Applied Intelligence (2021) 51:3405–3420
https://doi.org/10.1007/s10489-020-01906-x

An adaptive adjustment strategy for bolt posture errors based on an improved reinforcement learning algorithm

Wentao Luo[1] · Jianfu Zhang[1,2] · Pingfa Feng[1,2,3] · Haochen Liu[4] · Dingwen Yu[1] · Zhijun Wu[1]

Published online: 13 November 2020

Abstract
Designing an intelligent and autonomous system remains a great challenge in the assembly field. Most reinforcement learning (RL) methods are applied to experiments with relatively small state spaces. However, the complicated situation and high-dimensional spaces of the assembly environment cause traditional RL methods to behave poorly in terms of their efficiency and accuracy. In this paper, a model-driven adaptive proximal proximity optimization (MAPPO) method was presented to make the assembly system autonomously rectify the bolt posture error. In the MAPPO method, a probabilistic tree and adaptive reward mechanism were used to improve the calculation efficiency and accuracy of the traditional PPO method. The size of the action space was reduced by establishing a hierarchical logical relationship for each parameter with a probabilistic tree. Based on an adaptive reward mechanism, the phenomenon that the algorithm easily falls into local minima could be improved. Finally, the proposed method was verified based on the Unity simulation engine. The advancement and robustness of the proposed model were also validated by comparing different cases in simulations and experiments. The results revealed that MAPPO has better algorithm efficiency and accuracy compared with other state-of-the-art algorithms.

Keywords Model-driven method · Intelligent assembly · Probabilistic tree · Adaptive reward mechanism · Reinforcement learning · Physical simulation engine

1 Introduction

With the advancements in machine learning theory, complex tasks can be autonomously executed in many essential fields, such as the automated detection of heart diseases or organ cancer in medical clinics [1–3], the interpersonal influence analysis in sociology [4] and quality inspection in engineering [5]. Recently, researchers have proposed machine learning theory in the continuous control task of machines to improve the intelligent ability of machines based on accumulated learning experiences, which makes the era of human intelligence possible, especially in industry applications [6]. Gullapalli et al. [7] considered the impact of environmental uncertainty on robot control and established a nonlinear neural network structure to train a robot so that the robot could choose different control modes according to environmental changes. Yang et al. [8] proposed a progressive learning method that aimed to stimulate robot learning impedance control rules to effectively suppress control instability under high-speed assembly conditions. Nuttin et al. [9] presented a learning controller that was capable of increasing the insertion speed during consecutive peg-into-hole operations, without increasing the contact force level. However, the above control algorithms require a long training time and ample expert knowledge to learn a new complicated assembly task. Therefore, an advanced efficient learning algorithm without requiring abundant expert knowledge needs to be applied in the assembly process.

Reinforcement learning (RL) [10], one of the machine learning methods, offers an effective way for the agent to select and learn from a sequence of actions that maximizes the accumulated reward over time without expert knowledge. As an edging technique in continuous action decision-making,

✉ Jianfu Zhang
zhjf@tsinghua.edu.cn

1 Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China

2 State Key Laboratory of Tribology, and Beijing Key Lab of Precision/Ultra-precision Manufacturing Equipment and Control, Tsinghua University, Beijing 100084, China

3 Division of Advanced Manufacturing, Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China

4 School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China

RL has been successfully applied in versatile domains to provide the learning ability of intelligent systems [11, 12]. In automatic or intelligent assembly, robots are often utilized to put a bolt into a bolt hole [13, 14]. However, due to the uncertainty related to factors such as gaskets and materials, bolt posture adjustment is a difficult and unsolved task in assembly, which often results in lower assembly quality. Therefore, the intelligent adjustment ability of assembly systems has become a popular research topic. Xu et al. [15] proposed a continuous-control-based deep deterministic policy gradient (DDPG) algorithm to accomplish the assembly task through the learned policy. Neural net-based RL was proposed in [16] to assure the high precision of the motion in an assembly of small components, which is applicable to a wide class of mechanical systems. A probabilistic-inference-based RL algorithm (PILCO) was proposed to overcome the time-consuming challenge in training the agent to learn from the real environment, and it achieves an unprecedented speed of learning compared with state-of-the art RL [17]. The constant improvement of the RL algorithm and more efforts to combine the RL algorithm with complicated assembly problems in real factories lay a solid foundation for the boost of global intelligent assembly.

Proximal policy optimization (PPO) [18], a new family of RL methods proposed in 2017, is much simpler to implement, more general, and has better performance on a collection of benchmark tasks than state-of-the-art RL methods such as A2C [19] and ACER [20]. Considering the few applications of PPO on assembly problems, executing PPO in real applications may achieve better performance than previous RL methods. However, due to the complex situation and high-dimensional space of the real assembly process, a PPO method with a random exploration learning policy could result in two disadvantages:

1. Long time cost: In complicated assembly tasks, PPO requires many robot executions to obtain high control precision [21], but it is costly to explore all the possible action scenarios in a continuous and large search space. Moreover, high dimensional space exponentially increases the difficulty of space exploration. Building a hierarchical structure to show interrelationships among each action and state of space is an effective way to simplify the complex space, but few relevant studies have been done in the RL domain.
2. Low algorithm accuracy: The reward function of PPO methods guides the agent in exploring and exploiting the large search space [22]. However, it may be difficult for the agents to converge or they may easily fall into local optimal values during the process of action exploration if the reward function is not designed well. Some reward forms, such as sparse rewards, hierarchical rewards and linear rewards, are commonly applied in the RL method [23–25], but these performances are unstable in different applications. Therefore, it is challenging to select the best function reward for a specific RL method.

To address the problems mentioned above, a model-driven adapted proximal policy optimization (MAPPO) algorithm is proposed in this paper. First, a probabilistic tree model was established to reduce the dimensions of the exploration space and accelerate the speed of the training process. Second, a new adaptive reward function was designed to reduce the probability of the proposed algorithm being trapped in local suboptimal values. Finally, a physical engine-based assembly simulation platform was established to verify the effectiveness of the algorithm, and sufficient comparison experiments were conducted to indicate the advancement and robustness of our model. Specifically, the good performance of MAPPO with the probabilistic tree method and the new adaptive reward function were validated to compare them with the traditional PPO method in assembly adjustment tasks. Moreover, a new simulation situation was built, and more state-of-art RL methods, such as DDPG [26], SAC [27] and TRPO [28], were cited and applied to prove the leading role in RL methods and generalization ability of our model in versatile situations.

The novelty and contribution of this paper is as follows:

Algorithm novelty: A model-driven adaptive PPO method is proposed and proven to have better performance than the traditional PPO method and other advanced RL methods when executing dexterous assembly tasks such as bolt pose adjustment.

Technical contribution: The model we propose provides the assembly machine with the ability to autonomously make decisions, which is more useful than manually preprograming a robot, as our model can learn tasks from uncertain and complicated situations in reality. Moreover, the unsolved and high-level engineering assembly problem is detailed and modelled in a mathematical way, and the intelligent model is put into practice and solves the challenging problem in the assembly field, proving the great contribution of our model in the intelligent assembly domain. Therefore, an intelligent assembly paradigm is provided in this paper.

2 Related works

This section explains the theoretical background that is the basis for understanding the remainder of this paper.

2.1 Reinforcement learning

RL can be formulated as a Markov decision process (MDP), which describes a sequential action decision problem requiring the agent to select the sequence of actions that maximizes the accumulated reward over time. An MDP can be defined as quadruples $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where,

- $\mathcal{S} = (S_1, S_2, \cdots S_n)$ is an n-sized state space, and each state $\{S_i\}_{1 \leq i \leq n}$ represents one position in the dynamic environment. In most practical problems, the state space size is infinite.
- $\mathcal{A} = (A_1, A_2, \cdots, A_m)$ is an m-sized action space and each action $\{a_j\}_{1 \leq j \leq m}$ represents one action the agent executes.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a transition matrix and $\mathcal{T}\left(S, A, S^{'}\right)$ represents the transition probability when taking action A in state S to reach state $S^{'}$. The transition probability can also be expressed as $\mathcal{T}\left(S, A, S^{'}\right) = P\left(S^{'} | S, A\right)$. The following equation is introduced in the MDP: $\mathrm{p}(s^1 = s_{t+1} | s = s_t, a = a_t) = \mathrm{p}(s_{t+1} | s_1 s_2, \cdots, s_t)$
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is a reward function and $R(s, a)$ represents for the reward value the agent acquires when executing an action $A \in \mathcal{A}$ in state $S \in \mathcal{S}$.

A series of state-action pairs consists of the agent policy, denoted by π. The agent policy represents a series of continuous actions taken by an agent to achieve its goal. Policy π can be denoted as $\Pi = \{(S_1, A_1), (S_2, A_2), \cdots, (S_n, A_m)\}$.

Given policy π, the accumulated reward expectation upon executing action A in state S is called the action-state function, which can be defined as:

$$Q_\pi(S, A) = \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) | S_t = S, A_t = A\right)$$

where $Q_\pi(S, A)$ is an action-state value function, $\mathbb{E}$ is an expectation operator, (S_t, A_t) is a random state-action pair produced by policy π, and $\gamma \in [0, 1]$ is a discount factor, indicating the effect of future rewards on current agent behavior. $\gamma = 1$ means that the reward value of the future state has a great influence on the action-state function, while $\gamma = 0$ means that the reward value of the future state has little influence on the action-state function.

