learning algorithms to train the inverted pendulum control model, and analyze the differences between the different algorithms based on their training data and test data (Section 4.4.1). Finally, we discuss the current research deficiencies and propose future research directions (Section 4.4.2).

4.4.1. Deep reinforcement learning algorithm validation and performance evaluation

This section compares the experimental results of three deep reinforcement learning algorithms (DQN, policy-gradient, DDPG) in the dynamic environment of the inverted pendulum system, including the impact of the DRM settings on the experimental accuracy, and the differences between different algorithms. The experiment is divided into two parts: i) The influence of the setting of the detail-reward function under three deep reinforcement learning algorithms on the experimental accuracy in the training and testing phase. ii) Differences between three deep reinforcement learning algorithms under the same detail-reward function. In the dynamic environment of the inverted pendulum system, the initial states of the inverted pendulum system in the training and testing phase are shown in Table 1.

1) Experimental comparison of reward functions with different details under the same algorithm

Tables 2 and 3 show the reinforcement learning parameters and network parameters of the DQN algorithm in the inverted pendulum system.

In Table 3, we set the number of outputs of the action-value Q network equal to 25. It can be seen from reference [61] that the richer the choice of control vector $u(k)$, the higher the controllability, but if it is too large, the convergence effect of the network model becomes worse in the training process. Therefore, reasonable selection of the number of action values output by Q network can improve the controllability of inverted pendulum system without affecting the effect of training.

Table 1. Initial state parameters of the inverted pendulum system.

experimental phases	Cart position (/m)	Cart speed (m/s)	Swing pole angle (/rad)	Swing pole speed (rad/s)
Training phase	(−0.2,0.2)	(−0.05,0.05)	(−0.174,0.174)	(−0.05,0.05)
Testing phase	−0.12	0.2	0.1	0.2

*Note: In the training phase of the inverted pendulum system, the initial state of the cart and swing pole is a random number within the corresponding parameter range. In the experiment, the position of the cart and the angle unit of the swing pole are the same as those in the table.

Table 2. Deep reinforcement learning parameters.

Episodes	Steps	Discount factor	Epsilon(ε)	Memory
2000	300	0.95	0.9	2000

Table 3. Neural network parameters.

Network	Network structure	Learning rate	Loss function	Activation function	Batch size
Action-value Q network	(4,128) (128,128) (128,25)	0.0002	MSE	RELU	32

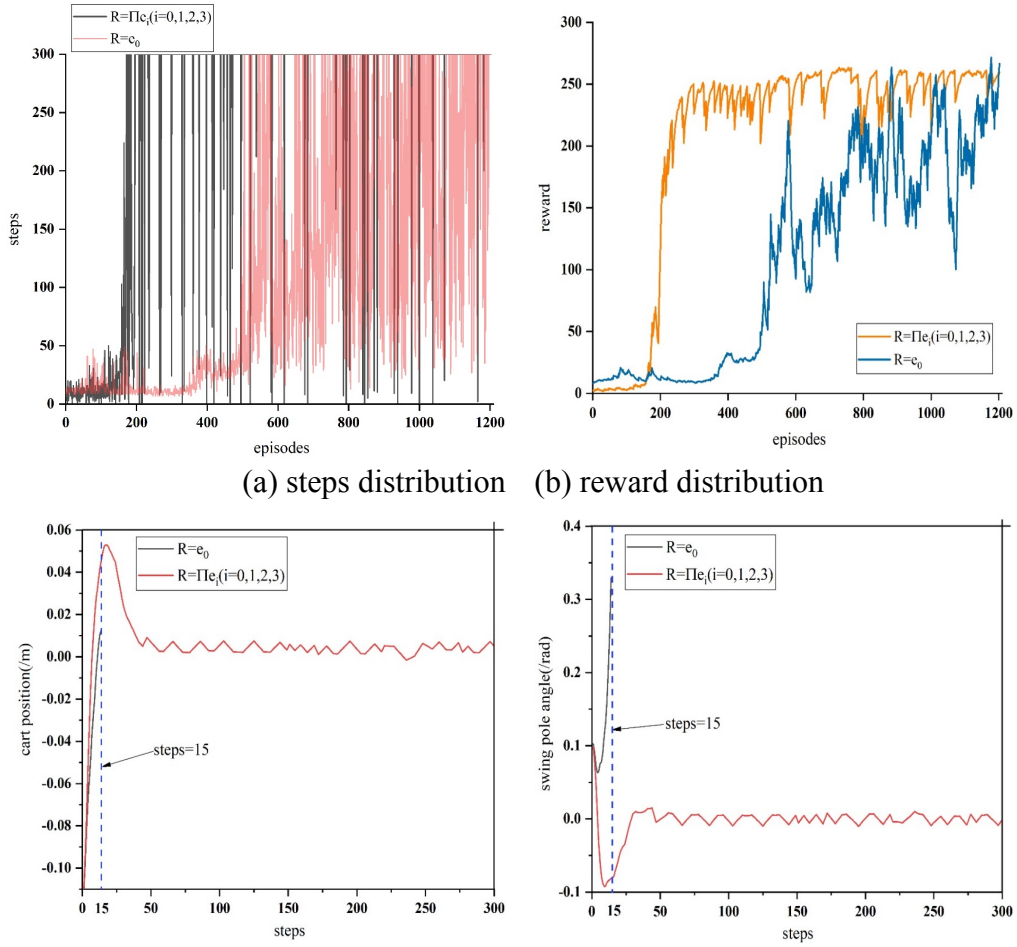
We first give an illustration of the agent training rules for the following experiments. Episode is a round of agent training, and in each episode, the maximum number of times the agent is allowed to learn is 300. The setting range of cart position is [−0.2, 0.2], and the setting range of the swing pole angle is [−0.174, 0.174]. During the training of the agent, as long as the cart and the swing pole are in the areas of the set values, the agent can continue to learn in this episode, and the step value is increased by 1 while the sum of the reward value is also added to the instant reward value, otherwise the learning round will be exited, and the sums of steps and rewards under each episode is recorded.

