TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 11/24 pp1422–1440
DOI: 10.26599/TST.2023.9010100
Volume 29, Number 5, October 2024

Offline Reinforcement Learning with Constrained Hybrid Action Implicit Representation Towards Wargaming Decision-Making

Liwei Dong, Ni Li, Guanghong Gong*, and Xin Lin*

Abstract: Reinforcement Learning (RL) has emerged as a promising data-driven solution for wargaming decision-making. However, two domain challenges still exist: (1) dealing with discrete-continuous hybrid wargaming control and (2) accelerating RL deployment with rich offline data. Existing RL methods fail to handle these two issues simultaneously, thereby we propose a novel offline RL method targeting hybrid action space. A new constrained action representation technique is developed to build a bidirectional mapping between the original hybrid action space and a latent space in a semantically consistent way. This allows learning a continuous latent policy with offline RL with better exploration feasibility and scalability and reconstructing it back to a needed hybrid policy. Critically, a novel offline RL optimization objective with adaptively adjusted constraints is designed to balance the alleviation and generalization of out-of-distribution actions. Our method demonstrates superior performance and generality across different tasks, particularly in typical realistic wargaming scenarios.

Key words: offline Reinforcement Learning (RL); wargaming; decision-making; hybrid action space

1 Introduction

Wargaming is a critical decision-support tool widely used in military and security domains to evaluate strategies for real-world operations[1]. Generally, wargaming can be described as a simulation of conflict or competition where individuals make decisions targeting specific tasks and respond to the outcomes of those decisions[2]. In modern wargaming on computer-based simulation systems, a decision-making model is needed to produce sequential commands under various wargaming scenarios. However, traditional wargaming decision-making methods are often based on knowledge-driven approaches[3, 4], which lack the flexibility to handle complex and dynamic wargaming scenarios.

Reinforcement Learning (RL) has shown its effectiveness and superiority in various challenging real-world decision-making tasks, such as energy management[5, 6], satellite scheduling[7], mobile navigation[8, 9], neural speech enhancement[10], manufacturing scheduling[11], vehicle charging scheduling[12], and math solving[13], etc. Therefore, RL recently has emerged as a promising data-driven solution for wargaming decision-making problems[14–16]. However, these prior works have not

• Liwei Dong and Xin Lin are with School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China. E-mail: vivian_keith@buaa.edu.cn; lx@buaa.edu.cn.
• Ni Li is with School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China; State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China; and also with Zhongguancun Laboratory, Beijing 100191, China. E-mail: lini@buaa.edu.cn.
• Guanghong Gong is with School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China; and also with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China. E-mail: ggh@buaa.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2023-07-24; revised: 2023-08-31; accepted: 2023-09-18

explored the following two critical domain issues: First, these works adopt an inefficient online paradigm, which requires a tremendous amount of online training costs. Abundant offline domain data produced by historical decision-making models are neglected to accelerate RL's deployment. These data can be leveraged by the offline RL paradigm to train a reasonable policy while avoiding massive online interactions. Second, wargaming decision-making commands naturally form a hybrid action space with both discrete and continuous action properties (detailed in Section 3.2.1). However, most previous work directly conducts discrete/continuous control or prunes the action space based on scenario-specific priors, which probably leads to inadequate control performance. Noteworthy, these two significant issues also exist in a wide range of other real-world domains, like automatic driving. Specifically, the hybrid action space is a more general and natural action abstraction paradigm, and it is increasingly needed to leverage rich and readily available domain offline data to rapidly deploy of RL-based decision-making model. Therefore, we seek an effective RL schema to address above two issues.

Many studies have been conducted on offline RL[17–24] that can learn from pre-collected static datasets, and RL in hybrid action space[5, 25–30] that can deal with discrete-continuous hybrid control tasks. However, current RL methods cannot simultaneously handle these two challenging issues. Specifically, the action space is either continuous or discrete in existing offline RL methods. Conversely, existing RL algorithms targeting hybrid actions are designed and validated under the online paradigms. Moreover, although some RL algorithms[25, 26] targeting hybrid action space build upon off-policy algorithms, a significant decrease in the performance of off-policy algorithms is demonstrated when applied directly to static offline datasets[17, 18]. Thereby, there is currently a strong need for RL approaches targeting hybrid actions tailored for offline settings.

Aiming at the current dilemma at the significant intersection of offline RL and hybrid control, this paper proposes Constrained Hybrid Action Implicit Representation (CHAIR), a novel offline RL method targeting hybrid action space. CHAIR comprises two learning stages: an implicit representation learning stage and a constrained offline RL stage.

In the implicit representation learning stage, we transform the problem from hybrid control into continuous control, by learning an implicit representation of the original hybrid action within the offline dataset's support. Inspired by recent advances in representation learning[18, 25, 31], we develop a new constrained action representation technique, which implicitly maps the original hybrid action space into a semantically consistent and compact laten representation space. It is allowed to learn a continuous latent policy in the constructed representation space with better exploration feasibility and action scalability. The representation is reversible, that is, the learned latent policy can be reconstructed as a hybrid policy needed for the deployment in realistic environments. In contrast to a previous work, Hybrid Action Representation (HyAR)[25], which uses the similar representation technique and relies on online interactions to further refine the representation, CHAIR develops more strict representation constraints to build a more semantically consistent representation space within the support of the offline dataset. To be specific, CHAIR makes three key improvements to meet offline settings: (1) explicitly characterizing the dependency between the discreate and continuous components according to the natural decision-making sequence in the hybrid action space; (2) explicitly constraining the reconstruction shift of the discrete action component; and (3) constraining the similarity between the original and reconstructed hybrid actions by explicitly modeling their consistent impacts on environmental dynamics.

In the constrained offline RL stage, we perform offline RL for the latent policy learning in the constructed latent representation space. To alleviate the Out-Of-Distribution (OOD) action shift problem[20, 32] under offline settings, we augment the offline RL optimization objective using two critical techniques: introducing a behavior cloning penalty and using a maximum-minimum combined Q-learning target. Moreover, we develop an adaptive weight adjustment mechanism to dynamically align and control the components of the offline RL optimization objective during learning. The dynamical adjustment balances the alleviation and generalization of OOD actions, resulting in better policy robustness and flexibility.

Through comprehensive experimental evaluation on toy game tasks and typical wargaming scenarios, our

method shows its superior performance and good stability and generality. Unlike previous domain works deploying RL into the wargaming field, our method can effectively learn reasonable decision-making policies with hybrid commands only based on offline data. The main contributions of this paper are summarized as follows:

- Targeting domain demands, we propose a novel offline RL method crafted for discrete-continuous hybrid action spaces using the action representation technique.
- We develop a new constrained hybrid action representation technique crafted for offline RL. This technique realizes a bidirectional mapping between the original hybrid action space and a latent space in a semantically consistent way. The representation space within the support of the offline dataset comes with better exploration feasibility and action scalability, allowing training a latent policy that can be reconstructed back to a needed hybrid policy.
- We design a novel augmented offline RL optimization objective with constraint components to further constrain the OOD action shift. Correspondingly, an adaptive weight adjustment approach is developed to dynamically control the objective components for a better balance between conservatism and generalization of the offline RL policy.

The remainder of this paper is structured as follows. In Section 2, we review and discuss related previous works. Section 3 provides the formulation of key concepts and the domain problem. In Section 4, we present our proposed method in detail. In Section 5, we evaluate our method on toy game tasks to validate its preliminary effectiveness. In Section 6, we present a systematic case study of our proposed method in typical wargaming scenarios. Section 7 provides a discussion, and Section 8 is the summary of this work.

2 Related Work

2.1 Offline reinforcement learning

In offline RL, the agent must rely on a static dataset of historical data to learn the policy[32]. Many offline RL methods build on top of the existing off-policy RL algorithms, but additionally constrain the notorious OOD action shift to deal with the extrapolation error problem under offline settings. For example, REM[33] uses a randomized Q-function ensemble to learn a robust value estimation that mitigates the overestimation of potential OOD future actions. Other work suggests to prevent OOD actions by constraining the learning policy with some explicit regularization, such as employing a Kullback-Leibler (KL) divergence or Maximum Mean Divergence (MMD)[20, 34], leveraging auxiliary behavioral cloning loss[19], and using a model-based paradigm with conservative penalties[35].

To achieve the constraint, our method mainly follows another direction that uses the action representation technique to model the action distribution in the offline dataset to implicitly constrain the policy. For example, BCQ[17] uses Conditional Variational Auto-Encoder (CVAE)[36] to produce a close action distribution, and constrains the policy by sampling from the distribution. Polocy in the Latent Action Space (PLAS)[18], a similar work to ours, employs the CVAE model to learn a latent action space where a latent policy is directly optimized, which implicitly constrains the policy by action construction. However, PLAS primarily performs the updates in the original action space with decoded one-step future action, which is only applicable to continuous action spaces, whereas our method specifically designed for hybrid action spaces is executed totally in the latent space constrained within the support of the offline dataset. Nevertheless, the action space is either continuous or discrete in these previous studies, that is, they are unable to deal with hybrid actions.

2.2 Reinforcement learning in hybrid action spaces

It is intractable to effectively deal with a complex heterogeneous discrete-continuous hybrid action space by using most traditional RL algorithms directly. Currently, research on this problem is still relatively limited. Some study puts efforts into modifying the existing RL algorithms to align with the hybrid action space.

For example, Parameterized Action with Deep Deterministic Policy Gradient (PADDPG)[29] modifies the Deep Deterministic Policy Gradient (DDPG) framework to let the actor output a unified continuous vector that concatenates the values of all discrete actions and all of their corresponding continuous parameters. This unnatural approach somewhat introduces parameterization redundancy, which has a scalability problem in high-dimensional scenarios.