The goal of an agent is to find the optimal sequence of state-value pairs π^* to obtain the maximum action-state function $Q_\pi(S, A)$, and this process can be defined in formula (1).

$$\pi^* = argmax_\pi \sum_{s \in S} \sum_{A \in A} Q_\pi(S, A) \tag{1}$$

2.2 Proximal policy optimization algorithm

To obtain the optimal policy, PPO adopts the policy gradient method to explore the solution. Figure 3 shows an execution sequence π of the agent system. As the actions taken by the agent system in different states may not be the same, a statistical method is used to describe the agent's actions. The probability of the execution sequence π can be represented by the probability function, as shown in formula (2) (Fig. 1).

$$\begin{aligned} P_\theta(\pi) &= P(S_1) P_\theta(A_1|S_1) P(S_2|S_1, A_1) P_\theta(A_2|S_2) P(S_3|S_2, A_2) \cdots \\ &= P(S_1) \prod_{t=1}^{T} P_\theta(A_t|S_t) P(S_{t+1}|S_t, A_t) \end{aligned} \tag{2}$$

where $P_\theta(A_t| S_t)$ is the probability function upon applying action A_t in state S_t, and θ is the parameter of the function. The Gaussian function is often adopted as a probability function to perform the action selection task. $P(S_{r+1}| S_t, A_t)$ is the state transition probability upon taking action A_t in state S_t leading to state S_{t+1}.

The accumulated reward of an agent if it adopts policy π can be calculated in formula (3).

$$R(\pi) = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} (0 < \gamma < 1) \tag{3}$$

where $r_{t'}$ is the reward value upon taking an action at time $_{t'}$ γ is a discount factor. The expected reward value of all the policies can be expressed in formula (4).

$$\overline{R}_\theta = \sum_\pi R(\pi) P_\theta(\pi) = E_{\tau \sim P_\theta(\pi)}[R(\pi)] \tag{4}$$

Next, the gradient boost method is used to adjust the probability function parameters θ. The gradient solution process is shown in formula (5).

$$\begin{aligned} \nabla \overline{R}_\theta &= \sum_\pi R(\pi) \nabla P_\theta(\pi) = \sum_\pi R(\pi) P_\theta(\pi) \frac{\nabla P_\theta(\pi)}{P_\theta(\pi)} \\ &= \sum_\pi R(\pi) P_\theta(\pi) \nabla log P_\theta(\pi) \\ &= E_{\pi \sim P_\theta(\pi)}[R(\pi) \nabla log P_\theta(\pi)] \approx \frac{1}{N} \\ &\quad \times \sum_{n=1}^{N} R(\pi^n) \nabla log P_\theta(\pi^n) \end{aligned} \tag{5}$$

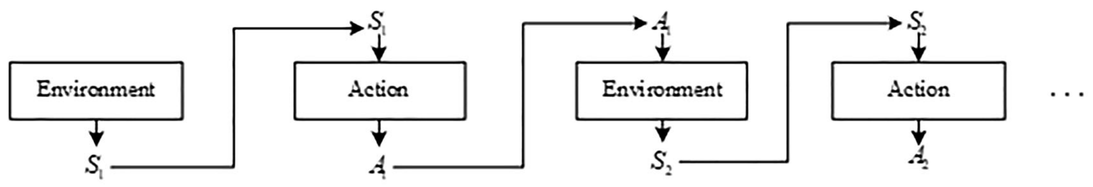
In the above calculation process, N policy samples are used to approximate the expectation value to solve the infinite-time problem approximately. Where π^n is the n-th policy of the policy samples, and ∇ is a gradient symbol.

Then importance sampling is used to increase the training speed and the expected reward value can be shown in formula (6).

$$\nabla \overline{R_\theta} = E_{\tau \sim P_{\theta'}(\tau)} \left[\frac{P_{\theta(\tau)}}{P_{\theta'(\tau)}} R(\tau) \nabla log P_\theta(\tau) \right] \tag{6}$$

where θ' is a predesigned parameter of the probability function based on experience.

According to formula (6), we can find that the gradient value is always positive, so a baseline is used to ensure that the gradient value can be negative (see formula (7)).



$$\pi=\{S_1,A_1,S_2,A_2,\cdots,S_T,A_T\}$$

Fig. 1 Markov process of an agent system

$$\nabla\overline{R_\theta}=E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{P_\theta(a_t|s_t)}{P_{\theta'}(a_t|s_t)}(R(\tau)-E(R(\tau)))\nabla logP_\theta(a_t|s_t)\right] \tag{7}$$

Using the following formula, we can convert the gradient value into a likelihood function (see formula (8)).

$$\nabla f(x)=f(x)\nabla logf(x) \tag{8}$$

Combining formulas (7) and (8), the objective function is calculated in formula (9).

$$J^{\theta'}=E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{P_\theta(a_t|s_t)}{P_{\theta'}(a_t|s_t)}A^{\theta'}(s_t,a_t)\right]$$
$$A^{\theta'}(s_t,a_t)=(R(\tau)-E(R(\tau)))\nabla logP_\theta(a_t|s_t) \tag{9}$$

To avoid the large difference between the two parameter distributions, the PPO model adds a clipping term to the objective function. The final objective function is shown in formula (10).

$$J^{\theta^k}_{pp0_2}(\theta)\approx\sum\nolimits_{(S_t,A_t)min}min\left(\frac{P_\theta(A_t|S_t)}{P_\theta k(A_t|S_t)}A^{\theta^k}(s_t,a_t),clip\left(\frac{P_\theta(A_t|S_t)}{P_{\theta^k}(A_t|S_t)},1-\varepsilon,1+\varepsilon\right)A^{\theta^k}(s_t,a_t)\right) \tag{10}$$

where $P_\theta(A_t|S_t)$ is the state transition probability of the target training environment, $P_{\theta^k}(A_t|S_t)$ is the state transition probability of the new sampling environment, γ is the loss coefficient, and R_t is the reward value of the time step t.

3 Bolt posture adjustment problem description

The quality of a bolt assembly has a great impact on the overall quality of the assembly, and the posture error of the bolt during tightening is one of the main factors that affects the quality of the bolt assembly. After the bolt is inserted into the bolt hole, there will be a tilt or deviation in the spatial range of the bolt posture due to interference from factors such as washers and the bolt hole clearance. As shown in Fig. 2(a), the bolts after loading are tilted due to the bolt hole clearance. The dotted line *o-o* in Fig. 2 is the central axis of the bolt hole, and line *b-b* is the central axis of the bolt shaft. The ideal bolt position is where the *o-o* line coincides with the *b-b* line. However, it can be seen from Fig. 2(b) that the solid line *b-b* is heterogeneous in spatial geometry with the *o-o* line, so at least two steps of translation and rotation are required to meet the requirements of a qualified assembly, which increases the auxiliary time for assembly. The state space of the bolt is shown in formula (11).

$$S(Bolt)=\{s_h,s_p\}=\{heterogeneous,parallel\} \tag{11}$$

where S represents the state space set of the bolts and s_h and s_p represent the heterogeneous and parallel relationship between the current bolt state and the target position state, respectively.

To improve the assembly efficiency, it is usually necessary to fine tune the position of the tightening shaft to improve the bolt's pose tilt in the stage of screwing the shaft into the nut. The principle of this fine-tuning process is as follows: when the tightening shaft sleeve is screwed into the bolt nut head, the bolt pose can be controlled by moving the tightening shaft. Specifically, if the bolt is tilted, moving the tightening shaft will provide a force on the bolt, and the connected workpiece will also give a reaction force at the bolt shaft. As the two forces are equal in magnitude, in opposite directions, and not on the same line, a rotational torque is generated on the bolt (as shown in Fig. 3(a)). If there is no tilt of the bolt, moving the tightening shaft will cause the bolt to move

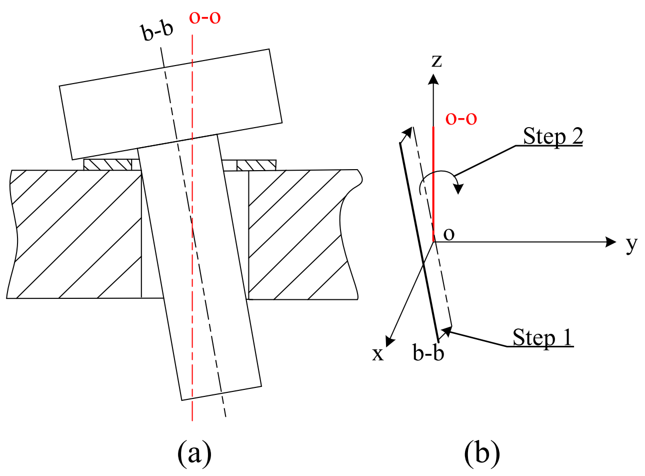


Fig. 2 Bolt pose geometric diagram. (**a**) Bolts with posture errors after loading. (**b**) Schematic diagram of bolt posture correction

horizontally, thereby achieving a fine translation adjustment (as shown in Fig. 3(b)).

4 MAPPO algorithm

The PPO algorithm is displayed in section 2.2. However, the PPO model has the following problems: 1. The amount of data explored by the agent in the continuous action space is too large, and the convergence speed is relatively slow; 2. The existing reward function setting mechanism easily causes the algorithm to fall into the local optimal solution and affects the accuracy of the model. Therefore, based on the PPO model, this paper proposes a probabilistic tree method and an adaptive reward mechanism method. Specifically, a probabilistic tree method is proposed to reduce the size of the action space by establishing a hierarchical relationship between the agent's action space and the state space based on experience. The probabilistic tree method solves the problem of the slow convergence speed of the current PPO algorithm. In addition, an adaptive reward mechanism method is established to provide agents with a heuristic guidance to explore the optimal strategy. This reduces the chance that the current PPO algorithm optimization process will easily fall into the local optimal solution, which affects the accuracy of the model.