Figure 4 shows the training and testing process of an inverted pendulum system using the DQN algorithm in a dynamic environment. Figure 4(a) shows the number of admissible control actions output by the inverted pendulum system agent under each episode in the training process. Figure 4(b) shows the cumulative sum of the reward values obtained by the agent under each episode during the training process. Figure 4(c),(d) show the experimental results in the testing phase. Figure 4(c) shows the position of the cart at each step. Figure 4(d) shows the swing pole angle at each step. In the inverted pendulum system, the cart should be controlled to be near the midpoint of the rail ($\chi = 0$) and the swing pole angle to be near the 0 degree ($\theta = 0$).

As can be seen from Figure 4, the setting of the DRM affects the reinforcement learning effect of the inverted pendulum. In Figure 4(a),(b), a single-reward function (i.e., $R = e_0$) is used, when the episode value is less than 400, the number of times the rule is met is less than 50, and the reward value under each episode is less than 50. When the number of episodes is greater than 400, the reward value gradually increases, but the reward distribution curve fluctuates greatly, the generalization of the trained model is weak, and the agent cannot learn useful experience well. When detail-reward function (i.e., $R = \prod e_i$) is used, the number of times the rule is met is proportional to the number of the episodes. In Figure 4(b), when the episodes are greater than 200, the training process tends to converge, the reward value is close to 250, and the reward distribution curve is relatively flat.

In Figure 4(c),(d), when steps = 15 and a single-reward function is used, the position of the cart is 0.01 and the swing pole angle is 0.33, which means that the agent control fails. After 40 steps, by using detail-reward function, the position of the cart is close to zero, the swing pole angle is close to zero degrees, and the distribution of the position and angle curves is relatively gentle.

The above analysis of the distribution curves of steps, rewards, cart position, and swing pole angle in Figure 4 shows that when using the DQN algorithm to train an agent in a dynamic environment of inverted pendulum system, the experimental effect of detail-reward function is better than a single-reward function.



(a) steps distribution (b) reward distribution

(c) cart position (d) swing pole angle

Figure 4. The training and testing results of DQN.

Tables 4 and 5 show the reinforcement learning parameters and network parameters of the policy gradient algorithm in the inverted pendulum system. Figure 5 shows the training and testing process of an inverted pendulum system using the policy gradient algorithm in a dynamic environment.

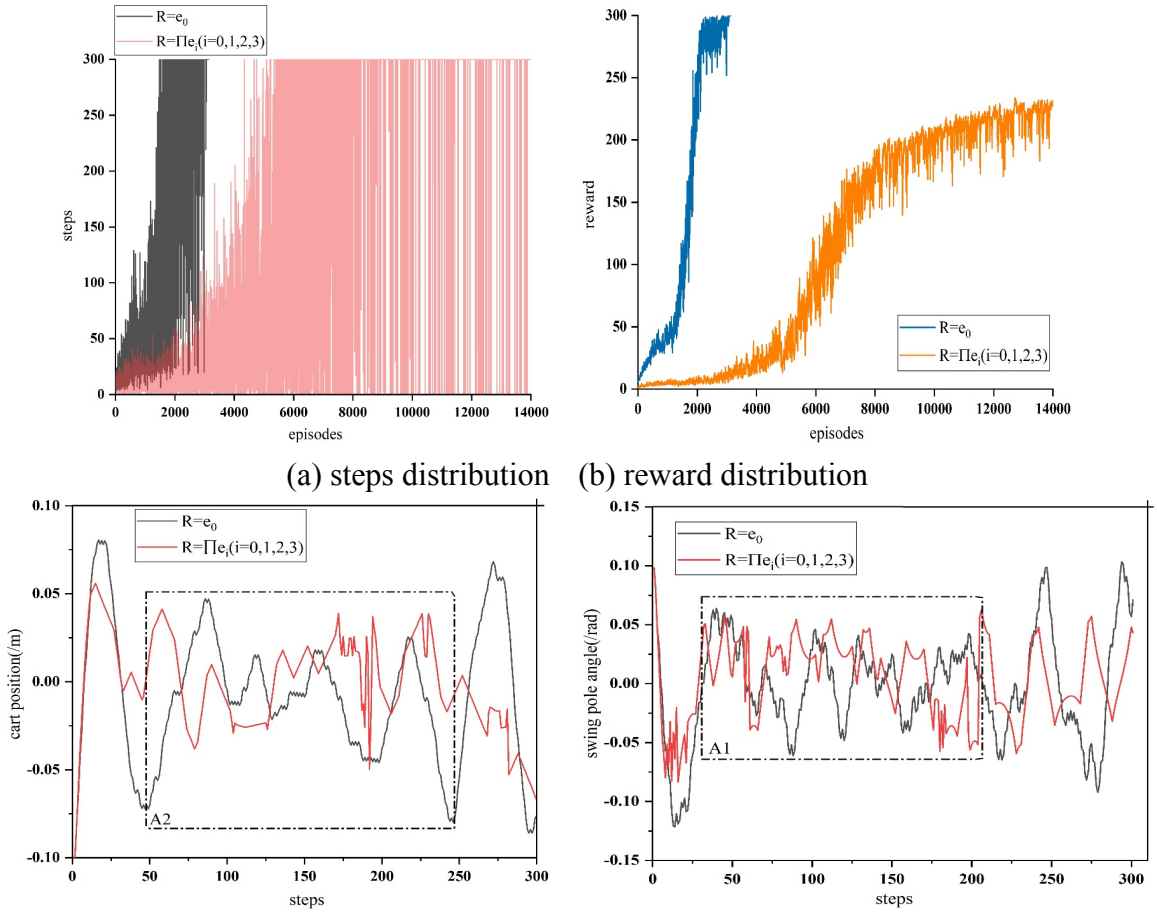
Table 4. Deep reinforcement learning parameter.