Hybrid Proximal Policy Optimization (HPPO)[30] modifies the Proximal Policy Optimization (PPO) framework to use a state-value critic to address the over-parameterization problem of DDPG, and employs two parallel actors with different output structures to model the discrete and continuous components, respectively, as mentioned earlier. However, the neglect of the implicit dependence between the discrete and continuous components in these two methods may pose problems, as this dependence is crucial for modeling the original property of hybrid actions[25]. Pamameterized Deep Q-Nerwork (PDQN)[26] considers such a dependence by combining the Deep Q-Network (DQN) and DDPG structures, where the DDPG actor outputs all continuous parameters and the DQN critic takes a discrete action concatenated with all of these parameters to output a value estimate. However, PDQN still comes with the parameter redundancy problem. A recent work, HyAR[25] provides another direction based on the action representation technique[18, 37] to address the hybrid action issue. It maps the original hybrid action space into a unified and decodable representation space, and let the RL agent learn a latent policy in such a space, which resembles the paradigm of PLAS[18]. However, these previous work has demonstrated to be effective in online settings. There is currently a lack of RL approaches targeting hybrid actions tailored for offline settings.

2.3 Wargaming decision-making

Traditional wargaming decision-making has heavily relied on the knowledge-based paradigm, such as rule-based expert systems[3], Bayesian inference[38], and Finite State Machine (FSM)[39]. However, these approaches have limitations in dealing with modern complex wargaming scenarios, lacking flexibility and error tolerance. Specifically, it is challenging to develop a comprehensive rule-based expert system with sufficient inference rules covering massive wargaming situations. Bayesian inference is also severely affected by the designer's expertise, which determines how well the Bayesian network structure can extract and perceive the wargaming situations. For FSMs, determining the state transitions scenario by scenario may be challenging.

In recent times, there has been a growing interest amongst researchers in using RL methods for decision-modeling in the field of wargaming[14–16]. This can be seen as a data-driven paradigm that has demonstrated potential in some wargaming scenarios. However, these previous work has primarily focused on the deployment of RL, i.e., adapting existing RL algorithms to some wargaming scenarios. They train RL-driven decision-making models from scratch with an online paradigm, which largely relies on massive interactions with the wargaming environment, resulting in low training efficiency. Moreover, few of these previous works consider modeling the wargaming decision-making problem in a natural hybrid action space. Most of them consider simple continuous control or discrete control by manually discretizing the continuous parameters.

3 Background

3.1 Preliminaries

(1) Markov decision process. In RL, an agent interacts with its environment sequentially, which can be modeled as a standard Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ consisting of five components: a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, state transition probabilities $\mathcal{P}: \mathcal{S}\times\mathcal{A}\times\mathcal{S}\to[0,1]$, a reward function $\mathcal{R}: \mathcal{S}\times\mathcal{A}\times\mathcal{S}\to\mathbf{R}$, and a discount factor $\gamma\in[0,1]$. At each time step $t\in\{0, 1, \ldots, T\}$ in an MDP, the agent continuously performs its policy $\pi: \mathcal{S}\to\mathcal{A}$, then obtains a instant reward r_t and transfers to a next state $s_{t+1}\in\mathcal{S}$ following $\mathcal{P}$. Such a trahectory segment (s_t, a_t, r_t, s_{t+1}) is referred to as a transition, often denoted by (s, a, r, s'). The goal of the RL agent is to find an optimal policy π^* that maximizes the expected discounted cumulative reward (i.e., return), that is $\pi^* = \underset{\pi}{\text{argmax}}\, E_\pi[\sum_t \gamma^t r_t]$, where E denotes the expectation operator. The action-value function (Q-function) is defined as $Q^\pi(s, a) = E_\pi[\sum_t \gamma^t r_t \mid s_0 = s, a_0 = a]$, and always updated using Temporal Difference (TD) learning[40] with Bellman iterations,

$$Q^\pi(s, a) = E_{s'}[r_t + \gamma Q^\pi(s', a')] \tag{1}$$

where s' denotes the state on the next time step, and $a' = \pi(s')$ is the action on the next time step.

DDPG[41], which is crafted for continuous control with Actor Critic (AC) framework. The actor of DDPG is a deterministic parameterized policy $\pi_\omega(s)$, that is optimized with respect to the critic, a parameterized action-value estimate $Q_\theta^{\pi_\omega}(s,a)$. DDPG optimizes the policy following Deterministic Policy Gradient

(DPG)[42],

$$\omega \leftarrow \underset{\omega}{\operatorname{argmax}}\ E_s\left[Q_\theta^{\pi_\omega}(s, \pi_\omega(s))\right] \tag{2}$$

which corresponds to learning an approximation to the maximum of a critic Q_θ, with the gradient ascent propagating through both an actor $\pi_\omega(s)$ and the critic Q_θ. Our method in this paper builds on the top of TD3[43], an efficacious enhancement of DDPG.

(2) Offline reinforcement learning. An offline RL problem can be described in the following way: We are given a static dataset $\mathcal{D} = \left\{\left(s_i, a_i, r_i, s_i'\right)\right\}_{i=1}^{N}$ with a finite number of transitions that are beforehand collected with one or more behavioral policies. Only by exploiting $\mathcal{D}$, the problem of offline RL is to learn a policy that can achieve the highest possible cumulative reward when it can be applied to real MDPs[24].

Offline RL suffers the extrapolation error induced by OOD actions, also known as the action distribution shift problem[17, 20, 32]. Specifically, the distribution of actions in the offline dataset may differ significantly from the distribution of actions, that the agent would select if it was to interact with the environment in real-time. Thus, the agent may learn to exploit biases in the offline dataset, eventually leading to a suboptimal policy performing poorly in the real environment. In terms of TD learning with Eq. (1), when we bootstrap $Q^\pi(s,a)$ using $Q^\pi(s',a')$ that are missing or rare in the offline dataset, the value estimation may be accumulated to be arbitrarily wrong[17, 18].

(3) Hybrid action space. Following the classic notations[27], a hybrid action space can be described in the following mathematical way: A hybrid action a_k is denoted as a tuple (k, x_k), where $k \in \mathcal{K}$ is a discrete action selected from a finite set $\mathcal{K} = \{k_1, k_2, \ldots, k_K\}$, and each discrete action has its corresponding continuous parameter $x_k \in \mathcal{X}_k \subseteq \mathbf{R}^{m_k}$ where m_k denotes the parameter dimension. Then, the entire hybrid action space $\mathcal{A}$ is given,

$$\mathcal{A} = \bigcup_{k \in \mathcal{K}} \{a_k := (k, x_k) \mid x_k \in \mathcal{X}_k\} \tag{3}$$

which denotes the union of each optional discrete action with all its possible parameters.

There is a noteworthy dependency between the two heterogeneous components (discrete actions and continuous parameters) of hybrid actions[25]. That is, a discrete action determines the dimension, the valid range, and the practical semantics of its associated continuous parameters.

Considering the dependency, the RL agent in a hybrid action space should perform the natural decision-making sequence, that is, it should select the discrete action first based on the current state, and then choose the corresponding continuous parameters.

3.2 Problem formulation

3.2.1 Problem statement

This paper aims at the domain problem of wargaming In this study, we concentrate on the domain decision-making, detailed below problem for simulated wargaming scenarios. Below, we present this domain problem and elucidate some relevant technical concepts.

We first elucidate some relevant domain concepts as follows. A force unit comprises a controllable group of equipment platforms of a specific size, governed by certain formation rules, such as aircraft, warships, or tanks. In computer-simulated wargaming, a force unit is represented as a wargaming entity that captures its physical and behavioral characteristics. These wargaming entities respond to decision commands when activated in the simulation. A wargaming scenario refers to the specific simulated wargaming confrontation circumstances, encompassing mission goals, the wargaming area, and the exact composition of the fighting forces, among other factors. In general, the fighting forces in a wargaming scenario are divided into two sides: red and blue. Each side has its own collaborative force units, whose actions are controlled to fulfill a specific combat mission, such as occupying the opponent's command base or bombing the opponent's ship.

A Course Of Action (COA) represents a high-level command that covers valid actions with clear semantic information for all individual units, such as "Unit one attacks the targets in area A", "Unit two moves to area B", and so on. In wargaming tasks, a COA is often a typical hybrid action. For example, when deciding to execute an attack action, it is necessary to determine the associated attack distance; when executing a movement action, it is needed to choose the associated position and range of the movement destination area.

At each simulation time step in a wargaming scenario, the force units of a specific side need to be commanded collaboratively to complete the specific mission. Hence, the key problem of this study is constructing a decision-making model to generate sequential COAs conditioned on the dynamic

wargaming situation for mission completion. The decision-making model takes the current wargaming situation as input and produces valid COAs using decision-making algorithms.

3.2.2 Modeling schema

In Fig. 1, we present how we model the wargaming decision-making problem with an offline RL framework in a hybrid action space.

As shown in the lower block of Fig. 1, we use the proposed method CHAIR to offline train an RL-driven wargaming decision-making model, that can be deployed into a realistic wargaming environment, merely based on the early-prepared wargaming offline dataset. The upper block of Fig. 1 presents the online wargaming MDPs with existing decision-making policies to prepare historical transitions for building the needed offline dataset. For our problem formulation, the wargaming MDP is abstracted as follows.

Agent. The agent in a wargaming MDP represents the wargaming decision-making model. Its policy outputting sequential COAs directs the collaborative wargaming entities towards fulfilling a particular mission in the wargaming scenario.

Environment. The environment in a wargaming MDP pertains to the simulated wargaming environment established for a specific wargaming scenario. It interacts continuously with the agent by responding to the agent's action and providing the latest wargaming situation data and reward signal.

State. The state in a wargaming MDP is a continuous vector that is abstracted from the raw wargaming situation data. Each element of the vector is normalized and indicates an environmental feature that the agent observes at the current time step.