4.1 Probabilistic tree method

The traditional reinforcement learning model assumes that the agent's action space and state space are independent of each other, that is, the actions generated by the agent are not affected by the current state. However, in a dynamic environment, there is a strong coupling relationship between the action space and state space. Specifically, an agent will generally explore all the states and action spaces to find an optimal policy, but some pre-constraints and pre-knowledge in engineering are ignored. For example, if the agent is staying in state *A,* it is better to choose action *B* or *C* in the next step based on pre-knowledge, but the RL model hardly understands this coupling relationship and explores all the possible actions, which makes the training process complicated. Therefore, this section proposes a probabilistic tree method to describe the strong interaction between the action space and state space based on experience and then establishes a mathematical relationship between the two types of spaces to reduce the agent's exploration space and improve the efficiency of the algorithm. The probabilistic tree method is mainly divided into two steps. The first step is to set up a hierarchical tree of the action space. The second step is to build a neural network of the relationship between the action space and state space.

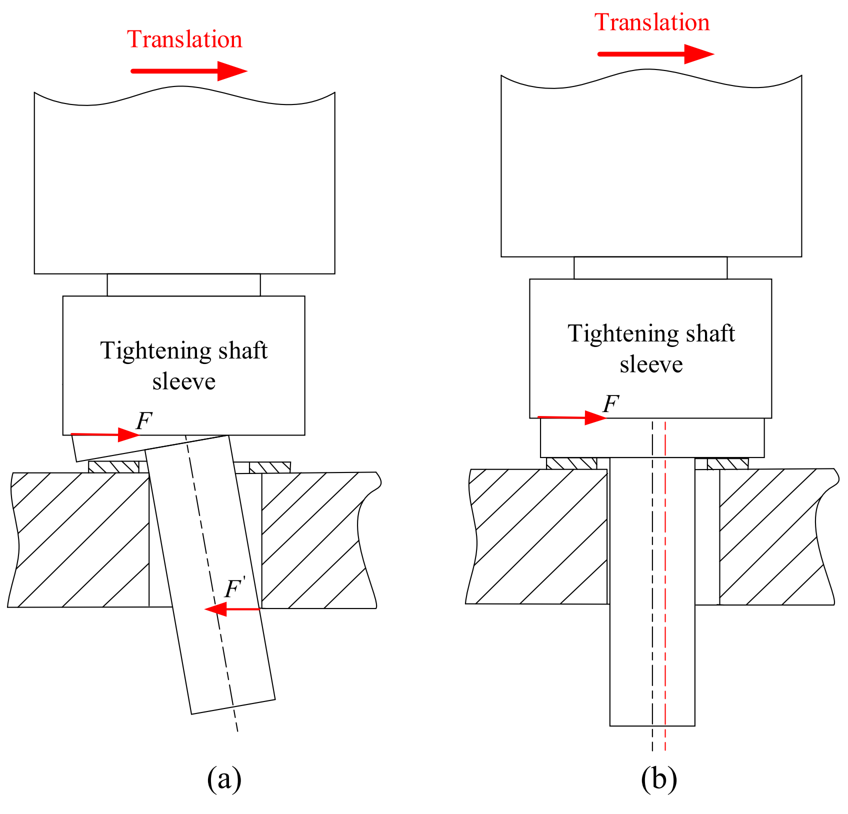


Fig. 3 Bolt posture adjustment method

4.1.1 Hierarchical tree of the action space

The action space of the agent system is expressed as. $A = \{A_1, A_2, A_3, \ldots, A_n\}$. An adjacency matrix **B** is created based on the interrelationship between the elements in set A.

$A_{tj} = 0$ means that the i-th action has no direct relationship with the j-th action, and $A_{tj} = 1$ means that the i-th action has a direct relationship with the j-th action.

$$B = \begin{bmatrix} A_{11} & A_{12} & L & A_{1n} \\ A_{21} & A_{22} & L & A_{2n} \\ M & M & O & M \\ A_{n1} & A_{n2} & L & A_{nn} \end{bmatrix}, \{A_{ij} | A_{ij} = 0, 1\}$$

The adjacent matrix **B** is exponentially calculated to obtain the reachable matrix **C**, where **I** is the identity matrix (see formula (12))

$$C = (B+1)^M, where (B+1)^M == (B+1) \tag{12}$$

The reachable set $R(S_t)$ is set up to represent the column number of the element whose i-th row in the reachable matrix **C** is equal to 1, and a predecessor set $Q(S_i)$ is established to represent the row number of the element whose value is equal to 1 in the i-th column of the reachable matrix **C**.

Traversing the entire action space, when i exists such that $R(S_i) \cap Q(S_i) = R(S_i)$, then A_t is used as the root node of the hierarchical tree. Next, the sequence number i is deleted from the sets R and Q to obtain a new set R^1, Q^1. In addition, traversing is continued in the action space, when the sequence number j exists such that $R^1(S_i) \cap Q^1(S_i) = R^1(S_i)$, then A_j is a child node of the existing node of the hierarchical tree. The above operation is repeated until all the action spaces are traversed, that is, the hierarchical tree is generated.

The hierarchical tree can sort out the logical relationship and hierarchical relationship between the elements of the action space, reduce the complexity of the action space, and lay the foundation for the establishment of the probabilistic tree.

4.1.2 Construction of the neural network

All the paths of the hierarchical tree from the root node to the leaf node are collected to form set F, which is the simplified action space. The number of paths in set F is the size of the new action space. The relationship between the state space S and the new action space F can be represented by building a multilayer fully connected neural network. As shown in Fig. 4.

The calculation relationship of the neural network at the l layer can be expressed in formula (13).

$$x_j^l = f\left(\sum_{t=1}^{d^{l-1}} x_t^{l-1} w_{if}^{l-1} + b^{l-1}\right) \tag{13}$$

where x_j^l represents the output value of the j-th neuron of the l-th layer of the neural network, d^l is the number of neurons of the l-th layer, w_{ij}^{l-1} is the weight vector between the l-th layer neural network and the $(l+1)$-th layer neural network, and b^{l-1} is the threshold vector. Each action space element value output by the neural network is the probability that the element value occurs.

In summary, the relationship between the state space and hierarchical action space can be simplified by a probabilistic tree, which can simplify the number of tuples in the action space. By referencing offline strategy learning, the agent system can learn experience from different environments and improve the learning efficiency. However, the agent easily converges to a local optimum, and this is inefficient for evaluating a strategy. To increase the performance of the agent, this paper starts by exploring the mechanism (that is, the design of the reward function) for improvement.

4.2 Adaptive reward mechanism

Designing a reward function is one of the difficulties of reinforcement learning algorithms. Inverse reinforcement learning can realize the design of the reward function by giving the machine an action paradigm and solving a certain reward function to make the paradigm optimal. However, this method requires the neural network to be re-established, which increases the computational complexity. At the same time, the neural network lacks the ability to be combined with actual physical problems, and the reward effect is low. Therefore, this paper proposes a reward function design method based on adaptive shifting theory, which achieves better reward results with less calculation cost.

For an agent in the process of reinforcement learning, ideally, the agent will actively explore the action space in a state with a large deviation from the target to reduce the chance of falling into the local minimum; however, when the agent moves closer to the target value, the action of obtaining model convergence should not be actively pursued. The specific mathematical model is described in the following section.

4.2.1 State parameter normalization

The state space of the system is $S = \{S_1, S_2, \ldots S_H\}$. As the parameter units of different physical meanings in state space S are different, it is impossible to establish a mathematical

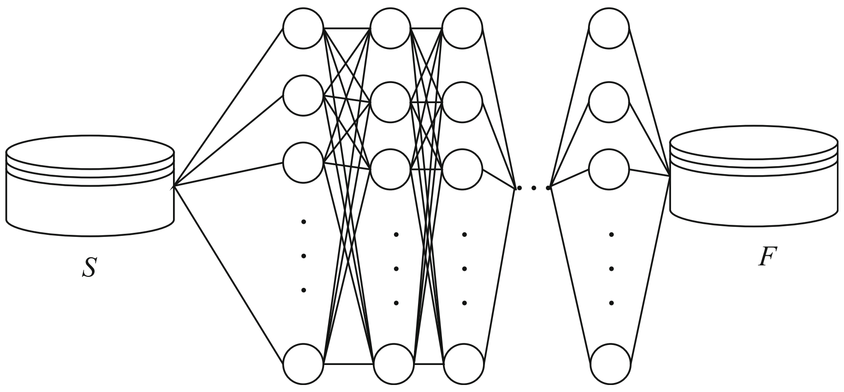


Fig. 4 Neural network between the state space and new action space

relationship, so it is necessary to normalize the elements in S. The normalization formula is shown in formula (14).

$$S_i^{norm} = \frac{S_t}{max(S_t)}, i = 1, 2, \ldots, H \tag{14}$$

where max(·) represents the upper limit of the value range of S_t. Let the system's target state be S^G, then the total loss under the current state of the system can be obtained by using the 2-norm method (see formula (15)).

$$Loss = \| \left(S^{norm} - S^G\right) \|_2 \tag{15}$$

4.2.2 Reward function based on the adaptive shifting theory

A trigonometric function is utilized to form the adaptive reward function in this section (see formula (16)).

$$R(Loss) = \cos \frac{\pi}{2Loss_{max}} Loss - 1, (0 \le Loss \le Loss_{max}) \tag{16}$$

It can be seen from the function relationship that when the loss value is large, the reward value function is a large negative number, which is called the penalty value, and the larger the total posture error is in this range, the slower the change of the reward value function is. As a result, the agent tries as many new actions as possible to escape the high penalty. In contrast, if the agent keeps stagnating, the cumulative penalty will be very large; when the loss value is small, the reward function is a small negative number. In this range, the smaller the loss value is, the slower the growth rate of the reward function. Therefore, when the agent approaches the target state, it tends to fine tune, reduces the choice of the large-scale space, and prefers to make local small-scale action selections.