Episodes	Steps	Discount factor
15000	300	0.99

Table 5. Neural network parameters.

Network	Network structure	Learning rate	Loss function	Activation function
Policy network	(4,128) (128,128) (128,3)	1×10^{-3}	Cross entropy	Adam

As can be seen from Figure 5(a),(b), the training process converges faster when a single-reward function is used. When the episode is equal to 3000, the step and the reward reach the maximum value of 300 respectively; when using the detail-reward function, the training process converges slowly and the reward rises more gently. When the number of episodes is 14,000, the step and reward reach their maximum values of 300 and 250 respectively. According to Eq (37), a single-reward function means that only the evaluation function $e_0(\cdot)$ plays a role, and its values are only 0 and 1. The final value range of the detail-reward function is [0, 1]. Therefore, the maximum values of reward in these two cases are different.



(a) steps distribution (b) reward distribution

(c) cart position (d) swing pole angle

Figure 5. The training and testing results of policy gradient.

It area A2 of Figure 5(c), when the detail-reward function is used, the position curve distribution of the cart is relatively flat, the minimum position is −0.05, and the maximum position is 0.04. While using a single-reward function, the cart position distribution curve oscillates greatly, the minimum

position and maximum position are −0.077 and 0.05 respectively. It can be seen from area A1 of Figure 5(d) that when the detail-reward function is used, the variation range of the swing pole angle is [−0.05, 0.05]. When the single-reward function is used, the range of the swing pole angle is [−0.07, 0.06], and the data illustrate the swing angle range of the former is smaller than that of the latter.

By analyzing the distribution curves of step, reward, cart position and swing pole angle in Figure 5, we can get: When an agent is trained with the policy-gradient algorithm, the single-reward function makes the training rounds less than the detail-reward function, but in the test phase, the experimental effect of detail-reward function is better than that of single-reward function.

Tables 6 and 7 show the reinforcement learning parameters and network parameters of the DDPG algorithm in the inverted pendulum system. Figure 6 shows the training and testing process while using the DDPG algorithm in a dynamic environment.

Table 6. Deep reinforcement learning parameters.

Episodes	Steps	Discount factor	Soft update coefficient	Memory
2000	300	0.99	1×10^{-2}	10,000

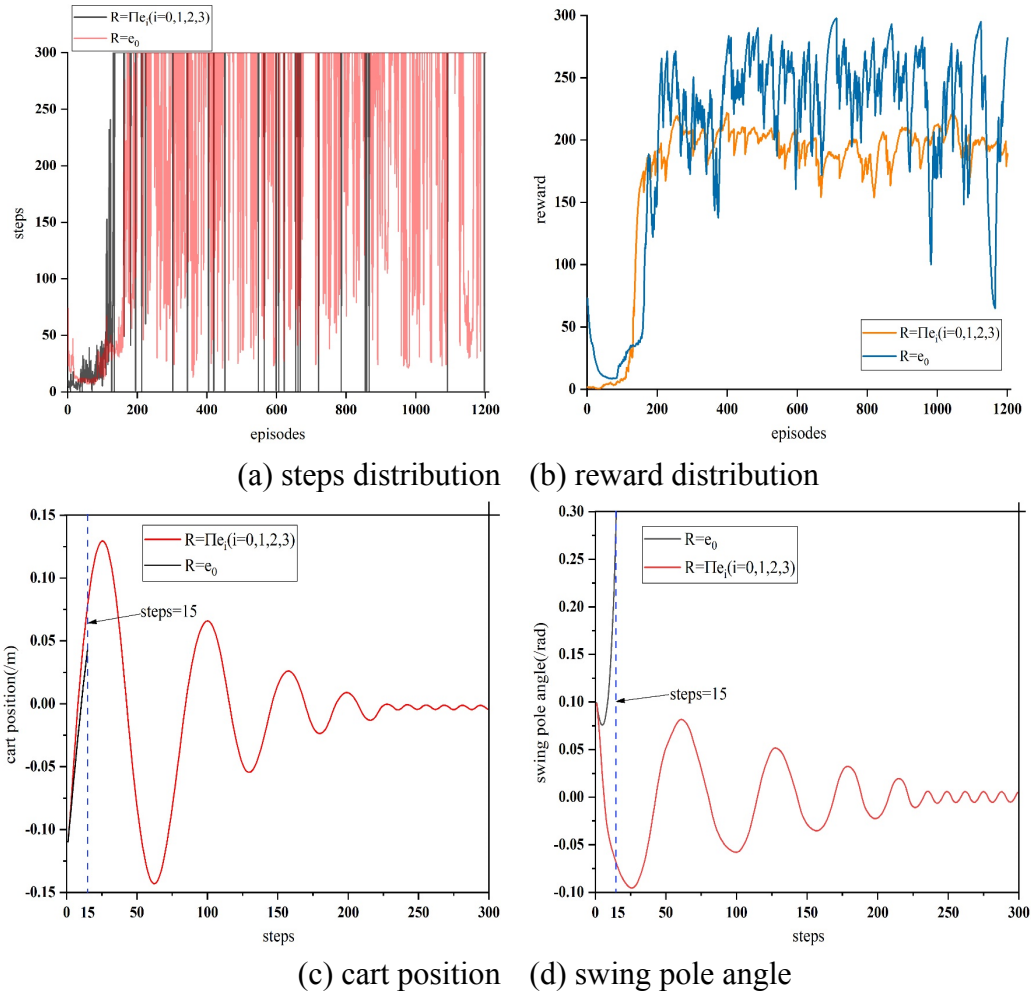
Table 7. Neural network parameters.