Action. The action in a wargaming MDP is a hybrid action that corresponds to a specific COA generated by the decision-making model, as shown in Fig. 1. Without loss of generality in wargaming scenarios, this paper mainly considers two typical discrete actions: "attack" (to attack the targets in some area) and "move" (to move to some area for patrolling). The action "move" has four associated continuous parameters: the x and y coordinates of the area center, as well as the width and length of the area. Besides

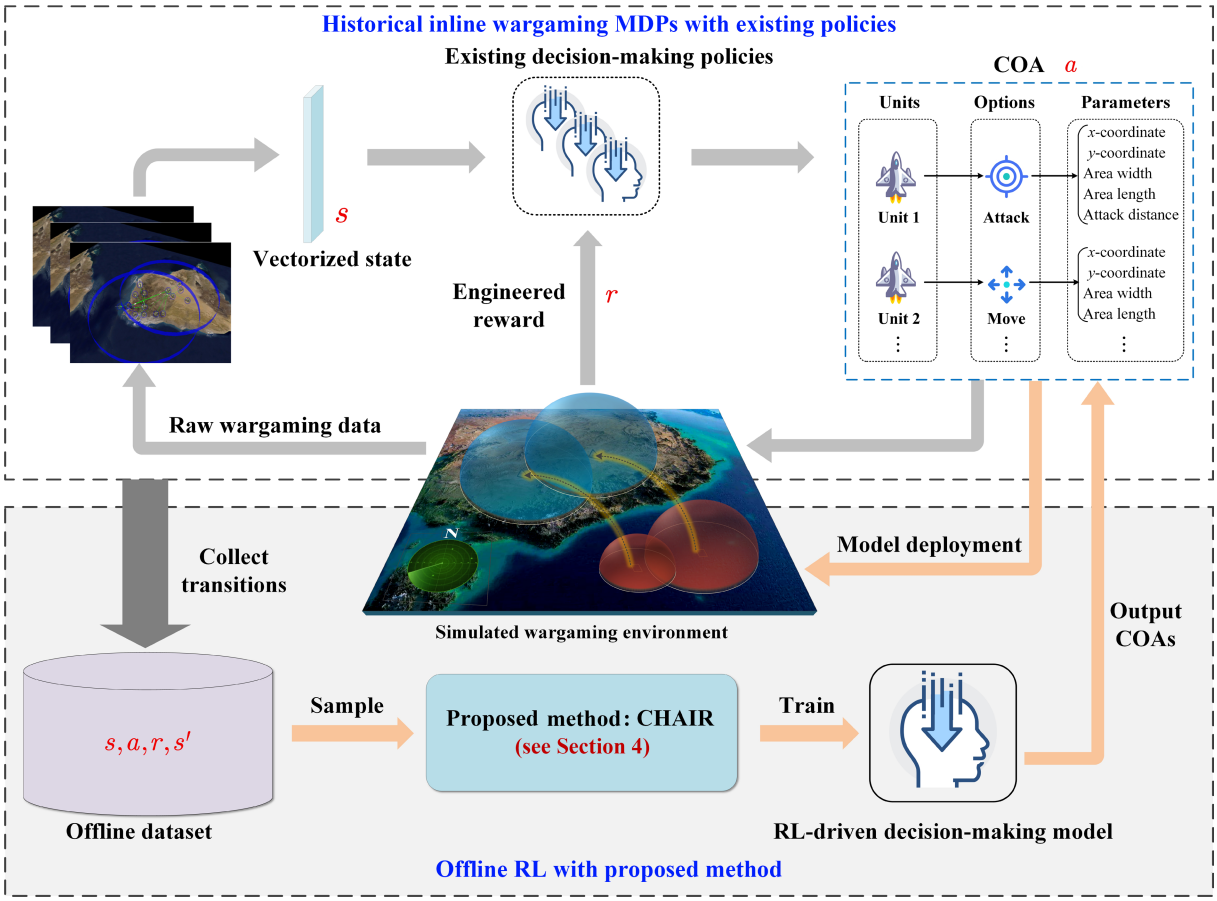


Fig. 1 Modeling framework of the wargaming decision-making problem.

these four parameters, the action "attack" is accompanied by another continuous parameter: the attack distance, which defines the farthest distance from which the targets can be attacked.

Reward. The reward in a wargaming MDP is a scalar value given by a task-specific reward function at each time step to drive RL. Generally, the reward function is manually engineered scenario by scenario.

In Section 6.1, we present wargaming scenario cases with more specific modeling details, including the specific construction of vectorized states and hybrid actions, as well as the specific manually designed reward functions.

4 Method

In this section, we expound on CHAIR, depicted in Fig. 2. As Fig. 2 shows, CHAIR is composed of the following two learning stages.

(1) Implicit representation learning

In this stage, we seek a unified and compact representation for the heterogeneous hybrid action space encompassing both discrete and corresponding continuous components. That is, the representation goal is to map the original hybrid actions $a_k = (k, x_k)$ into a latent action $\tilde{a}_k = (e_k, z_{x_k})$ in a latent space, where e_k and z_{x_k} are the latent variables of the discrete and continuous components, respectively. This representation of the hybrid action enables us to transform the offline dataset $\mathcal{D}$ in the original hybrid action space to a dataset $\tilde{\mathcal{D}}$ in a latent continuous action space, thereby helping to apply off-policy RL algorithms applicable to continuous action spaces for addressing offline settings. Furthermore, performing offline RL in the represented latent space demonstrates benefits in terms of better exploration feasibility and action scalability[18].

(2) Constrained offline RL

In this stage, we perform constrained offline RL in the previously obtained laten representation space. With several critical constraints that are adaptively adjusted, the offline RL employs the offline data in latent representation space to learn a latent RL policy, which maps from a state to a latent action. The latent policy is reconstructed into a hybrid policy with the prior implicit representation technique to be deployed in a realistic environment.

4.1 Implicit representation learning

First, we demonstrate that the desired representation should satisfy the following requirements:

(1) The representation should consider the dependence between discrete and continuous components of the original hybrid actions.

(2) The representation should be reversible, that is, we can reconstruct the original hybrid action from its representation, enabling interaction with the environment for further deployment.

(3) The representation should consider the OOD constraints within the support of the offline dataset to be applicable to offline RL settings.

4.1.1 Representation and reconstruction of hybrid action

The workflow of the proposed implicit representation schema is shown in Fig. 3. As Fig. 3 shows, CHAIR introduces the embedding technique[44] and CVAE for the representation and reconstruction of hybrid actions, as shown in Fig. 3. For the representation of the discrete actions, two parameterized components are built: an embedding layer E_{ζ_1} and an extracting layer E_{ζ_2}. For the representation of the continuous parameters, we use a CVAE model composed of an encoder q_ϕ with parameter ϕ and a decoder p_ψ

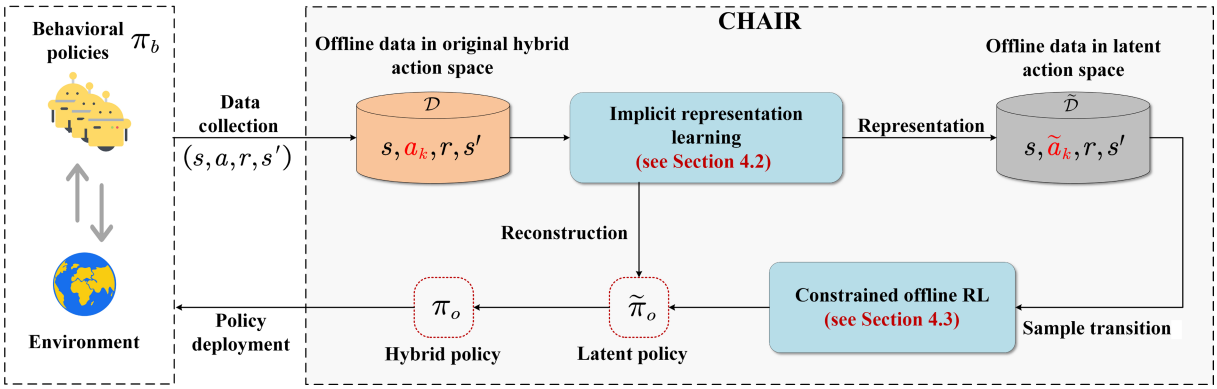


Fig. 2 Overall framework of CHAIR.

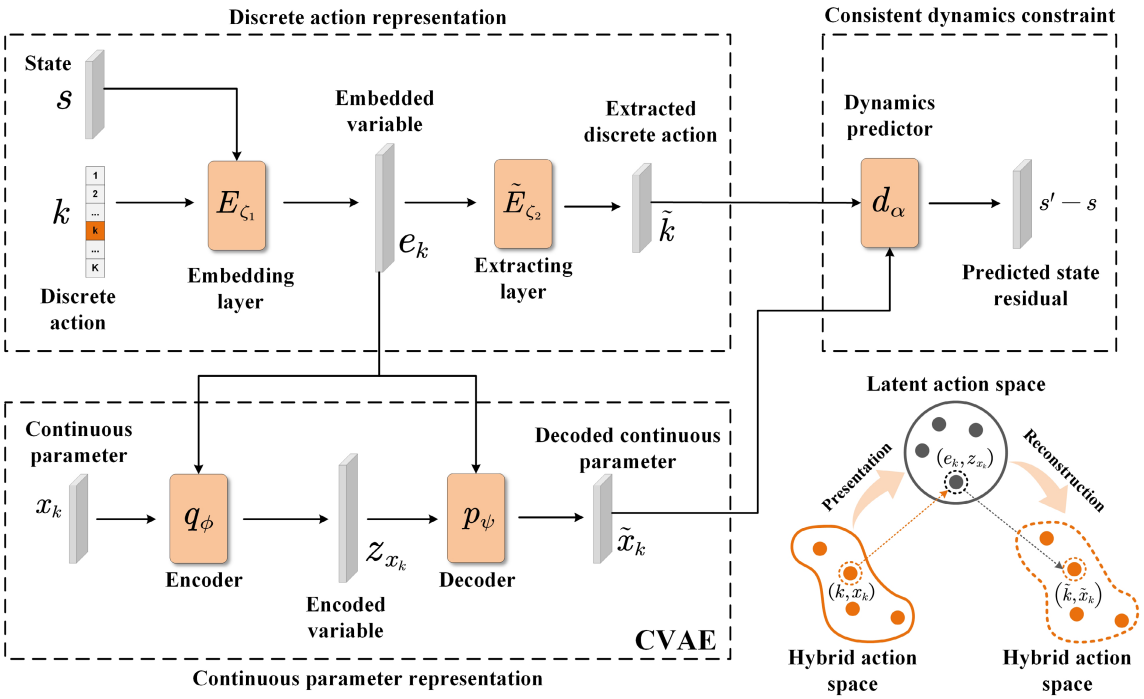


Fig. 3 Workflow of the implicit representation schema.

parameterized by ψ.