After completing the design of the reward function, the parameters of the neural network can be backpropagated using the policy gradient theorem. The complete algorithm is shown in Table 1.

5 Discussion: Adaptive adjustment of the bolt pose

This section demonstrates the feasibility of the MAPPO algorithm and the effectiveness of the probabilistic tree method and adaptive reward strategy in a bolt pose adjustment simulation model. A spatial coordinate system is constructed with the bolt hole center axis as the z axis, as shown in Fig. 5. The dotted line represents the central axis of the bolt. There are two fine adjustment methods for the tightening shaft, corresponding to two actions a_x, a_y. The bolt has two states of tilt and an upright position.

For bolts with different postures, although the same tightening shaft fine-tuning method is used, the actions generated by the bolts are still different. We can see that the interior of the action space of the bolt system is interrelated, so a hierarchical tree of action space A is established. The calculation result of the hierarchical tree is shown in formula (17).

$$F = \left\{a_x \rightarrow \left\{M_y, T_x\right\}; a_y \rightarrow \left\{M_x, T_y\right\}\right\} \tag{17}$$

where a_x represents the displacement of the tightening shaft in the x direction, a_y represents the displacement of the tightening shaft in the y direction, M_y represents the rotation angle of the bolt along the y axis, M_x represents the rotation angle of the bolt along the x axis, T_x represents the parallel displacement of the bolt along the x axis, and T_y represents the parallel displacement of the bolt along the y axis, as in Fig. 6.

The posture of the bolt shaft can be expressed by the relative position of the central axis of the bolt shaft with respect to the central axis of the bolt hole. Due to the possibility of a heterogeneous spatial relationship between the central axis of the bolt shaft and the central axis of the bolt hole, it is difficult to express the spatial angle in a three-dimensional coordinate system, so the relative position relationship between the two axes is described by the coordinate plane projection method. As shown in Fig. 7. In addition, there are certain measurement errors when the posture data of the bolt are acquired in both the virtual and real assembly situations. In the simulation environment, the highest precision data type is double, which

Table 1. Model-driven Adaptive Proximal Policy Optimization (MAPPO)

```
Algorithm 1: MAPPO

Define the state space $\mathcal{S}$ and action space $\mathcal{A}$
1. Probability tree method
Simplify the action space $\mathcal{A}$ based on prior knowledge to form a hierarchical action space $F$.
Random $G$-sized action-state pairs are obtained from $\mathcal{A}$ and $F$
For $j$=1,2,⋯,Epochs do
    For $k$=1,2,⋯,batch do
        Input the action-state pairs to train the fully-connected neural network to build the coupling
        relationship for the action and state space, where state values are input data and action values
        are label data in the training process
    End for
    The cross-entropy loss function $L_N$ of each batch is obtained by:
        $L_N = -\frac{1}{G}\sum_k \left[ y_l \ln y_o + (1-y_l)\ln(1-y_o) \right]$
    Where, $y_l$ is the label of the neural network (action value), and $y_o$ is the real output of the
    neural network.
    The weight vector of the neural network can be refreshed by:
        $w^{k+1} = w^k - \alpha \nabla L_N$
    where $w^k$ is the $k$-th step of the weight vector, $\alpha$ is a learning rate, and $\nabla$ is a gradient
    operator.
The trained probability tree method is used to select the policy of the agent.
End for
2. Adaptive reward mechanism
For $i$=1,2,⋯,N do
    Run policy $\pi$ for $T$ timesteps, obtaining $\{S_t = \{S_1, S_2, \cdots, S_H\}, F_t = \{F_1, F_2, \cdots, F_d\}\}$,
    Normalize the state value by $S_i^{norm} = \frac{S_i}{max(S_i)}, i = 1,2,\cdots,H$ in formula (14)
    Define $Loss = \|(S^{norm} - S^G)\|_2$ in formula (15)
    Calculate the adaptive reward function by:
    $R(Loss) = \cos\frac{\pi}{2Loss_{\max}} Loss - 1, (0 \le Loss \le Loss_{\max})$ in formula (16)
3. RL training process
    For $u$=1,2,⋯,M do
        Calculate the final objective function of the agent in formula (10):
            $J_{ppo_2}^{\theta^k}(\theta) \approx \sum_{(S_t,A_t)} min\left(\frac{P_\theta(A_t|S_t)}{P_{\theta^k}(A_t|S_t)}\left(\sum_{t'>t}\gamma^{t'-t}R_{t'} - E_{\pi^\phi}[\sum_{k=1}^{\infty}\gamma^k R_{t+k+1}|S_t =\right.\right.$
            $\left.\left.S]\right), clip\left(\frac{P_\theta(A_t|S_t)}{P_{\theta^k}(A_t|S_t)}, 1-\varepsilon, 1+\varepsilon\right)\left(\sum_{t'>t}\gamma^{t'-t}R_{t'} - E_{\pi^\phi}[\sum_{k=1}^{\infty}\gamma^k R_{t+k+1}|S_t = S]\right)\right)$
        Update $\theta$ by a gradient method with regard to $J_{ppo_2}^{\theta^k}(\theta)$
    End for
End for
The optimal policy $\pi^*$ can be calculated by $\pi^* = \arg\max_\pi J_{ppo_2}^{\theta^k}$
```

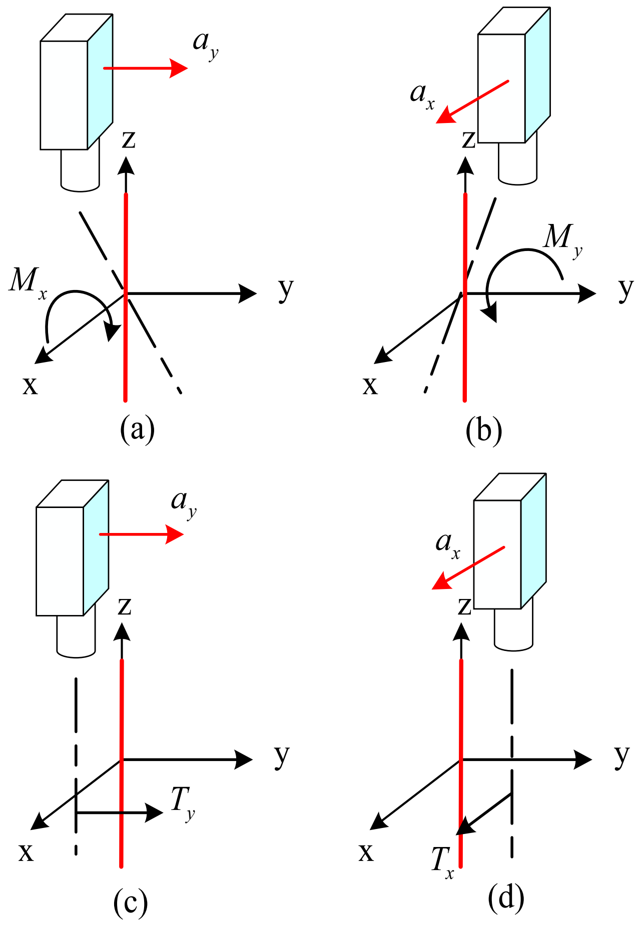


Fig. 5 Hierarchical tree of the bolt system

can ensure an accurate calculation for 15 decimal. Therefore, the error measurement of the simulation bolt can reach 1.0E-15 for both the angle (°) and displacement (mm) measurements. In a real assembly environment, the measurement accuracy cannot be as high as the simulation accuracy. The average vision measurement error is 0.1 mm for the

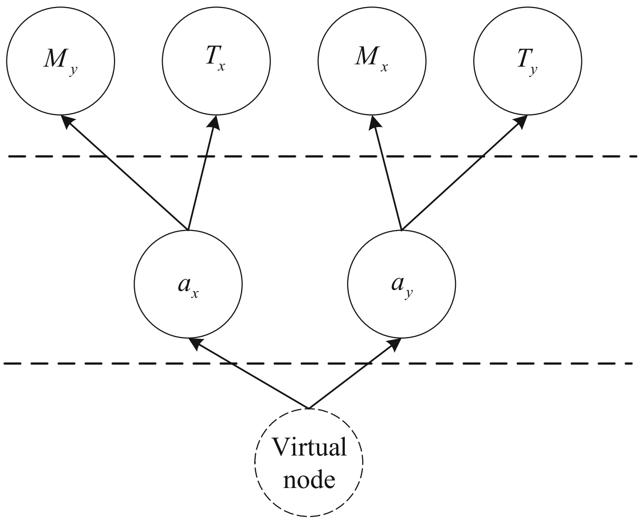


Fig. 6 Bolt system action space

displacement and 0. 1° for the angle. To make the simulation environment closer to the real environment, the precision of the variable is constrained to 0.1.