Network	Network structure	Learning rate	Loss function	Activation function	Batch size
Actor network	(4,128) (128,128) (128,1)	1×10^{-3}	$-Q(s,a\|w^Q)$	Adam	64
Critic network	(5,128) (128,128) (128,1)	1×10^{-2}	MSE	Adam	64

It can be seen from Figure 6(a),(b) that in the case of using single-reward function, when the episodes are less than 180, the number of times the rule is met is less than 100, and the reward value under each episode is less than 50. As the episodes gradually increase, the distribution of step and reward curves fluctuates greatly. In contrast, using detail-reward function, when the episodes are greater than 180, the distribution of the reward curve is relatively flat, and the reward distribution under each episode is around 210.

In Figures 6(c),(d), when steps = 15 and the single-reward function is used, the position of the cart is located at 0.05 in the positive direction, and the swing pole angle is 0.3, which means that the agent control fails. After 200 steps, by using detail-reward function, both the cart position and the swing pole angle are close to zero, while the distribution of the position and angle curves is relatively gentle.

From the above analysis it can be concluded that in the process of using the DDPG algorithm to train an agent, the reward obtained by using the single-reward function is greater than that obtained by using the detail-reward function in most cases. However, the distribution of the reward curve under the single-reward function fluctuates greatly and the model stability is poor. In addition, the model trained with single-reward function will lead to its out of control in the process of testing. The experimental effect of using the detail-reward function is better than that of using the single-reward function.



(a) steps distribution (b) reward distribution

(c) cart position (d) swing pole angle

Figure 6. The training and testing results of DDPG.

2) Comparison of experimental effects of different algorithms under the same detail-reward function

Through the analysis and comparison of the above experimental results, the following conclusions can be drawn: i) The method of using DRM in the inverted pendulum system under deep reinforcement learning is effective and innovative. ii) A good design of evaluation functions is crucial for the training and testing process. Under different algorithms, the detail-reward function produces better experimental results than a single-reward function.

Figures 7 and 8 show the experimental comparison of the models generated by different deep reinforcement learning algorithms during training under the same detail-reward function in the testing phase. Figure 7 shows the distribution curves of cart positions in each step of the inverted pendulum system when DRM is applied in the experiment. After 50 steps, the model generated by DQN algorithm converges to a point close to 0 in the test phase, while the model generated by policy-gradient algorithm fluctuates in the range of [−0.06, 0.05], and the model generated by DDPG algorithm shows that the moving position of the cart gradually converges to the vicinity of zero with the increase of the number of steps. The right part of Figure 7 is an enlarged schematic diagram of area B. It can be seen from the figure that the model generated by the policy-gradient algorithm fluctuates greatly, i.e., the moving position of the cart fluctuated wildly in the range of

more than 0.06. The data of area B also shows that the models generated by DQN and DDPG algorithms has a small fluctuation range, both mean that the cart fluctuates reasonably near the equilibrium point.

Figure 8 is the distribution curves of swing pole angle at each step under the models generated by three algorithms under the application of DRM in the experiment. After 50 steps, the model generated by DQN algorithm converges to near zero during the test process, the model generated by the policy-gradient algorithm fluctuates in the range of [−0.05, 0.05], and the model generated by the DDPG algorithm gradually converges to zero with the increase of steps. The right part of Figure 8 is the enlarged schematic diagram of area A. After 200 steps, the models generated by DDPG and DQN will guide the swing pole angle to vibrate around 0. The fluctuation range of the swing pole angle under the model generated by policy-gradient is [−0.06, 0.06], which is larger than that of the above two models.