These components function in the following way. The embedding layer and the encoder are designed to construct the latent representation of the hybrid actions. Specifically, the embedding layer E_{ζ_1} takes the state s as a condition and maps the original discrete action k into a latent embedded variable $e_k = E_{\zeta_1}(k, s) \in \mathbf{R}^{d_1}$. Then, the encoder q_ϕ takes the embedded variable e_k as a condition, and maps the continuous parameter x_k into a latent encoded variable $z_{x_k} = q_\phi(x_k, e_k) \in \mathbf{R}^{d_2}$. Note that this mapping models the implicit dependence between discrete actions and continuous parameters, which conforms to the natural decision-making process, satisfying the abovementioned Requirement (1). The extracting layer and the decoder execute the process of reconstructing the hybrid action from its latent representation. In specific, the decoder p_ψ takes the same condition as the encoder, and obtains the original continuous parameter's reconstruction $\tilde{x}_k = p_\psi(z_{x_k}, e_k)$, from the latent encoded variable z_{x_k}. The extracting layer E_{ζ_2} reconstructs the discrete action $\tilde{k} = E_{\zeta_2}(e_k)$ from the latent embedding variable e_k. This reconstruction helps satisfy the abovementioned Requirement (2).

Optimizing CVAE. The goal of the CVAE model is to generate action samples that come from the same action distribution as the original dataset. To achieve this, the encoder $q_\phi(x_k, e_k)$ outputs the mean μ and standard deviation σ of a Gaussian distribution $\mathcal{N}(\mu, \sigma)$. The latent encoded variable z_{x_k} is sampled from this Gaussian and passed into the decoder $p_\psi(z_{x_k}, e_k)$ to reconstruct the original action. The CVAE model is trained to maximize the variational lower bound[39], namely, to minimize the loss $\mathcal{L}_{\text{VAE}}$,

$$\mathcal{L}_{\text{VAE}} = E_{\mathcal{D}}\left[\|\tilde{x}_k - x_k\|^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) \,\|\, \mathcal{N}(0, 1))\right] \quad (4)$$

where the first term is the reconstruction loss with the Mean Square Error (MSE), and the second term is a regularization loss with the KL-divergence between the distribution of the latent encoded variable and its standard Gaussian prior. Minimizing this loss function can drive the extracting layer to reconstruct the original discrete actions from the latent embedded vectors as faithfully as possible. Meanwhile, this minimization constrains the reconstructed discrete actions to be within the support range of the offline data, which is crucial for offline RL.

Optimizing extracting layer. The goal of the extracting layer is to reconstruct the original discrete action from the latent embedded variable e_k. In this respect, the distribution of the reconstructed discrete actions should be close to the original distribution from the offline dataset. Thus, the extracting layer is trained to minimize the loss $\mathcal{L}_{\text{EXT}}$,

$$\mathcal{L}_{\mathrm{EXT}} = E_{\mathcal{D}}\left[H\left(\mathrm{softmax}\left(E_{\zeta_2}(e_k)\right), \mathrm{onehot}(k)\right)\right] \quad (5)$$

where $H(\,)$ is the Cross Entropy (CE) of two categorical distributions, the $\mathrm{softmax}(\,)$ operator converts the output of the extracting layer into a categorical distribution, and the $\mathrm{onehot}(\,)$ operator converts the index of the discrete action into a one-hot vector. By minimizing the above losses, our schema implicitly models the state-action distribution in the offline dataset, as well as the dependencies between discrete and continuous components. This minimization eventually constrains the action representation and reconstruction to be within the support of the offline dataset, which helps meet the abovementioned Requirement (3).

4.1.2 Representation constraint with consistent dynamics

In the preceding discussion, we present our schema for constructing a unified and compact latent representation space for hybrid actions. However, the obtained representation space may suffer from pathologies that makes it unable to discriminate between the different impacts of hybrid actions on the environment. That is, it is uninvolved in the consistent impact of both original and reconstructed hybrid actions on the environmental dynamics. Consequently, such a representation may be ineffective when applied to learning RL in an MDP framework, which relies heavily on knowledge of environmental dynamics. These intuitions were demonstrated in other previous studies[45, 46].

Motivated by these intuitions, a semantically smoother latent representation space is suggested, where the latent actions in colse proximity correspond to the original hybrid actions with similar environmental impacts[25]. To this end, we propose a consistent dynamics constraint to refine the representation of hybrid actions, which helps to better meet the abovementioned Requirement (3).

In specific, we build a dynamics predictor d_α parameterized by α to explicitly model the impact of reconstructed hybrid action (the discrete action and continuous parameters jointly function) on the state transitions. It takes reconstructed hybrid action $\left(\tilde{k}, \tilde{x}_k\right)$ and outputs the prediction of the environmental dynamics driven by the reconstruction, as shown in Fig. 3. In principle, the prediction should be consistent with the real environmental dynamics. That is, we optimize the dynamics predictor by minimizing the loss $\mathcal{L}_{\mathrm{DYN}}$,

$$\mathcal{L}_{\mathrm{DYN}} = E_{\mathcal{D}}\left[\left\|d_\alpha\left(\tilde{k}, \tilde{x}_k\right) - \Delta s\right\|^2\right] \quad (6)$$

where $d_\alpha\left(\tilde{k}, \tilde{x}_k\right)$ is the output of the dynamics predictor, and the state residual $\Delta s = s' - s$ represents the real environmental dynamics. This unsupervised loss of the dynamics predictor acts as an additional crucial constraint to force the latent representation to be semantically smoother. This property is beneficial for RL to learn the knowledge environment. Intuitively, this constraint also makes the latent representation more consistent with the offline data distribution, which will facilitate the subsequent offline RL.

In this way, we obtain the complete loss function $\mathcal{L}_{\mathrm{IR}}$ for the implicit representation learning stage,

$$\mathcal{L}_{\mathrm{IR}} = \mathcal{L}_{\mathrm{VAE}} + \mathcal{L}_{\mathrm{EXT}} + \mathcal{L}_{\mathrm{DYN}} \quad (7)$$

As shown in the lower right of Fig. 3, training with this complete loss function ensures that our implicit representation schema satisfies the three requirements stated earlier. As a result, the hybrid action space can be mutually mapped with the latent representation space. By incorporating several constraints that we have added, a hybrid action space that closely approximates the original one can be reconstructed, where the reconstructed actions are distributed as much as possible within the support of the offline dataset.

4.2 Constrained offline reinforcement learning

In the preceding stage, we obtain an implicit representation of the hybrid action space, whereby we map original hybrid actions into latent actions. Meanwhile, a constrained hybrid action space can be reconstructed from the latent action space, which helps us to execute offline RL. In this stage, we propose to perform offline RL in the latent action space rather than in the original hybrid action space. In specific, the offline RL learns a latent RL policy which maps from a state to a latent action. When the offline RL is finished, the learned latent policy can be transformed into a realistic hybrid policy through the reconstruction components in deployment in the real environment. This helps to transform the problem of hybrid control into continuous control, thereby incorporating off-policy RL algorithms for continuous control.

4.2.1 Offline TD3 with latent action constraints

In this paper, we use TD3[19, 43]. To further ensure the offline RL effectively learns a latent policy, we

develop two additional techniques to constrain the OOD action shift. First, we add a Behavior Cloning (BC) loss term to penalize the latent action shift out of the latent representation space. Second, we use a variant of Clipped Double Q-learning (CDQ) to penalize uncertainty over the future estimate to constrain the Q-function overestimation bias. These two constraints are inspired by previous works[17, 19], but we make a critical improvement by granting them dynamic weights to adaptively control the constraint degree. Below, we present the details.

Offline TD3 framework. TD3 parameterizes an actor network that models a latent policy $\pi_\omega(s)=(e,\ z)$. Besides, double critic networks are parameterized to estimate the latent action value Q^{π_ω}, that is, $Q_{\theta_{i=1,\,2}}(s,\ \pi_\omega(s))$. These three networks have their corresponding target networks: $\pi_{\bar{\omega}}$ and $Q_{\bar{\theta}_{i=1,\,2}}$ to stabilize learning. Regularly, the parameters of the target networks are softly updated to the current network parameters,

$$\begin{cases}\bar{\omega}\leftarrow\tau\omega+(1-\tau)\bar{\omega},\\ \bar{\theta}_i\leftarrow\tau\theta_i+(1-\tau)\bar{\theta}_i,\ \ i=1,\ 2\end{cases}\tag{8}$$

Offline TD3 learning. In the above AC framework, with an offline dataset $\widetilde{\mathcal{D}}=\{(s,\ \tilde{a}_k,\ r,\ s')\}$, where the original hybrid actions $a_k=(k,\ x_k)$ are represented by latent actions $\tilde{a}_k=(e_k,\ z_{x_k})$, the latent policy $\pi_\omega(s)$ is optimized,

$$\pi_\omega(s)\leftarrow\underset{\pi}{\operatorname{argmax}}\ E_{\mathcal{D}}[\lambda\mathcal{J}_Q-\beta\mathcal{L}_{\mathrm{BC}}]\tag{9}$$

where $\mathcal{J}_Q$ is the objective term of maximizing the overall action value corresponding to Eq. (2), and $\mathcal{L}_{\mathrm{BC}}$ is the abovementioned additional BC loss term. λ and β are their dynamic weights of them (detailed in the following Section 4.2.2). $\mathcal{L}_{\mathrm{BC}}$ is the MSE between the latent policy's output and the latent actions supported in the offline dataset,

$$\mathcal{L}_{\mathrm{BC}}=\|\pi_\omega(s)-\tilde{a}_k\|^2=\|(e,\ z)-(e_k,\ z_k)\|^2\tag{10}$$

The objective term $\mathcal{J}_Q$ is optimized by DPG ascent propagating through both actor and critic. Concretely, its gradient $\nabla_\omega\mathcal{J}_Q$ is induced as follows:

$$\nabla_\omega\mathcal{J}_Q=E_{\mathcal{D}}\left[\nabla_\omega\pi_\omega(s)\nabla_{\tilde{a}}Q_{\theta_1}(s,\ \tilde{a})\,|\,\tilde{a}=\pi_\omega(s)\right]\tag{11}$$

where the critics are updated by minimizing the abovementioned CDQ loss $\mathcal{L}_{\mathrm{CDQ}}$,

$$\mathcal{L}_{\mathrm{CDQ}}=E_{\mathcal{D}}\left[\left(r_t+\gamma T_{\mathrm{CDQ}}-Q_{\theta_i}(s,\pi_\omega(s))\right)^2\right],\quad i=1,\ 2\tag{12}$$

Critically, a variant of CDQ target T_{CDQ} is adopted here for TD updating, given by a minimum-maximum convex combination,

$$\begin{aligned}\mathcal{L}_{\mathrm{CDQ}}=&\beta\min_{i=1,\,2}Q_{\bar{\theta}_i}(s',\ \pi_{\bar{\omega}}(s'))+\\&(1-\beta)\max_{i=1,\,2}Q_{\bar{\theta}_i}(s',\ \pi_{\bar{\omega}}(s'))\end{aligned}\tag{13}$$

where the minimum term penalizes the future estimates in uncertain regions and encourages the latent policy to prioritize actions that lead to states within the support. This minimum serves to reduce the overestimation bias and high variance in TD learning. In contrast the maximum term reflects the level of greediness in the Q-function update. Here, we reuse the dynamic weight $\beta\in[0,1]$ to control the penalty strength on the uncertainty of future estimates. If $\beta=1$, the update corresponds to the original CDQ that performs the most conservative Q-function updating.