The central axis of the bolt axis is projected to three coordinate planes. For the inclined bolt central axis (that is, the central axis is not parallel to the z axis), the central axis is projected on the yz plane and xz plane, respectively, and the resulting projection line and Z axis form an intersection angle of $\theta_{yz,}\theta_{xz}$. The center point of the upper surface of the bolt head is projected on the xy plane to form a point, and the distance from this point to the x axis and y axis is S_y, S_x. For the central axis that has a deviation in position but does not tilt at an angle (that is, the central axis is parallel to the z axis), projecting the central axis on the xy plane will form a point, and the distances between the point and the x and y axes are S_y, S_x. Therefore, the state space S of the tightening process can be expressed in formula (18).

$$S = \{\theta_{yz,}\theta_{xz,}S_y,S_x\} \tag{18}$$

We can see that there is a coupling relationship between the action space and state space in this model. Since the hierarchical tree F has been constructed, the transition probabilistic formula of the hierarchical action space can be further refined in formula (19).

$$P_\theta(A_t|S_t) = \begin{cases} P_\theta(a_x, M_y), if\, \theta_{xz} \neq 0 \cup A_{upper} = a_x \\ P_\theta(a_x, T_x), if\, \theta_{xz} = 0 \cup A_{upper} = a_x \\ P_\theta(a_y, M_x), if\, \theta_{yz} \neq 0 \cup A_{upper} = a_y \\ P_\theta(a_y, T_y), if\, \theta_{yz} = 0 \cup A_{upper} = a_y \end{cases} \tag{19}$$

where A_{upper} represents the action space near the root node layer. Taking the first item in formula (19) as an explanation, if the state of θ_{xz} is not zero and the agent selects actiona_x in the current step, then the sub-action M_y is more likely to be executed in the bolt.

In summary, prior knowledge about the state can affect the choice of action. To effectively describe the effect of the state on the action choice, a state transfer neural network is established to describe this relationship. The establishing process of a neural network is shown in Fig. 8. This paper establishes a digital model of the bolt pose space in the tightening system and uses visual equipment to extract the state space vector S of the bolt system, where S_t represents the space vector of the relative position relationship of the bolt pose at time t.

As the relationship between the input and output data is not complex, a 3-layer fully-connected neural network is constructed to build the probability tree. The number of neurons in the input layer is the number of parameters in the state space. The input data are the value of each parameter in the state space. The number of neurons in the output layer is the number of parameters in the action space. The output data are the probability of each action parameter being selected.

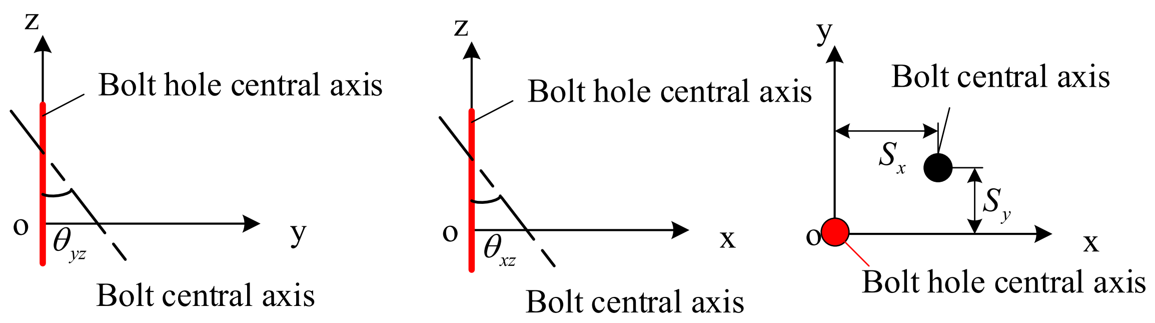


Fig. 7 Plane projection of the bolt central axis

The training and validation data are collected from random sampling based on prior knowledge in formula (19). In detail, the training data are composed of the state vector set $S = (S_i | S_i = \{\theta_{yz}, \theta_{xz}, S_y, S_x\}, i = 1, 2, \cdots, N)$, and the training labels consist of the four actions $A = \{(a_x, M_y), (a_x, T_x), (a_y, M_x), (a_y, T_y)\}$. A total of 4000 samples are collected for training, and 1000 samples are collected for validation. As it belongs to the classification problem, the loss function and activation function can be selected based on state-of-the-art works. Therefore, cross-entropy is adopted as the loss function of these neural networks. The activation function of the first and second neural network layers is a rectified linear unit (ReLU), and the last layer is a Softmax function.

The loss and accuracy of this neural network are shown in Figs. 9 and 10. After 50 epochs, the performance of the neural network is greatly improved, and the final classification accuracy is 70.9%. Compared with previous random exploration, the probability tree based on prior knowledge would have a 70.9% probability to guide the agent to select a better action and 29.1% probability to randomly explore the next action.

The adaptive reward mechanism is shown in Fig. 11:

5.1 State parameter normalization

The state space of the tightening system is $S = \{\theta_{yz}, \theta_{xz}, S_y, S_x, \}$. There are two kinds of parameters in space: the angle and displacement. The mathematical relationship cannot be established because of the different units of the two. Therefore, the parameters are normalized (see formula (20)).

$$\theta_{yz}^{norm} = \frac{\theta_{yz}}{\theta_{max}}, \theta_{xz}^{norm} = \frac{\theta_{xz}}{\theta_{max}}$$
$$S_y^{norm} = \frac{S_y}{S_{max}}, S_x^{norm} = \frac{S_x}{S_{max}} \tag{20}$$

where $\theta_{\max} = \frac{\pi}{2}$, which represents the maximum angle between the bolt central axis and z axis; S_{max} represents the bolt hole radius, that is, the maximum deviation of the bolt posture. Based on the 2-norm method, the total error of bolt position and attitude can be obtained (see formula (21))

$$Loss = \| \left(\theta_{yz}^{norm}, \theta_{xz}^{norm}, S_y^{norm}, S_X^{norm} \right) \|_2 \tag{21}$$

5.2 Reward function design

To meet the refinement conditions of a large rate of exploration for the bolt posture and a small rate of exploration when the posture error is large, the adaptive reward function is designed by using the properties of the trigonometric function, which is shown in formula (22). The trend of the function change is shown in Fig. 9.

$$R(Loss) = \cos \frac{\pi}{2Loss_{max}} Loss - (0 \le Loss \le 2) \tag{22}$$

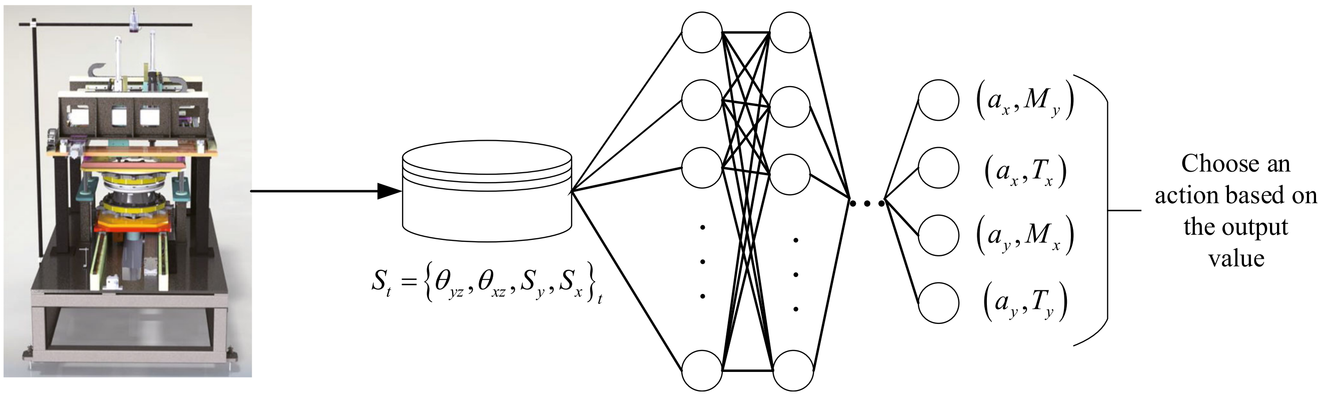

$S_t = \{\theta_{yz}, \theta_{xz}, S_y, S_x\}_t$
(a_x, M_y)
(a_x, T_x)
(a_y, M_x)
(a_y, T_y)
Choose an action based on the output value

Fig. 8 Neural network between state and action space

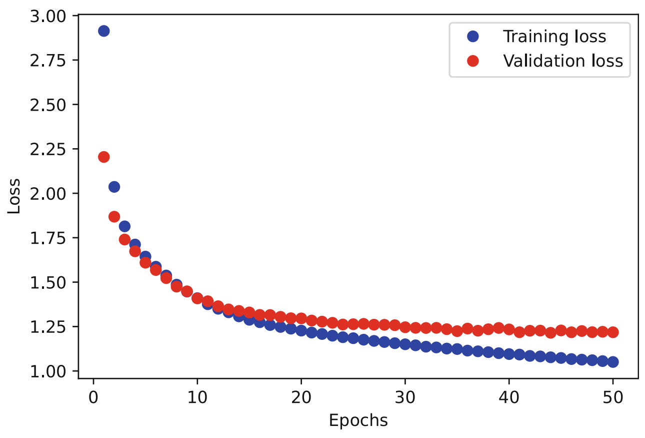


Fig. 9 The loss curve of the training process

6 Experiment and verification

To validate the advantage of the proposed algorithm, a tightening system action simulation platform is built based on the Unity physics engine, where the Unity software has a dynamic simulation function, which can accurately simulate the dynamic behavior and interference behavior of objects. The Python language is selected to run the reinforcement learning algorithm. To establish the digital connection between Python and the Unity platform, the communication interface is used to implement the Unity platform mathematical model and interact with Python software. The virtual model of the tightening system established in this paper is shown in Fig. 10. The system consists of support parts, a tightening system, a servo movement system, bolted connections, and vision equipment. Among them, the servo movement system can control the tightening axis of the tightening system to move in the x-y direction, and the vision device located above the bolt connection can capture the relative position of the bolt space. As the proposed algorithm is model-driven, the agent will take actions according to the current state, and the state