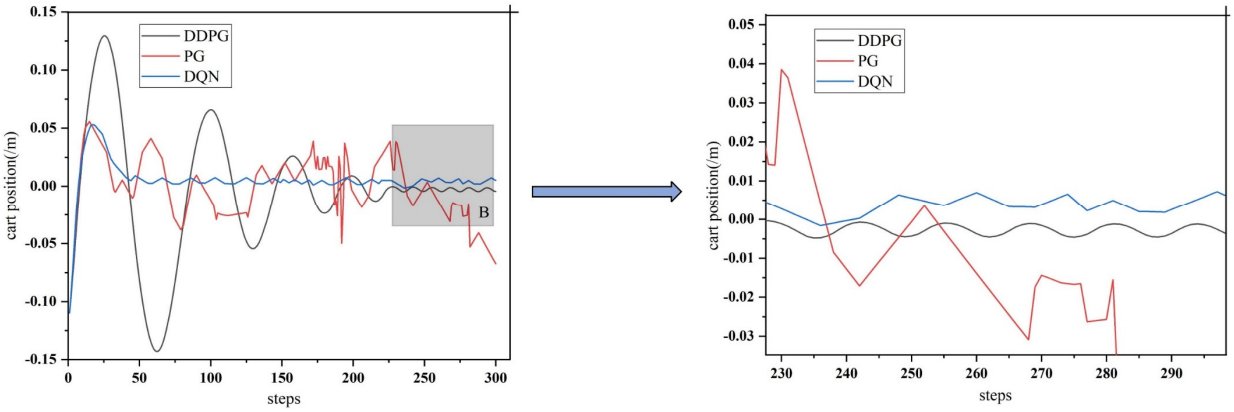


Figure 7. Comparison of test accuracy under different algorithms (cart position).

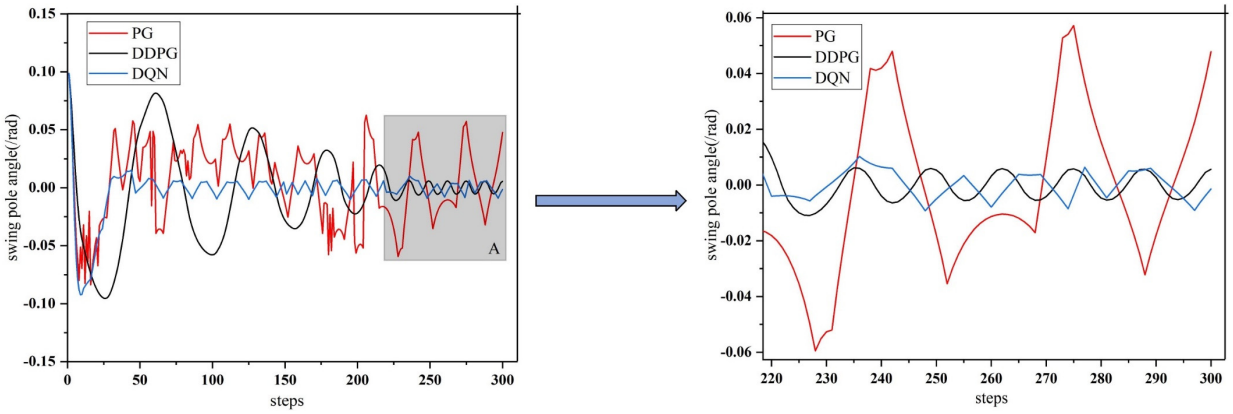


Figure 8. Comparison of test accuracy under different algorithms (swing pole angle).

By analyzing the distribution curves in Figures 7 and 8, the following conclusions can be drawn: i) The system stability of the policy-gradient algorithm is weaker than that of DQN and DDPG during the test process, and the car position and the swing pole angle are difficult to converged to zero. ii) During the test process of the DQN algorithm, the cart position and the swing pole angle distribution are both close to the zero. However, the output of the DQN algorithm is a discrete action.

In a continuous control system, the frequent output of discrete actions will affect the stability of the system, and on a physical platform, the large-scale pulsed current output can greatly reduce the lifetime of electronic devices. iii) DDPG algorithm is continuous action output. Compared to the DQN algorithm, the convergence rate is slower. During the test, as the number of steps increased, the cart position and the swing pole angle distribution gradually converged to zero. In an inverted pendulum system, the output of the DDPG model can ensure the stability of the system. Compared with the above three deep reinforcement learning algorithms, as a continuous action output algorithm, DDPG can be applied to the inverted pendulum control system and achieve good experimental results by introducing DRM into the algorithm.

4.4.2. Insufficient research and future work

According to the characteristics of the inverted pendulum NCS, deep reinforcement learning with DRM can solve the optimal control problem of the system. Through the above experiments, the control accuracy of the inverted pendulum system under different reward functions is compared. At present, this paper has the following deficiencies in experimental details: 1) This paper just introduce DRM to the inverted pendulum system without using more complex nonlinear systems. 2) In this paper, a novel DRM method is given, and the evaluation functions and reward function are established by taking the inverted pendulum system as an example, but the unified rules for the establishment of evaluation function are not given, i.e., it depends on the designer's personalized design. 3) The parameters of the inverted pendulum system used have limitations. For example, we only set the angle of the swing pole in the range of [−0.174, 0.174], and did not study the control problem of a broader state space. 4) When designing the dynamic environment of the inverted pendulum system in this paper, the unstable disturbance of the real inverted pendulum experimental platform is not considered, which has certain limitations for deploying agent models to real physical platforms.

Future work will improve in the following aspects: 1) We will use the method of DRM in more nonlinear systems and summarize the applicability of the method in a wider experimental setting. 2) The construction of the evaluation function is a reward realization for a specific observation perspective. In the course of experiments in more nonlinear environments, the validity of the evaluation function and the design method will be summarized. 3) We will increase the control range of the swing pole angle in the inverted pendulum system, and use the deep reinforcement learning algorithms to solve the global optimal control problems. 4) We will design the target policy and action policy separately according to the importance sampling off policy learning theory in deep reinforcement learning. The action strategy is a strategy optimized for training in a dynamic environment, and the target strategy is a real experimental platform. The control strategy uses the action strategy to sample the data in the dynamic environment, which is used to predict the target strategy under the real experimental platform, so as to realize the optimal control of the real system.