4.2.2 Adaptive weight adjustment

In this section, we present an adaptive weight adjustment technique to dynamically control the components of the offline RL optimization objective. That is, we adjust the following two weights: λ and β. The primary objective of designing an adaptive weight adjustment mechanism is to strike some balances. Regarding λ, we seek a balance between the scales of two optimization objective terms in Eq. (9) for better training stability. Regarding β, we seek a balance between the constraints on OOD actions and the generalization of exploration feasibility. To be specific, on the one hand, offline RL requires constraints on the OOD action shift. However, on the other hand, these constraints should not be overly stringent, as it may restrict the exploration feasibility leading to the degradation of policy generalization. This outcome may result in a significant loss in overall policy performance improvement.

As shown in Eq. (9), the policy optimization objective is to maximize $\mathcal{J}_Q$ and minimize $\mathcal{L}_{\mathrm{BC}}$. These two terms are needed to obtain balance because their scales are often different. Therefore, we dynamically adjust the weight λ to normalize the scale of $\mathcal{J}_Q$ to be aligned with the scale of $\mathcal{L}_{\mathrm{BC}}$,

$$\lambda=\frac{1}{N}\sum_{\{(s,\,\tilde{a}_k)\}_1^N}\frac{\|\tilde{a}_k\|}{|Q_{\theta_1}(s,\ \tilde{a}_k)|}\tag{14}$$

where N is the offline dataset size. In practice, this mean term is estimated over mini-batches, rather than the entire dataset. This is similar to the normalization

trick used in a previous work[19], but we directly compare the norms of the action and its value estimate, avoiding the introduction of extra hyper-parameters.

As stated earlier, the weight β serves two constraint-adjustment purposes: controlling the BC term to penalize the action shift out of the latent representation space, and controlling the CDQ target to penalize the uncertainty over the future estimates. Intuitively, these two adjustments are consistent. Specifically, in the early stage of learning, a strong penalty should be imposed to constrain the policy within the support of the offline dataset. As the policy is optimized and stabilized, the constraints can be relaxed appropriately to increase the flexibility to select actions, improving the policy's OOD generalization moderately[18]. To this end, β is linearly annealed from 1 to its final value β_{F},

$$\beta \leftarrow \beta - \frac{1-\beta_{\mathrm{F}}}{T} \tag{15}$$

where T is the total training steps. Hereby, the dynamical β somewhat balances the alleviation and generalization of OOD actions during offline learning. It is worth noting that the constraints should not be overly relaxed to violate the constraint premise of offline RL, so the final value should not be too small. In practice, we use $\beta_{\mathrm{F}} = 0.85$. Besides stabilizing and improving policy learning, the above adaptive constraint adjustment technique helps our method to avoid additional hyper-parameter fine-tuning.

4.3 Complete algorithm

We outline the proposed CHAIR in the pseudocode provided in Algorithm 1.

Algorithm 1 CHAIR

Input: Offline dataset $\mathcal{D}=\{(s_i, a_i, r_i, s_i')\}_{i=1}^{N}$;
maximum representation learning steps M;
maximum offline RL steps T

// Implicit representation learning

for i=1 to M **do**
  Sample minibatches from $\mathcal{D}$;
  Optimize ζ_1, ζ_2, φ, and ψ with Eq. (7);

// Constrained offline RL

Initial the latent policy network π_ω, two critic networks, Q_{θ_1} and Q_{θ_2}, and their target networks, $\pi_{\bar{\omega}}$, $Q_{\bar{\theta}_1}$, and $Q_{\bar{\theta}_2}$;

for i=1 to T **do**
  Sample a minibatch of B transitions $\{(s_i, a_i, r_i, s_i')\}_{i=1}^{B}$ from $\mathcal{D}$;
  Transform the original actionin each transition into the latent space: $a_i \rightarrow \tilde{a}_i$, using the learned representation components;
  Optimize the latent policy with Eq. (9);

5 Effectiveness evaluation

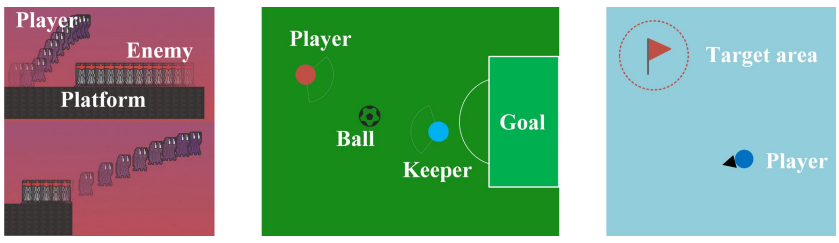
In this section, we aim to evaluate the preliminary effectiveness of our proposed approach. Subsequently, we provide the detailed experimental settings and results.

5.1 Experimental Environments

We built three toy game environments which are used as typical benchmarks in previous works[25–27, 30] towards hybrid actions, visualized in Fig. 4.

Platform. In this environment, the agent has three discrete actions: "run", "hop", and "leap", and each is accompanied by a continuous parameter to determine the horizontal displacement. The agent's goal is to successfully reach the goal platform by hopping over enemies and leaping across gaps between platforms. A six-dimensional state space describes the position and velocity of the agent and local enemy, as well as the lengths of the current and next platform. The reward for a step is calculated as the distance covered in this step divided by the total route length, adding a penalty of –0.5 if this step fails.

Robot soccer. In this environment, the agent aims to score a goal against a keeper who tries to intercept the ball. There are two discrete actions: "kick" and "move", each with two parameters that define the target position's coordinates. A 14-dimensional state space gives features, including the position, velocity,



(a) (b) (c)

Fig. 4 Three toy game environments, (a) platform, (b) robot soccer, and (c) target search.

and orientation of the player, keeper, and ball. The reward for a step is 0 for intermediate steps, 50 for a terminal goal state, and $-c$ for a terminal non-goal state, where c is the distance between the ball and the goal.

Target search. In this environment, the agent tries to move towards a target circle area and stop in it. The agent has three hybrid actions: "turn" with a parameter that defines the rotation value, "accelerate" with a parameter that defines the acceleration power, and "break" without parameters. The 10-dimensional state space describes features, such as the player's position, speed, direction, and distance relative to the target, as well as an indicator that becomes 1 if the player is inside the target zone. The reward for a step is the distance of the player from the target of the last step minus the current distance.

5.2 Experimental settings

Offline dataset. Considering the availability of public implementations and the stability of performance, we select four existing RL algorithms (PADDPG, PDQN, HPPO, and HyAR) crafted for hybrid actions as behavioral policies to prepare our offline dataset. For each environment, beforehand, we online run them separately to collect their transitions for the offline dataset construction. A total of 10^5 transitions from each algorithm are collected during the online interaction with the environment, resulting in an offline dataset composed of 4×10^5 transitions. The collection begins after the first task is successful to adequately capture diverse transitions that cover successful experiences. The adoption of four distinct behavioral policies for data collection is designed to emulate the practical diversity of offline data sources encountered in real-world tasks.

Method comparison. As far as our knowledge extends, currently no other offline RL algorithms exist targeting hybrid actions. Hence, to compare against our proposed method CHAIR, we construct two strong baselines based on typical existing typical algorithms tailored for hybrid actions: (1) offline-HyAR that directly applies HyAR to the offline setting, and (2) offline-PDQN that directly applies PDQN to the offline setting. The main rationale behind choosing HyAR and PDQN as baselines is their off-policy nature, which inherently enables the use of data from other behavioral policies, making them suitable for adaptation under offline settings. Due to the inability to utilize off-policy data[26], the offline version of PADDPG is not considered as our baseline. Similarly, HPPO is also excluded as it is based on on-policy PPO. Our proposed method is trained and compared with two baselines using the same prepared offline dataset.

Algorithm setup. To ensure comparison fairness, the shared hypermeters are set as identical for each algorithm in three environments. The algorithm setups are detailed as follows. All networks are structured with fully connected layers and optimized using Adam[47]. Both the encoder and the decoder of CVAE have two hidden layers (500, 500). Both the actor and the critic have two hidden layers (300, 300). The embedding layer and the extracting layer of CHAIR have 300 units each. The dynamics predictor of CHAIR has two hidden layers (400, 400). The learning rates for the actor, the critic, and the CVAE are set as 10^{-4}, 10^{-3}, and 10^{-4}, respectively. The target network's update rate is set to $\tau=0.005$. The discount factor is $\gamma=0.99$. Each algorithm trains the offline RL policy with the same mini-batch size of 128 and the same training iterations of 10^6 time steps. For CHAIR and offline-HyAR, the representation components are trained in 10^6 time steps before the offline RL stage, and the latent action dimension (discrete and continuous latent action) is set as $d_1=4, d_2=4$. The continuous parameters are padded to the same length aligned with their maximum dimension. The representation components are fixed when training the offline RL. The vectorized states are normalized over mini-batches during the training of each algorithm to make it has proven well-suited for offline settings[19]. All hyperparameters involved were set heuristically and optimized through a coarse grid search.