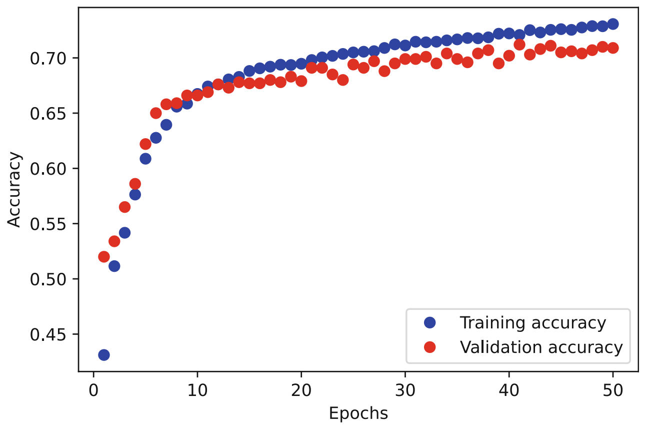


Fig. 10 The accuracy curve of the training process

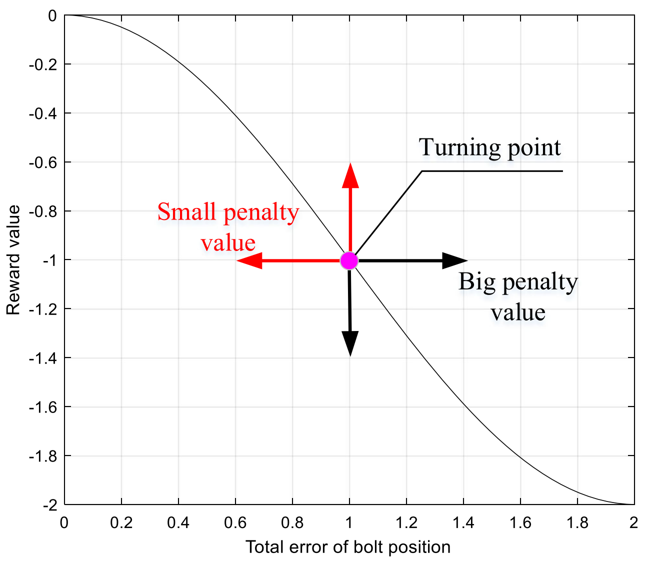


Fig. 11 Reward function of the bolt system

will be immediately perceived by vision equipment at the next moment, increasing the correlation between the simulation and the real world.

In the bolt posture adjustment simulation environments, the number of neural network layers is 3, the maximum number of steps is 5 million, and the loss coefficient λ is 0.99. In each epoch, the initial inclination angle of the bolt θ_{xz}, θ_{yz} is selected randomly in a small range under the boundary condition of the bolt hole. The initial positions of the bolts S_x, S_y are randomly determined in a small range and the initial position should not exceed the bolt hole radius. At each time step, the position and tilt angle of the bolt will be captured by the visual device.

6.1 Probabilistic tree method validation

The performance of the probabilistic tree method is evaluated by comparing it with the traditional PPO algorithm. When the probabilistic tree method is not used, the bolt system has a total of 6 action spaces. As shown in Figs. 12, 13 and 14, the agent using the probabilistic tree method can converge faster and learn a better adjustment policy with fewer steps for the loss trend of the bolt system learning process.

We can see that in Fig. 13, the convergence speed of the optimal strategy is improved after we add the probability tree method, and it can reach near the optimal strategy after 200,000 iterations, while the model without the probability tree method needs more than 500,000 iterations to find the optimal strategy. In Fig. 14, the length of the steps needed for the improved model to find the optimal solution decreases, while the traditional method does not have such a good convergence performance. Therefore, we can conclude that the probability tree method can improve the convergence speed and reduce the training time.

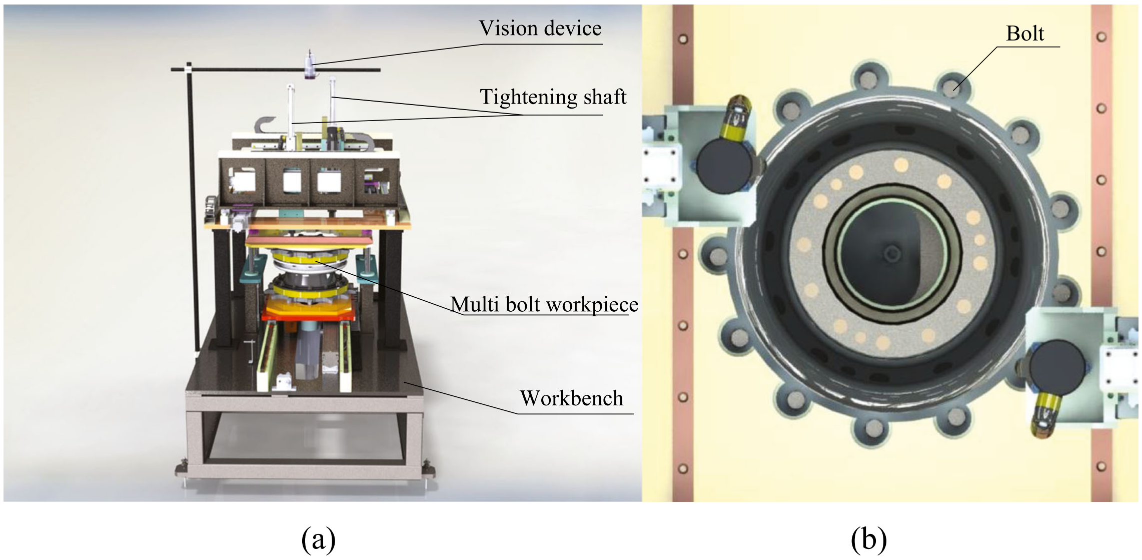


Fig. 12 Virtual model of the tightening system. (**a**) Tightening system in operation. (**b**) Picture from the perspective of the visual equipment

6.2 Adaptive reward mechanism

To verify the superiority of the adaptive reward mechanism, we first describe the traditional reward mechanism commonly used by scholars, and then compare the proposed method with the traditional reward function mechanism in three different cases. The traditional reinforcement learning reward mechanisms include:

1) Hierarchical reward method: Divide the action space into multiple intervals and assign different reward values according to the distance from the interval to the target;
2) Sparse reward method: Set reward values only at target points;
3) Linear reward method: The reward assigned is linear with the distance of the target.

The validation results are shown in Figs. 15 and 16.

It can be seen from Fig. 13 that after 500,000 iterations of the algorithm, the trigonometric function reward method has the lowest loss value and the most stable downward trend, while the final value reward method and the linear reward method have obvious downward effects, but the loss value fluctuates greatly at the beginning of the iteration. It can be seen from Fig. 14 that the cumulative reward values of the trigonometric function reward method and final value reward method are stable near 0, but the cumulative reward values of the HRM method are in a large negative range, that is, they fall into a local minimum and cannot extricate themselves.

In summary, the trigonometric function method has faster convergence and more stable performance than the traditional reward mechanism and does not easily to fall into the local optimum.