5. Conclusions

In this paper, a reward mechanism based on control details is proposed for the first time. According to the specific requirements of control objectives, the state space is divided into different observation perspectives, and then the evaluation function is designed. Finally, the final reward

function is composed in the form of continuous multiplication of the evaluation functions, which replaces the utility function in HJB equation. Based on Lyapunov stability theory and Q-learning, the stability of the control law is proved. Then taking the nonlinear inverted pendulum system as the experiment object, the detail-reward mechanism is introduced into the deep reinforcement learning algorithms. After designing the reward function according to the detail-reward mechanism, the effects of different reward functions on the experimental accuracy are compared under three algorithms: DQN, policy-gradient and DDPG. The effectiveness of this method is proved by experiments, and the optimal reward function and deep reinforcement learning algorithm model of inverted pendulum system are determined. We will do the following work in the future: 1) Apply the detail reward mechanism to other deep reinforcement learning algorithms for verification, such as A3C, TD3 and PPO. 2) Apply the detail reward mechanism to more nonlinear control systems, such as multi-agent, robot motion control systems, etc. 3) Optimize the model generated by the deep reinforcement learning algorithm to improve the control effect.

Acknowledgments

The author acknowledge the support from the Joint Development Research Institute of Intelligent Motion Control Technology of the Liaoning Provincial Department of Education and the National Key R&D Program of China (Grant No. 2017YFB1300700).

Conflict of interest

The authors declared that they have no conflicts of interest in this work.

References

1. J. Wu, W. Sun, S. F. Su, Y. Q. Wu, Adaptive quantized control for uncertain nonlinear systems with unknown control directions, *Int. J. Robust Nonlinear Control*, **31** (2021), 8658–8671. https://doi.org/10.1002/rnc.5748
2. A. Shatyrko, J. Diblík, D. Khusainov, M. Růžičková, Stabilization of Lur'e-type nonlinear control systems by Lyapunov-Krasovskii functionals, *Adv. Diff. Equations*, **2012** (2012), 1–9. https://doi.org/10.1186/1687-1847-2012-229
3. K. Tatsuya, Limit-cycle-like control for 2-dimensional discrete-time nonlinear control systems and its application to the Hénon map, *Commun. Nonlinear Sci. Numer. Simul.*, **18** (2013), 171–183. https://doi.org/10.1016/j.cnsns.2012.06.012
4. Y. H. Wei, Lyapunov stability theory for nonlinear nabla fractional order systems, *IEEE Trans. Circuits Sys.,* **68** (2021), 3246–3250. https://doi.org/10.1109/TCSII.2021.3063914
5. G. Pole, A. Girard, P. Tabuada, Approximately bisimilar symbolic models for nonlinear control systems, *Automatica*, **44** (2008), 2508–2516. https://doi.org/10.1016/j.automatica.2008.02.021
6. H. G. Zhang, X. Zhang, Y. H. Luo, J. Yang, An overview of research on adaptive dynamic programming, *Acta Autom. Sin.*, **39** (2013), 303–311. https://doi.org/10.1016/S1874-1029(13)60031-2

7. M. Volckaert, M. Diehl, J. Swevers, Generalization of norm optimal ILC for nonlinear systems with constraints, *Mech. Syst. Signal Proc.*, **39** (2013), 280–296. https://doi.org/10.1016/j.ymssp.2013.03.009
8. W. N. Gao, Z. P. Jiang, Nonlinear and adaptive suboptimal control of connected vehicles: A global adaptive dynamic programming approach, *J. Intell. Rob. Syst.*, **85** (2017), 597–611. http://doi.org/10.1007/s10846-016-0395-3
9. E. Trélat, Optimal control and applications to aerospace: Some results and challenges, *J. Optim. Theory Appl.,* **154** (2012), 713–758. https://doi.org/10.1007/s10957-012-0050-5
10. M. Margaliot, Stability analysis of switched systems using variational principles: An introduction, *Automatica*, **42** (2006), 2059–2077. https://doi.org/10.1016/j.automatica.2006.06.020
11. A. Maidi, J. P. Corriou, Open-loop optimal controller design using variational iteration method, *Appl. Math. Comput.*, **219** (2013), 8632–8645. https://doi.org/10.1016/j.amc.2013.02.075
12. F. H. Clarke, R. B. Vinter, The relationship between the maximum principle and dynamic programming, *SIAM J. Control Optim.,* **25** (1987), 1291–1311. http://doi.org/10.1137/0325071
13. R. W. Beard, G. N. Saridis, J. T. Wen, Approximate solutions to the time-invariant Hamilton–Jacobi–Bellman equation, *J. Optim. Theory Appl.*, **96** (1998), 589–626. http://doi.org/10.1023/A:1022664528457
14. J. A. Roubos, S. Mollov, R. Babuška, H. B. Verbruggen, Fuzzy model-based predictive control using Takagi–Sugeno models, *Int. J. Approximate Reasoning*, **22** (1999), 3–30. http://doi.org/10.1016/S0888-613X(99)00020-1
15. D. A. Bristow, M. Tharayil, A. G. Alleyne, A survey of iterative learning control, *IEEE Control Syst. Mag.*, **26** (2006), 96–114. https://doi.org/10.1109/MCS.2006.1636313
16. P. J. Werbos, W. T. Miller, R. S. Sutton, A menu of designs for reinforcement learning over time, *Neural networks for control*, MIT press, Cambridge, (1990), 67–95.
17. J. Wang, R. Y. K. Fung, Adaptive dynamic programming algorithms for sequential appointment scheduling with patient preferences, *Artif. Intell. Med.*, **63** (2015), 33–40. https://doi.org/10.1016/j.artmed.2014.12.002
18. D. V. Prokhorov, D. C. Wunsch, Adaptive critic designs, *IEEE Trans. Neural Networks*, **8** (1997), 997–1007. http://doi.org/10.1109/72.623201
19. J. J. Murray, C. J. Cox, G. G. Lendaris, R. Saeks, Adaptive dynamic programming. *IEEE Trans. Syst. Man Cybern.*, **32** (2002), 140–153. http://doi.org/10.1109/TSMCC.2002.801727
20. H. G. Zhang, Q. L. Wei, D. R. Liu, An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games, *Automatica*, **47** (2011), 207–214. http://doi.org/10.1016/j.automatica.2010.10.033
21. Q. L. Wei, H. G. Zhang, D. R. Liu, Y. Zhao, An optimal control scheme for a class of discrete-time nonlinear systems with time delays using adaptive dynamic programming, *Acta Autom. Sin.*, **36** (2010), 121–129. http://doi.org/10.1016/S1874-1029(09)60008-2
22. J. Ding, S. N. Balakrishnan, Approximate dynamic programming solutions with a single network adaptive critic for a class of nonlinear systems, *J. Control Theory Appl.*, **9** (2011), 370–380. http://doi.org/10.1007/s11768-011-0191-3