Performance evaluation. During the offline training, the policy is evaluated every 10^4 training time steps over 10 online episodes, and the Average Episode Return (AER, the average return of these 10 test episodes) is adopted as the performance criterion. For each environment and each algorithm, the experiment was run over five random seeds and the results are averaged across them.

5.3 Experimental results

Figure 5 shows the performance comparison results for our method and the other two baselines in three toy game tasks.

From Fig. 5, we can see that our method significantly outperforms the two baselines in all three environments, and our method exhibits better stability

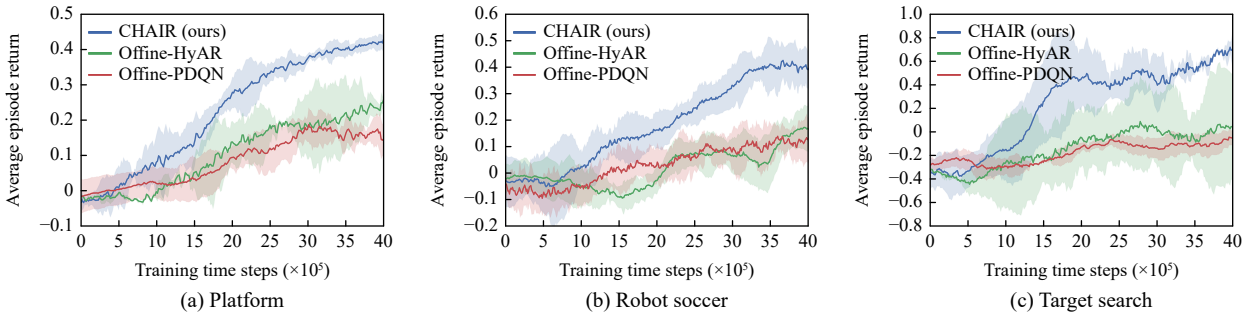


Fig. 5 Performance comparison for three algorithms in three toy game environments.

than the other two baselines across the training process. Moreover, our method shows good generality across different environments with steady policy-enhancing ability. In contrast, although the other two baselines achieve reasonable performance improvement in platform and robot soccer, they almost fail in target search with a very slight improvement. From the view of the final performance improvement, offline-HyAR is superior in two baselines (slightly surpassing offline-PDQN in all three environments), while our method achieves performance improvements that are about two times, four times and five times higher than offline-HyAR in the three environments, respectively.

These comparison results demonstrate that our method CHAIR effectively improves the offline RL agent's policy performance using the fixed offline dataset with good generality across tasks.

6 Case Study

In this section, we investigate the applicability and generalization of our proposed approach in realistic wargaming scenarios. To achieve this, we create two representative wargaming scenarios and conduct a comprehensive case study on each scenario. Below, we will provide details.

6.1 Scenario construction

Our case study is based on a pre-existing wargaming simulation system that encompasses a range of representative simulated wargaming entities, as well as a scheduling engine to facilitate their interactions. Leveraging this system, we can construct diverse wargaming scenarios that involve a variety of missions, areas, and force compositions, etc. Through abstracting and simplifying the properties of typical realistic wargaming scenarios, we first build a simulated wargaming ground called Wargaming Confrontation Ground (WCG) involving typical wargaming environments and platforms, illustrated in Fig. 6. In WCG, the blue and red sides with their respective force units, can engage in customized scenarios where three types of platforms (bombers, fighters, and ships) are involved in confrontations following specific rules. We depict basic confrontation rules as follows.

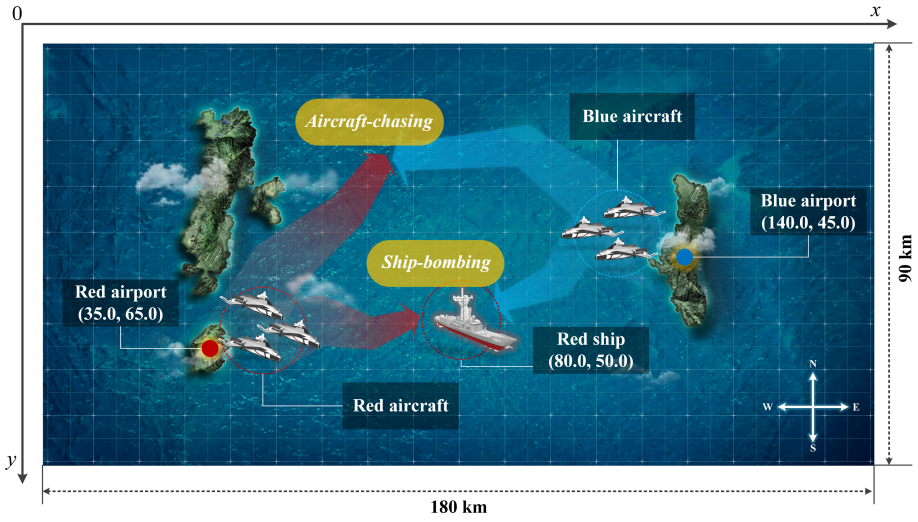


Fig. 6 Illustration of wargaming confrontation ground.

Confrontation rules. A force unit, directly commanded by the decision-making model, can be composed of a designated number of platforms that act in accordance with unified orders. Aircraft (including bombers and fighters) take off from their respective airports. The red ship is initially located at the near center of the ground and moves within a defined area to defend air space during confrontations. Each fighter is equipped with six air-to-air missiles solely for engaging aerial targets. Each bomber armed with two air-to-land missiles, is limited to bombing land or sea targets and is unable to retaliate in the air. The red ship has 15 sea-to-air missiles for attacking air targets. An aircraft is considered destroyed upon being hit once. The ship is regarded as sunk if it is hit twice. Notably, there exists a somewhat negative correlation between hitting accuracy and attack distance, necessitating a trade-off between swift attacks and precision. For simplicity, both sides have access to global environmental information throughout the confrontation.

As Fig. 6 shows, to maintain both generality and typicality, we constructed two wargaming scenarios with different difficulties based on WCG: aircraft-chasing and ship-bombing to perform this case study, detailed as follows.

6.1.1 Aircraft-chasing

Mission. As shown in Fig. 6, the objective of the aircraft-chasing scenario is for the blue side to deploy its fighters to pursue and attack the red side's bombers. The mission entails shooting down all the red side's bombers.

Two sides. The blue side controls one force unit comprising five fighters, while the red side has one force unit comprising five bombers. The decision-making model of the blue side is driven by an RL algorithm, whereas that of the red side is driven by a random algorithm.

Hybrid actions. For the blue side's RL agent, each force unit has two discrete action options: attack or move, and each discrete action comes with its associated continuous parameters, as stated in Section 3.2.2. When the attack action is activated, each aircraft of the unit searches for accessible targets within the area and attack range determined by the continuous parameters, subsequently launches missiles to attack the closest target or taking no action if no accessible targets are found. When the move action is activated, each aircraft in the unit collectively move to patrol the target area defined by the continuous parameters, following a unified motion pattern while maintaining a specific distance around the area center.

Vectorized state. A five-dimensional vector is employed to describe the state of each aircraft. The first three elements, normalized to the range of [0, 1], represent the aircraft's x-axis coordinate, y-axis coordinate, and ammunition consumption, respectively. The fourth element is a binary value indicating whether the aircraft is a bomber or a fighter (1 for bomber, 0 for fighter), while the fifth element is a binary value indicating whether the aircraft is alive (1 for alive, 0 for destroyed). Consequently, the global state can be represented as a 50-dimensional concatenated vector encompassing the information of all 10 aircraft.

Engineered reward. To drive the blue side's RL agent, a task-specific reward function is engineered considering the following factors: the bonus for approaching the red bombers, the bonus for shooting down the red bombers, and the penalty for ammunition waste. The final formulation of the engineered reward function for this scenario is given as follows:

$$r_t = w_1 \frac{N_d}{5} + w_2 e^{-\delta} - w_3 \frac{N_{aa}}{30} \tag{16}$$

where N_d represents the number of bombers destroyed at this time step, N_{aa} is the number of air-to-air missiles launched at this time step, and δ represents the distances between the two units (a unit's position is determined by the average coordinates of its existing aircraft). The weights w_1, w_2, and w_3 are empirically set as $w_1 = 0.50$, $w_2 = 0.75$, and $w_3 = 0.10$ after coarse fine-tuning.

Termination conditions. An episode ends with success when all red bombers are shot down. Conversely, it terminates in failure for either of the following conditions: (1) All the red fighters have depleted their ammunition, or (2) the episode lasts for over 200 time steps.

6.1.2 Ship-bombing

Mission. As shown in Fig. 6, in this ship-bombing scenario, the blue side is tasked to bomb the red side's ship, while the red side defends the ship to be safe. The mission is to successfully sink the red ship by bombing it with bombers.

Two sides. The blue side has three force units: two comprise five bombers each and one comprises 10 fighters (to interfere with the red ship and fighters

attacking the blue bombers). The red side has one force unit comprising six fighters and a ship. The decision-making model of the blue side is driven by an RL algorithm, whereas that of the red side is driven by a rule-based strategy.

Hybrid actions. The same hybrid action of a unit in the aircraft-chasing scenario is followed in this scenario, but the total number of discrete action options becomes $8 = 2^3$ that corresponds to all discrete combinations of three blue side's units.

Vectorized state. The same vectorized state setting of an aircraft in aircraft-chasing scenario is followed in this scenario. Besides, we use another four-dimensional vector to describe the state of the ship, where the elements are the normalized x-axis coordinate, y-axis coordinate, ammunition consumption, and a binary value indicating whether the ship has been hit once (equals to 0 if never hit, 1 if hit once). Therefore, the global state is denoted by a 134-dimensional concatenated vector encompassing the information of 20 blue side's aircraft, six red side's aircraft and a ship.