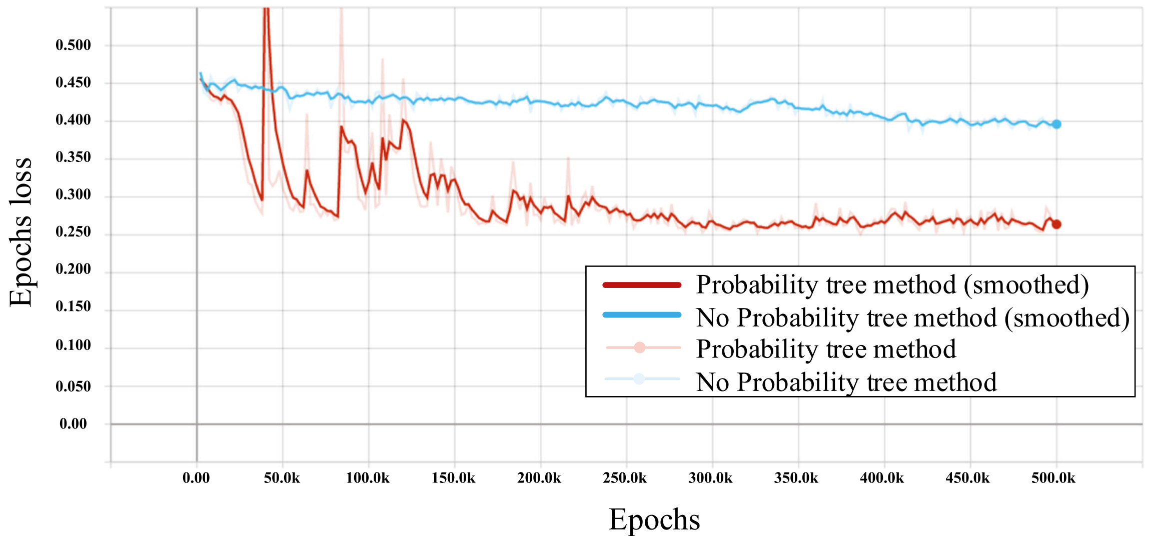


Fig. 13 Epochs length of the bolt system learning process

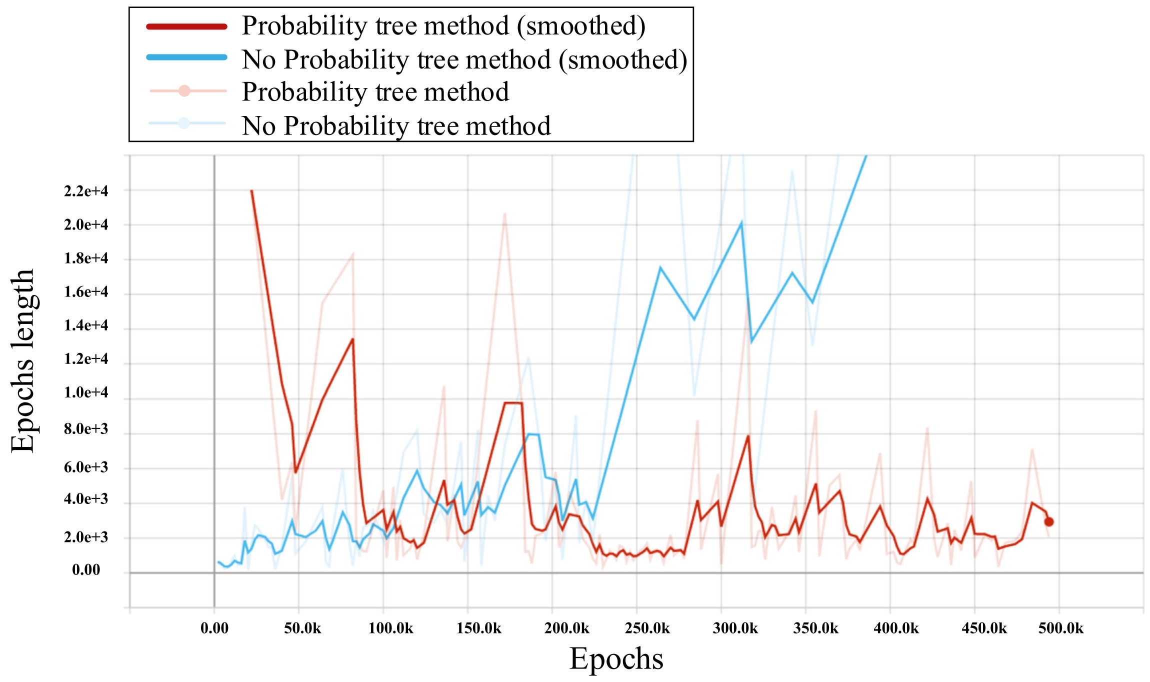


Fig. 14 Performance curves for the adaptive reward system for epoch loss

6.3 State-of-the-art of MAPPO

To prove the robustness and generalization ability of MAPPO, the classic RL task pendulum proposed by OpenAI Gym is adopted as the simulation platform, and three state-of-art RL methods DDPG [26], SAC [27] and TRPO [28] are selected as experimental comparison objects. The performance results of MAPPO and other state-of-art RL methods are shown in Figs. 17 and 18. In Fig. 17, the MAPPO and SAC methods achieve better performance than the other methods in terms of the learning effect, as the accumulated rewards of these methods are higher than those of the other methods. However, the training time of the SAC method is much longer than that of the other algorithms, which can be shown in Fig. 18. It can be concluded that MAPPO is the state-of-the-art method in terms of comprehensive performance.

7 Conclusion

Combined with the probabilistic tree and adaptive reward mechanism, the proposed MAPPO algorithm can perform well for the bolt posture adjustment. The state-of-the-art and robust nature of this algorithm has been demonstrated through

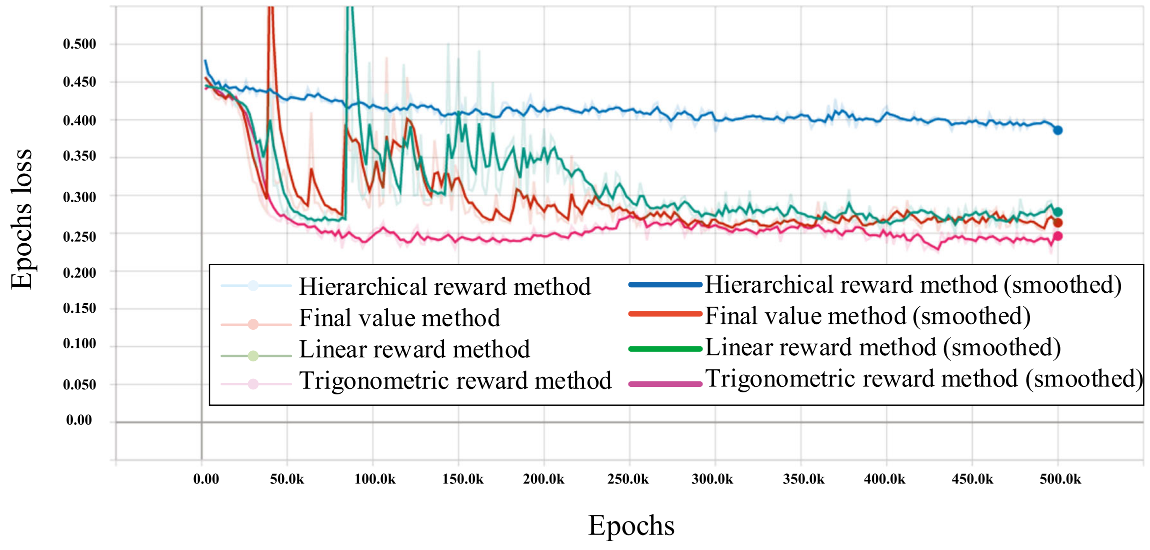


Fig. 15 Performance curves for the adaptive reward system with epoch loss

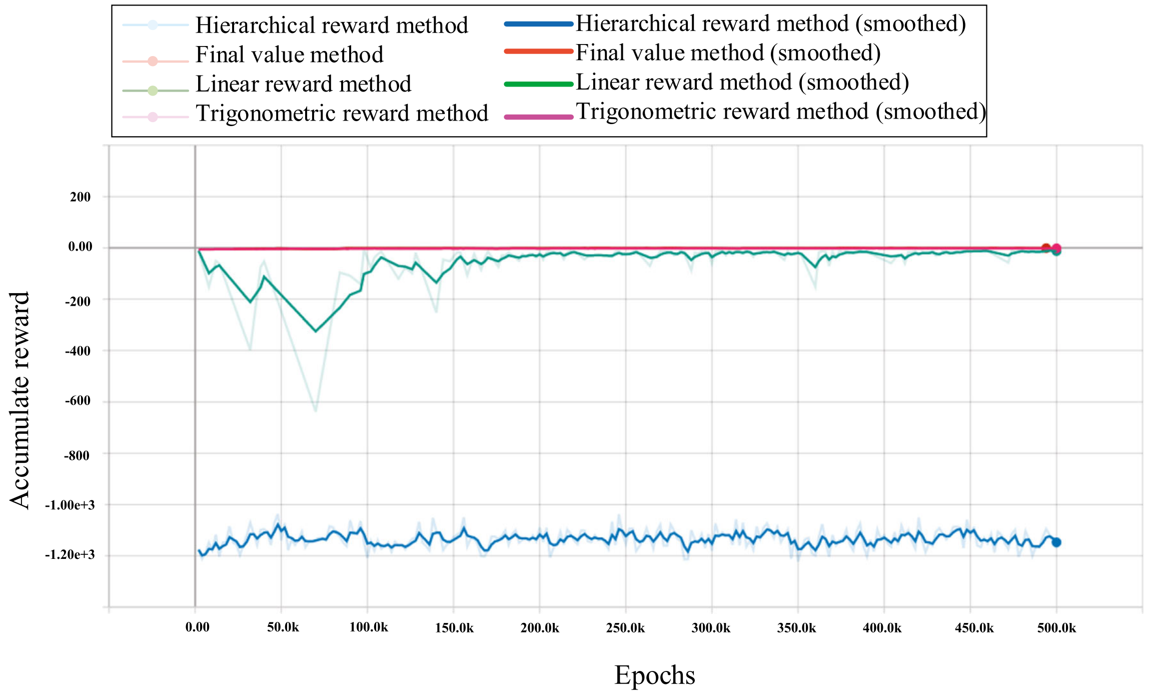


Fig. 16 Performance curves for adaptive reward system for accumulated rewards

a Unity simulation model based on a physics engine. As the digital model proposed in this article can accurately simulate the mechanical state of the bolt in the bolt hole adjustment, the results are convincing.

The probabilistic tree model mainly solves the problem of algorithm complexity caused by the large and spatially coupled parameters of the agent system action space. The hierarchical relationship between the parameters of the action space is established by a probabilistic tree, which simplifies the complexity of the model.

The state-based adaptive reward mechanism reduces the chance that the agent is prone to fall into the local minimum when exploring the continuous action space.