23. D. R. Liu, D. Wang, D. B. Zhao, Adaptive dynamic programming for optimal control of unknown nonlinear discrete-time systems, in *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL) IEEE*, (2011), 242–249. https://doi.org/10.1109/ADPRL.2011.5967357
24. J. Modayil, A. White A, R. S. Sutton, Multi-timescale nexting in a reinforcement learning robot, *Adapt. Behav.*, **22** (2014), 146–160. http://doi.org/10.1177/1059712313511648
25. C. X. Mu, Y. Zhang, Z. K. Gao, C. Y. Sun, ADP-based robust tracking control for a class of nonlinear systems with unmatched uncertainties, *IEEE Trans. Syst. Man Cybern. Syst.,* **50** (2019), 4056–4067. http://doi.org/10.1109/TSMC.2019.2895692
26. H. Y. Dong, X. W. Zhao, B. Luo, Optimal tracking control for uncertain nonlinear systems with prescribed performance via critic-only ADP, *IEEE Trans. Syst. Man Cybern. Syst.*, **52** (2020), 561–573. https://doi.org/10.1109/TSMC.2020.3003797
27. R. Z. Song, L. Zhu, Optimal fixed-point tracking control for discrete-time nonlinear systems via ADP, *IEEE/CAA J. Autom. Sin.*, **6** (2019), 657–666. https://doi.org/10.1109/JAS.2019.1911453
28. M. M. Liang, Q. L. Wei, A partial policy iteration ADP algorithm for nonlinear neuro-optimal control with discounted total reward, *Neurocomputing*, **424** (2021), 23–34. https://doi.org/10.1016/j.neucom.2020.11.014
29. B. Fan, Q. M. Yang, X. Y. Tang, Y. X. Sun, Robust ADP design for continuous-time nonlinear systems with output constraints, *IEEE Trans. Neural Networks Learn. Syst.*, **29** (2018), 2127–2138. https://doi.org/10.1109/TNNLS.2018.2806347
30. X. Yang, H. B. He, Self-learning robust optimal control for continuous-time nonlinear systems with mismatched disturbances, *Neural Netw*orks, **99** (2018), 19–30. https://doi.org/10.1016/j.neunet.2017.11.022
31. D. R. Liu, X. Yang, D. Wang, Q. L. Wei, Reinforcement-learning-based robust controller design for continuous-time uncertain nonlinear systems subject to input constraints, *IEEE Trans. Cybern.*, **45** (2015), 1372–1385. http://doi.org/10.1109/TCYB.2015.2417170
32. X. Yang, D. R. Liu, D. Wang, Reinforcement learning for adaptive optimal control of unknown continuous-time nonlinear systems with input constraints, *Int. J. Control*, **87** (2014), 553–566. https://doi.org/10.1080/00207179.2013.848292
33. J. G. Zhao, M. G. Gan, Finite-horizon optimal control for continuous-time uncertain nonlinear systems using reinforcement learning, *Int. J. Syst. Sci.*, **51** (2020), 2429–2440. https://doi.org/10.1080/00207721.2020.1797223
34. B. Zhao, D. R. Liu, C. M. Luo, Reinforcement learning-based optimal stabilization for unknown nonlinear systems subject to inputs with uncertain constraints, *IEEE Trans. Neural Networks Learn. Syst.*, **31** (2019), 4330–4340. https://doi.org/10.1109/TNNLS.2019.2954983
35. D. Wang, J. F. Qiao, Approximate neural optimal control with reinforcement learning for a torsional pendulum device, *Neural Networks,* **117** (2019), 1–7. https://doi.org/10.1016/j.neunet.2019.04.026
36. J. W. Kim, B. J. Park, H. Yoo, T. H. Oh, J. H. Lee, J. M. Lee, A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system, *J. Proc. Control*, **87** (2020), 166–178. https://doi.org/10.1016/j.jprocont.2020.02.003
37. F. Y. Wang, N. Jin, D. R. Liu, Q. L. Wei, Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with epsilon-error bound, *IEEE Trans. Neural Networks*, **22** (2010), 24–36. https://doi.org/10.1109/TNN.2010.2076370