Engineered reward. Likewise, we craft the reward function for this scenario considering the following factors: the bonus for approaching the red ship, the bonus for hitting the red ship, the penalty for aircraft loss, and the penalty for ammunition waste. Finally, the task-specific engineered reward function is

$$r_t = w_1 I_s + w_2 \mathrm{e}^{-\delta_{\mathrm{SB}}} - w_3 \frac{N_{al}}{20} - w_4 \frac{N_b}{10} - w_5 \frac{N_f}{5} \tag{17}$$

where I_s denotes if the red ship is hit at this time step (if hit, it equals 1, otherwise, 0), N_{al} is the number of the air-to-land missiles launched by blue bombers at this time step, N_b and N_f are respectively the loss numbers of blue bombers and blue fighters at this time step, δ_{SB} represents the distance between the red ship and bomber unit that has remaining ammunition and is closer to the ship (the unit's position is measured by the average coordinates of its existing bombers). Here, the weights of each part are empirically set as $w_1 = 0.80$, $w_2 = 0.40$, $w_3 = 0.15$, $w_4 = 0.10$, and $w_5 = 0.05$ after coarse fine-tuning.

Termination conditions. The episode terminates with success when the red ship is bombed to be sunk, while it terminates with failure when either of the following events happens: (1) all blue bombers are shot down, (2) all blue bombers have run out of ammunition, and (3) the episode lasts over 300 time steps.

6.2 Experimental settings

In this case study, we basically retain the experimental settings as those in Section 5.2, and customize some settings according to the features of wargaming scenarios, as follows.

Offline dataset. The offline dataset becomes larger with 6×10^5 transitions that are collected from four algorithms with 150k transitions each.

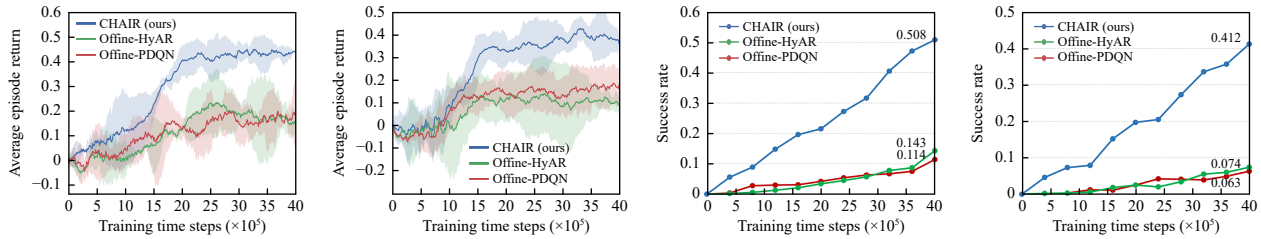
Algorithm setup. Likewise, we use the same algorithm setups in two wargaming scenarios. Differently, we make the following modifications to the setups of three algorithms to adapt the higher-dimension information in wargaming scenarios: The actor and the critic increase the hidden layer size to (500, 500). The hidden layer size in the encoder and the decoder becomes (800, 800), and that of the dynamics predictor becomes (600, 600). The widths of the embedding layer and the extracting layer become 500. The latent action dimension is set as $d_1 = 6$ and $d_2 = 6$. Each algorithm trains the RL policy training iterations 4.0×10^6 time steps. The training timesteps of representation learning become 2.0×10^6, and the offline RL training time steps remain at 4.0×10^6.

Performance evaluation. Considering the domain feature of wargaming tasks, we adopt several additional criteria besides AER to evaluate the policy performance: (1) Success Rate (SR, regularly evaluated 10 times over 100 online episodes during the whole policy training stage); (2) Force Cost Rate (FCR, evaluated when the training is finished over 100 online episodes, denoting the average of the force unit death rate over all successful episodes); and (3) Ammunition Cost Rate (ACR, evaluated when the training is finished over 100 online episodes, the average of ammunition consumption rate over all successful episodes). FCR and ACR indicate the wargaming losses of the decision-making model, and the lower these two criteria, the more effective the decision-making model.

6.3 Experimental results

We show the comparison results for our method and the other two baselines in two typical wargaming scenarios in Fig. 7 and Table 1.

From the Figs. 7a and 7b, we can obtain similar findings as those from the evaluation results in Section 5.3. Specifically, the AER of our method CHAIR significantly surpasses that of the other two baselines in both wargaming scenarios, and our method shows



(a) Aircraft-chasing (b) Ship-bombing (c) Aircraft-chasing (d) Ship-bombing

Fig. 7 Performance comparison for three algorithms in two wargaming scenarios.

Table 1 Comparison results of FCR and ACR for three algorithms in the two wargaming scenarios. FCR is not applicable in aircraft-chasing scenario. The best results are marked in bold.

Algorithm	Aircraft-chasing		Ship-bombing	
	FCR	ACR	FCR	ACR
CHAIR (ours)	–	**20.5**	**85.4**	**76.2**
Offline-HyAR	–	37.5	94.3	84.9
Offline-PDQN	–	40.2	96.6	86.5

better stability and generality. From the final AER improvement, offline-PDQN is slightly stronger than offline-HyAR in the two wargaming scenarios, but only reaches about 1/4 of CHAIR. Likewise, from the SR evaluation results shown in Figs. 7c and 7d, we can observe the following evidence to show CHAIR's considerable superiority: Notably, CHAIR steadily improves the success rate to 50.8% and 41.2% in aircraft-chasing and more complex ship-bombing, respectively. In contrast, the success rates of the other two baselines can barely exceed 10% (offline-HyAR: 14.3%; offline-PDQN: 11.4%) in aircraft-chasing, and cannot even surpass 10% in more difficult ship-bombing (offline-HyAR: 7.4%; offline-PDQN: 6.3%). The comparison results of the success rate demonstrate that CHAIR's policy has superior capability in solving wargaming tasks.

As can be seen from Table 1, CHAIR has much less wargaming loss (in terms of FCR and ACR) than the other two baselines. Considering that CHAIR has the highest success rate, this comparison of wargaming losses clearly indicates that CHAIR achieves a much more efficient policy than the other two baselines. That is, CHAIR's policy can solve the wargaming tasks at less cost.

To sum up, these the above empirical findings in this case study reveal several facts: (1) Our method CHAIR effectively improves the offline RL agent's policy only using the static offline wargaming dataset with hybrid COA actions. (2) The policy obtained by our method is more efficient, with less wargaming losses and higher success rates. (3) Our method has good generality across typical wargaming scenarios with the its steady policy-enhancing ability and superior performance.

7 Discussion

The reason why our method can demonstrate the superiority may lie in the following aspects: (1) During the implicit representation phase, we explicitly impose constraints on the consistency between discrete actions and state transitions. This imposition ensures better alignment of action space mappings, thereby guaranteeing stability and effectiveness in training within the offline paradigm. (2) In the offline RL phase, the introduction of dynamic OOD action constraints allows for a balanced exploration-exploitation trade-off and mitigates extrapolation errors, resulting in improved policy performance. In contrast, HyAR and PDQN are originally online algorithms that rely on continuous interaction with the environment to update TD estimates. They lack sufficient constraints to ensure successful policy improvement when transferred to the offline paradigm with a fixed dataset.

Future works may include several directions to improve our work, discussed as follows. First, we have not sufficiently investigated the impact of the latent action dimension and offline dataset quality on the learning performance, due to the limited experimental time cost. This issue may need to be further explored in future works based on systematic ablation studies. Second, we share the global information and perform centralized control for each unit in our wargaming scenario, which to some extent does not conform to the natural wargaming settings. Considering the applicability to wargaming scenarios with local observation and more abundant units, it may be a promising line to combine the proposed method with the multi-agent RL paradigm under a partially observed MDP framework. Third, future studies could try to refine the policy acquired through offline RL with

further online interactions in the real environment. Last, it is valuable to enhance the present representation method to wider forms of hybrid action space, for example, multi-level hybrid actions in a hierarchical structure.

8 Conclusion

This paper aims at the wargaming decision-making problem with discrete-continuous hybrid commands under offline settings, which current RL methods fail to address. Towards this domain demand, we propose CHAIR, a novel two-stage offline RL method crafted for hybrid action spaces. Using a new constrained action representation technique, CHAIR first transforms the problem from hybrid control into continuous control, allowing learning of a latent policy in a representation space with better exploration feasibility and action scalability. Then, for better policy robustness and flexibility, CHAIR develops a novel optimization objective with adaptively adjusted constraint components to balance the alleviation and generalization of OOD action under offline settings. Finally, CHAIR learns a latent policy that can be reconstructed back to a hybrid policy. Through the systematic evaluation, we demonstrate the superior performance, stability, and generality of CHAIR across typical toy game tasks and our realistic wargaming scenarios. Futhermore, this work can also provide a preliminary exploration to leverage the abundant domain offline data in the wargaming field to accelerate the RL deployment, rather than training RL-based decision-making models from scratch relying on time-consuming online interactions.

References

[1] R. R. Hill and J. O. Miller, A history of United States military simulation, in *Proc. 2017 Winter Simulation Conf.* (*WSC*), Las Vegas, NV, USA, 2017, pp. 346–364.

[2] J. Appleget, An introduction to wargaming and modeling and simulation, in *Simulation and Wargaming*, C. Turnitsa, C. Blais, and A. Tolk, Eds. Hoboken, NJ, USA: John Wiley & Sons, 2021, pp. 1–22.

[3] S. Wang and Y. Liu, Modeling and simulation of CGF aerial targets for simulation training, in *Proc. Int. Conf. Computer Intelligent Systems and Network Remote Control* (*CISNRC 2020*), doi: 10.12783/dtcse/cisnr2020/35167.

[4] Ö. F. Arar and K. Ayan, A flexible rule-based framework for pilot performance analysis in air combat simulation systems, *Turk. J. Elec. Eng. Comp. Sci.*, vol. 21, no. 8, pp. 2397–2415, 2013.

[5] C. Huang, H. Zhang, L. Wang, X. Luo, and Y. Song, Mixed deep reinforcement learning considering discrete-continuous hybrid action space for smart home energy management, *J. Mod. Power Syst. Clean Energy*, vol. 10, no. 3, pp. 743–754, 2022.

[6] K. Gao, Y. Huang, A. Sadollah, and L. Wang, A review of energy-efficient scheduling in intelligent production systems, *Complex Intell. Syst.*, vol. 6, no. 2, pp. 237–249, 2020.