In the proposed model, it is assumed that the initial bolt posture changes randomly within a small range at each episode, but for engineering applications, the initial bolt posture error emerges regularly. In the future, prior knowledge of the bolt postures from a real bolt tightening system will be applied to the proposed model to improve the application value of this model.

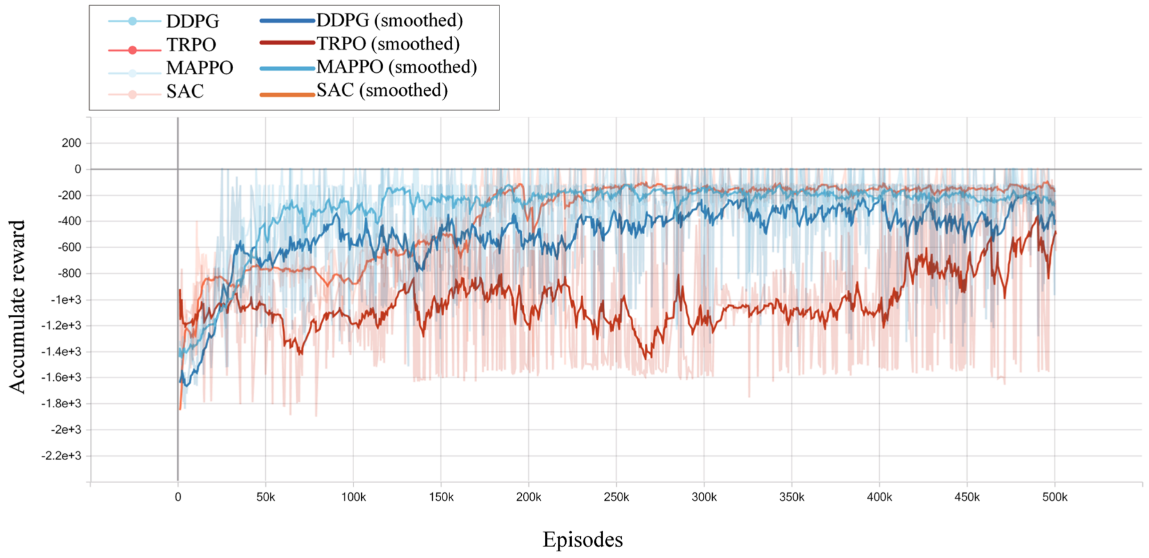


Fig. 17 Performance of the state-of-the-art methods for accumulated rewards

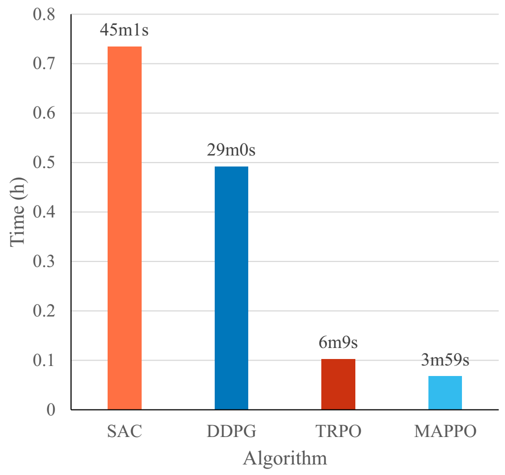


Fig. 18 Training time of the state-of-the-art methods for accumulated rewards

Acknowledgments We gratefully acknowledge the financial support from the National Defence Basic Scientific Research Program of China (JCKY2018208A001), and Tsinghua University-Weichai Power Joint Institute of Intelligent Manufacturing (JIIM02).

Compliance with ethical standards

Declaration of interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Acharya UR, Fujita H, Lih OS, Hagiwara Y, Tan JH, Adam M (2017) Automated detection of arrhythmias using different intervals of tachycardia ECG segments with convolutional neural network. Inf Sci 405:81–90. https://doi.org/10.1016/j.ins.2017.04.012
2. Sudarshan VK, Mookiah MRK, Acharya UR, Chandran V, Molinari F, Fujita H, Ng KH (2016) Application of wavelet techniques for cancer diagnosis using ultrasound images: a review. Comput Biol Med 69:97–111. https://doi.org/10.1016/j.compbiomed.2015.12.006
3. Acharya UR, Fujita H, Oh SL, Hagiwara Y, Tan JH, Adam M (2017) Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals. Inf Sci 415-416:190–198. https://doi.org/10.1016/j.ins.2017.06.027
4. Capuano N, Chiclana F, Fujita H, Herrera-Viedma E, Loia V (2018) Fuzzy group decision making with incomplete information guided by social influence. IEEE Trans Fuzzy Syst 26(3):1704–1718. https://doi.org/10.1109/TFUZZ.2017.2744605
5. Protopapadakis E, Voulodimos A, Doulamis A, Doulamis N, Stathaki T (2019) Automatic crack detection for tunnel inspection using deep learning and heuristic image post-processing. Appl Intell 49(7):2793–2806. https://doi.org/10.1007/s10489-018-01396-y
6. Villalonga A, Beruvides G, Castaño F, Haber RE (2020) Cloud-based industrial cyber–physical system for data-driven reasoning: a review and use case on an industry 4.0 pilot line. IEEE Trans Ind Informatics 16(9):5975–5984. https://doi.org/10.1109/TII.2020.2971057
7. Gullapalli V, Franklin JA, Benbrahim H (1994) Acquiring robot SKILLS via reinforcement learning. IEEE Control Syst Mag 14(1):13–24. https://doi.org/10.1109/37.257890
8. Yang BH, Asada H (1996) Progressive learning and its application to robot impedance learning. IEEE Trans Neural Netw 7(4):941–952. https://doi.org/10.1109/72.508937
9. Nuttin M, VanBrussel H (1997) Learning the peg-into-hole assembly operation with a connectionist reinforcement technique. Comput Ind 33(1):101–109. https://doi.org/10.1016/s0166-3615(97)00015-8
10. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. IEEE Trans Neural Netw 9(5):1054–1054. https://doi.org/10.1109/TNN.1998.712192
11. Ding S, Du W, Zhao X, Wang L, Jia W (2019) A new asynchronous reinforcement learning algorithm based on improved parallel PSO. Appl Intell 49(12):4211–4222. https://doi.org/10.1007/s10489-019-01487-4
12. Liu P, Zhao Y, Zhao W, Tang X, Yang Z (2019) An exploratory rollout policy for imagination-augmented agents. Appl Intell 49(10):3749–3764. https://doi.org/10.1007/s10489-019-01484-7
13. Yang CG, Zeng C, Cong Y, Wang N, Wang M (2019) A learning framework of adaptive manipulative Skills from human to robot. Ieee Trans Ind Informatics 15(2):1153–1161. https://doi.org/10.1109/tii.2018.2826064
14. Wan A, Xu J, Chen HP, Zhang S, Chen K (2017) Optimal path planning and control of assembly robots for hard-measuring easy-deformation assemblies. Ieee-Asme Trans Mechatron 22(4):1600–1609. https://doi.org/10.1109/tmech.2017.2671342
15. Xu J, Hou ZM, Wang W, Xu BH, Zhang KG, Chen K (2019) Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole assembly tasks. Ieee Trans Ind Informatics 15(3):1658–1667. https://doi.org/10.1109/tii.2018.2868859
16. Young Ho K, Lewis FL (2000) Reinforcement adaptive learning neural-net-based friction compensation control for high speed and precision. IEEE Trans Control Syst Technol 8(1):118–126. https://doi.org/10.1109/87.817697
17. Deisenroth MP, Fox D, Rasmussen CE (2015) Gaussian processes for data-efficient learning in robotics and control. IEEE Trans Pattern Anal Mach Intell 37(2):408–423. https://doi.org/10.1109/TPAMI.2013.218
18. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv e-prints
19. Mnih V, Badia AP, Mirza M, Graves A, Harley T, Lillicrap TP, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning, vol 48. Paper presented at the proceedings of the 33rd International Conference on International Conference on Machine Learning, New York
20. Wang Z, Bapst V, Heess N, Mnih V, Munos R, Kavukcuoglu K, de Freitas N (2016) Sample efficient actor-critic with experience replay
21. Mousavi SS, Schukat M, Howley E (2018) Deep reinforcement learning: an overview. In: Bi Y, Kapoor S, Bhatia R (eds) Proceedings of Sai Intelligent Systems Conference, vol 16. Lecture Notes in Networks and Systems. pp 426-440. https://doi.org/10.1007/978-3-319-56991-8_32
22. Singh S, Lewis RL, Barto AG, Sorg J (2010) Intrinsically motivated reinforcement learning: an evolutionary perspective. IEEE Trans Auton Ment Dev 2(2):70–82. https://doi.org/10.1109/tamd.2010.2051031
23. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533. https://doi.org/10.1038/nature14236

24. Gershman SJ, Daw ND (2017) Reinforcement learning and episodic memory in humans and animals: an integrative framework. Annu Rev Psychol 68:101–128. https://doi.org/10.1146/annurev-psych-122414-033625
25. Lewis FL, Vamvoudakis KG (2011) Reinforcement learning for partially observable dynamic processes: adaptive dynamic programming using measured output data. IEEE Trans Syst Man Cybernetics Part B (Cybernetics) 41(1):14–25. https://doi.org/10.1109/TSMCB.2010.2043839
26. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. Computer Science
27. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. Paper presented at the Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research,
28. Schulman J, Levine S, Moritz P, Jordan MI, Abbeel P (2015) Trust region policy optimization. arXiv e-prints:arXiv:1502.05477