38. K. G. Vamvoudakis, F. L. Lewis, Multi-player non-zero-sum games: Online adaptive learning solution of coupled Hamilton–Jacobi equations, *Automatica*, **47** (2011), 1556–1569. https://doi.org/10.1016/j.automatica.2011.03.005
39. Q. L. Wei, D. R. Liu, An iterative epsilon-optimal control scheme for a class of discrete-time nonlinear systems with unfixed initial state, *Neural Networks*, **32** (2012), 236–244. https://doi.org/10.1007/978-981-10-4080-1_2
40. D. R. Liu, Q. L. Wei, P. F. Yan, Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems, *IEEE Trans. Syst. Man Cybern. Syst.*, **45** (2015), 1577–1591. https://doi.org/10.1109/TSMC.2015.2417510
41. S. H. Li, H. B. Du, X. H. Yu, Discrete-time terminal sliding mode control systems based on euler's discretization, *IEEE Trans. Autom. Control*, **59** (2013), 546–552. https://doi.org/10.1109/TAC.2013.2273267
42. D. Bertsekas, Dynamic Programming and Optimal Control: Volume I, Athena scientific, 2012.
43. C. J. C. H. Watkins, P. Dayan, Q-learning, *Mach. Learn.*, **8** (1992), 279–292. https://doi.org/10.1007/BF00992698
44. A. Y. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, *LCML*, **99** (1999), 278–287.
45. L. Buşoniu, B. D. Schutter, R. Babuška, Approximate dynamic programming and reinforcement learning, in *Interactive collaborative information systems,* (2010), 3–44. https://doi.org/10.1007/978-3-642-11688-9_1
46. T. Aotani, T. Kobayashi, K. Sugimoto, Bottom-up multi-agent reinforcement learning by reward shaping for cooperative-competitive tasks, *Appl. Intell.*, **51** (2021), 4434–4452. https://doi.org/10.1007/s10489-020-02034-2
47. C. HolmesParker, A. K. Agogino, K. Tumer, Combining reward shaping and hierarchies for scaling to large multiagent systems, *Knowl. Eng. Rev.*, **31** (2016), 3–18. https://doi.org/10.1017/S0269888915000156
48. P. Mannion, S. Devlin, K. Mason, J. Duggan, E. Howley, Policy invariance under reward transformations for multi-objective reinforcement learning, *Neurocomputing*, **263** (2017), 60–73. https://doi.org/10.1016/j.neucom.2017.05.090
49. P. Mannion, S. Devlin, J. Duggan, E. Howley, Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning, *Knowl. Eng. Rev.,* **33** (2018). https://doi.org/10.1017/S0269888918000292
50. C. Y. Hu, A confrontation decision-making method with deep reinforcement learning and knowledge transfer for multi-agent system, *Symmetry*, **12** (2020), 631. https://doi.org/10.3390/sym12040631
51. A. G. Barto, R. S. Sutton, C. W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, *IEEE Trans. Syst. Man Cybern. Syst.*, **5** (1983), 834–846. https://doi.org/10.1109/TSMC.1983.6313077
52. L. B. Prasad, B. Tyagi, H. O. Gupta, Optimal control of nonlinear inverted pendulum system using PID controller and LQR: Performance analysis without and with disturbance input, *Int. J. Autom. Comput.*, **11** (2014), 661–670. https://doi.org/10.1007/s11633-014-0818-1
53. V. Mnih, K. Kavukcuoglu, D. Silver, J. Veness, A. Graves, M. Riedmiller, et al, Human-level control through deep reinforcement learning, *Nature,* **518** (2015), 529–533. https://doi.org/10.1038/nature14236

54. T. d. Bruin, J. Kober, K. Tuyls, R. Babuˇska, Experience selection in deep reinforcement learning for control, *J. Mach. Learn. Res.,* **19** (2018).
55. B. C. Stadie, S. Levine, p. Abbeel, Incentivizing exploration in reinforcement learning with deep predictive models, preprint, arXiv: 1507.00814.
56. Z. L. Ning, P. R. Dong, X. J. Wang, JJPC. Rodrigues, F. Xia, Deep reinforcement learning for vehicular edge computing: An intelligent offloading system, in *ACM Transactions on Intelligent Systems and Technology* , **10** (2019), 1–24. https://doi.org/10.1145/3317572
57. H. Yoo, B. Kim, J. W. Kim, J. H. Lee, Reinforcement learning based optimal control of batch processes using Monte-Carlo deep deterministic policy gradient with phase segmentation, *Comput. Chem. Eng.*, **144** (2021), 107133. https://doi.org/10.1016/j.compchemeng.2020.107133
58. T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, et al, Continuous control with deep reinforcement learning, preprint, arXiv: 1509.02971.
59. S. Satheeshbabu, N. K. Uppalapati, T. Fu, G. Krishnan, Continuous control of a soft continuum arm using deep reinforcement learning, in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft), IEEE*, (2020), 497–503. https://doi.org/10.1109/RoboSoft48309.2020.9116003
60. Y. Ma, W. B. Zhu, M. G. Benton, J. Romagnoli, Continuous control of a polymerization system with deep reinforcement learning, *J. Proc. Control*, **75** (2019), 40–47. https://doi.org/10.1016/j.jprocont.2018.11.004
61. R. B. Zmood, The euclidean space controllability of control systems with delay, *SIAM J. Control*, **12** (1974), 609–623. https://doi.org/10.1137/0312045

AIMS Press