[7] Y. He, L. Xing, Y. Chen, W. Pedrycz, L. Wang, and G. Wu, A generic Markov decision process model and reinforcement learning method for scheduling agile earth observation satellites, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 3, pp. 1463–1474, 2022.

[8] K. Zhu and T. Zhang, Deep reinforcement learning based mobile robot navigation: A review, *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[9] K. Zhao and L. Ning, Hybrid navigation method for multiple robots facing dynamic obstacles, *Tsinghua Science and Technology*, vol. 27, no. 6, pp. 894–901, 2022.

[10] X. Hao, C. Xu, L. Xie, and H. Li, Optimizing the perceptual quality of time-domain speech enhancement with reinforcement learning, *Tsinghua Science and Technology*, vol. 27, no. 6, pp. 939–947, 2022.

[11] L. Wang, Z. Pan, and J. Wang, A review of reinforcement learning based intelligent optimization for manufacturing scheduling, *Complex System Modeling and Simulation*, vol. 1, no. 4, pp. 257–270, 2021.

[12] M. Tan, Z. Zhang, Y. Ren, I. Richard, and Y. Zhang, Multi-agent system for electric vehicle charging scheduling in parking lots, *Complex System Modeling and Simulation*, vol. 3, no. 2, pp. 129–142, 2023.

[13] Z. Liao and S. Li, Solving nonlinear equations systems with an enhanced reinforcement learning based differential evolution, *Complex System Modeling and Simulation*, vol. 2, no. 1, pp. 78–95, 2022.

[14] W. Shi, Y. H. Feng, G. Q. Cheng, H. L. Huang, J. C. Huang, Z. Liu, and W. He, Research on multi-aircraft cooperative air combat method based on deep reinforcement learning, (in Chinese), *Acta Autom. Sin.*, vol. 47, no. 7, pp. 1610–1623, 2021.

[15] B. Yuksek, U. M. Demirezen, and G. Inalhan, Development of UCAV fleet autonomy by reinforcement learning in a wargame simulation environment, in *Proc. AIAA Scitech 2021 Forum*, doi: 10.2514/6.2021-0175.

[16] Y. Sun, B. Yuan, Q. Xiang, J. Zhou, J. Yu, D. Dai, and X. Zhou, Intelligent decision-making and human language communication based on deep reinforcement learning in a wargame environment, *IEEE Trans. Hum. Mach. Syst.*, vol. 53, pp. 201–214, 2023.

[17] S. Fujimoto, D. Meger, and D. Precup, Off-policy deep reinforcement learning without exploration, in *Proc. 36th Int. Conf. Machine Learning*, Long Beach, CA, USA, 2019, pp. 2052–2062.

[18] W. Zhou, S. Bajracharya, and D. Held, PLAS: Latent action space for offline reinforcement learning, in *Proc. 2020 4th Conf. Robot Learning*, Cambridge, MA, USA,

2021, pp. 1719–1735.

[19] S. Fujimoto and S. Gu, A minimalist approach to offline reinforcement learning, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Virtual Event, 2021, pp. 20132–20145.

[20] A. Kumar, A. Zhou, G. Tucker, and S. Levine, Conservative Q-learning for offline reinforcement learning, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2020, p. 100.

[21] C. Zhao, K. Huang, and C. Yuan, DCE: Offline reinforcement learning with double conservative estimates, in *Proc. 11th Int. Conf. Learning Representations*, doi: 10.48550/arXiv.2209.13132.

[22] Y. Wu, G. Tucker, and O. Nachum, Behavior regularized offline reinforcement learning, arXiv preprint arXiv: 1911.11361, 2019.

[23] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma, MOPO: Model-based offline policy optimization, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2020, p. 1185.

[24] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, MOReL: Model-based offline reinforcement learning, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2020, p. 1830.

[25] B. Li, H. Tang, Y. Zheng, J. Hao, P. Li, Z. Wang, Z. Meng, and L. Wang, HyAR: Addressing discrete-continuous action reinforcement learning via hybrid action representation, in *Proc. the 10th Int. Conf. Learning Representations*, Virtual Event, doi: 10.48550/arXiv.2109.05490.

[26] J. Xiong, Q. Wang, Z. Yang, P. Sun, L. Han, Y. Zheng, H. Fu, T. Zhang, J. Liu, and H. Liu, Parametrized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space, arXiv preprint arXiv: 1810.06394, 2018.

[27] W. Masson, P. Ranchod, and G. Konidaris, Reinforcement learning with parameterized actions, in *Proc. 10th AAAI Conf. Artificial Intelligence*, Phoenix, AZ, USA, 2016, pp. 1934–1940.

[28] C. J. Bester, S. D. James, and G. D. Konidaris, Multi-pass Q-networks for deep reinforcement learning with parameterised action spaces, arXiv preprint arXiv: 1905.04388, 2019.

[29] M. Hausknecht and P. Stone, Deep reinforcement learning in parameterized action space, in *Proc. the 4th Int. Conf. Learning Representations*, San Juan, PR, USA, doi: 10.48550/arXiv.1511.04143.

[30] Z. Fan, R. Su, W. Zhang, and Y. Yu, Hybrid actor-critic reinforcement learning in parameterized action space, in *Proc. 28th Int. Joint Conf. Artificial Intelligence*, Macao, China, 2019, pp. 2279–2285.

[31] X. Lou, Q. Yin, J. Zhang, C. Yu, Z. He, N. Cheng, and K. Huang, Offline reinforcement learning with representations for actions, *Inf. Sci.*, vol. 610, pp. 746–758, 2022.

[32] S. Levine, A. Kumar, G. Tucker, and J. Fu, Offline reinforcement learning: Tutorial, review, and perspectives on open problems, arXiv preprint arXiv: 2005.01643, 2020.

[33] R. Agarwal, D. Schuurmans, and M. Norouzi, An optimistic perspective on offline reinforcement learning, in *Proc. 37th Int. Conf. Machine Learning*, Virtual Event, 2020, pp. 104–114.

[34] Y. Guo, S. Feng, N. Le Roux, E. Chi, H. Lee, and M. Chen, Batch reinforcement learning through continuation method, in *Proc. the 9th Int. Conf. Learning Representations*, Virtual Event, https://openreview.net/forum?id=po-DLlBuAuz, 2021.

[35] P. Swazinna, S. Udluft, D. Hein, and T. Runkler, Comparing model-free and model-based algorithms for offline reinforcement learning, *IFAC-Papers On Line*, vol. 55, no. 15, pp. 19–26, 2022.

[36] D. P. Kingma and M. Welling, Auto-encoding variational Bayes, arXiv preprint arXiv: 1312.6114, 2022.

[37] W. Whitney, R. Agarwal, K. Cho, and A. Gupta, Dynamics-aware embeddings, in *Proc. the 8th Int. Conf. Learning Representations*, Addis Ababa, Ethiopia, doi: 10.48550/arXiv.1908.09357.

[38] C. Huang, K. Dong, H. Huang, S. Tang, and Z. Zhang, Autonomous air combat maneuver decision using Bayesian inference and moving horizon optimization, *J. Syst. Eng. Electron.*, vol. 29, no. 1, pp. 86–97, 2018.

[39] M. Masek, C. P. Lam, L. Benke, L. Kelly, and M. Papasimeon, Discovering emergent agent behaviour with evolutionary finite state machines, in *Proc. 21st Int. Conf. PRIMA 2018*: *Principles and Practice of Multi-Agent Systems*, Tokyo, Japan, 2018, pp. 19–34.

[40] R. S. Sutton and A. G. Barto, *Reinforcement Learning*: *An Introduction*, 2nd ed. Cambridge, MA, USA: The MIT Press, 2018.

[41] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv: 1509.02971, 2019.

[42] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, Deterministic policy gradient algorithms, in *Proc. 31st Int. Conf. Machine Learning*, Beijing, China, 2014, pp. 387–395.

[43] S. Fujimoto, H. van Hoof, and D. Meger, Addressing function approximation error in actor-critic methods, in *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2018, pp. 1587–31596.

[44] C. Guo and F. Berkhahn, Entity embeddings of categorical variables, arXiv preprint arXiv: 1604.06737, 2016.

[45] A. Grosnit, R. Tutunov, A. M. Maraval, R. R. Griffiths, A. I. Cowen-Rivers, L. Yang, L. Zhu, W. Lyu, Z. Chen, J. Wang, et al., High-dimensional Bayesian optimisation with variational autoencoders and deep metric learning, arXiv preprint arXiv: 2106.03609, 2021.

[46] M. Schwarzer, N. Rajkumar, M. Noukhovitch, A. Anand, L. Charlin, R. D. Hjelm, P. Bachman, and A. C. Courville, Pretraining representations for data-efficient reinforcement learning, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Virtual Event, 2021, pp. 12686–12699.

[47] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv: 1412.6980, 2017.

Liwei Dong received the BEng degree in automation from University of Electronic Science and Technology of China in 2017. Since 2017, he is pursuing the PhD degree in navigation, control, and guidance at School of Automation Science and Electrical Engineering, Beihang University, China. His current research interests include system simulation and modeling, reinforcement learning, and intelligent decision modeling.

Guanghong Gong received the BEng, MEng, and PhD degrees from Beihang University, China in 1990, 1993, and 1997, respectively. She is currently a professor at School of Automation Science and Electrical Engineering, and Key Laboratory of Advanced Simulation Aeronautical Technology, Beihang University, China. She won the Science and Technology Progress Award of the Ministry of Education and the National Defense Science and Technology Progress Award for many times. Her research interests include virtual reality, artificial intelligence, and distributed interactive simulation.

Ni Li received the PhD degree in navigation, guidance, and control from Beihang University, China in 2006. She is currently a professor at School of Automation Science and Electrical Engineering, and Key Laboratory of Advanced Simulation Aeronautical Technology, Beihang University, China. Her main research interests include system modeling and simulation, digital prototyping, virtual reality, and intelligent modeling.

Xin Lin received the MEng and PhD degrees from Beihang University, China in 1996 and 2001, respectively. He is currently a lecturer at School of Automation Science and Electrical Engineering, Beihang University, China. His research interests include system modeling and simulation.