IM3HRL: Model-assisted Intrinsically Motivated Modular Hierarchical Reinforcement Learning

Wei Liu

liuwei@lntu.edu.cn

Liaoning Technical University

Jiaxiang Wang

Liaoning Technical University

Guangwei Liu

Liaoning Technical University

Haonan Wang

Johns Hopkins University

Article

Keywords: Modular, hierarchical structure, CLP, intrinsic motivation, goal relabeling

Posted Date: May 3rd, 2024

DOI: https://doi.org/10.21203/rs.3.rs-4299675/v1

Additional Declarations: No competing interests reported.

IM3HRL: Model-assisted Intrinsically Motivated Modular Hierarchical Reinforcement Learning

Wei Liu[1, 2, 3, *], Jiaxiang Wang[1], Guangwei Liu[4], Haonan Wang[5]

[1]College of Science, Liaoning Technical University, Fuxin, China
[2]Institute of Mathematics and Systems Science, Liaoning Technical University, Fuxin, China
[3]Institute of Intelligent Engineering and Mathematics, Liaoning Technical University, Fuxin, China
[4] College of Mines, Liaoning Technical University, Fuxin, China
[5]Johns Hopkins University, Maryland, USA
*e-mail: liuwei@lntu.edu.cn

***Abstract*—Goal-conditioned reinforcement learning (GCRL) excels in tackling intricate decision-making tasks by managing multiple goals simultaneously. To enhance the agent's exploration and learning capabilities within complex tasks, modular goal representation is first combined with a hierarchical structure. Additionally, we introduce the CLP task priority metric and intrinsic motivation reward function to guide the agent, facilitating cross-module and cross-goal learning for efficient exploration. Furthermore, we implement a Future Goal Relabeling Strategy (FGRS) to diversify goals. Leveraging a learned dynamics model, the short-rollout is performed from the current policy to generate predictive goals, thus improving the efficiency of the policy optimization process. Finally, by integrating the above improvements, model-assisted intrinsically motivated modular hierarchical reinforcement learning (IM3HRL) is proposed. Experimental results in the Modular Goal Fetch Arm test environment demonstrate that IM3HRL outperforms other baseline methods, achieving a minimum 15% improvement in learning speed and exhibiting robustness against forgetting and sensor perturbations. Our research results prove the excellent model performance of IM3HRL and hold significant practical value for GCRL to solve the problem of exploratory learning in complex tasks.**

***Index Terms*—Modular, hierarchical structure, CLP, intrinsic motivation, goal relabeling.**

I. INTRODUCTION

The success of Deep Reinforcement Learning (DRL) on continuous control problems and games (such as Atari and Go) has led researchers to extend the learning capabilities of agents to handle multiple goals simultaneously, known as Goal Conditional Reinforcement Learning (GCRL) [1], [2]. At the technical level, the goal is a core data structure used in various fields to describe the conditions that need to be achieved or maintained, such as *goal reasoning* [3] and *robotics* [4]. In GCRL, the agent connects the current state and the goal by learning a universal value function approximator (UVFA). This enables the agent to simultaneously satisfy both the policy and the value function, ultimately achieving optimal goal-oriented behavior [5].

Constructing autonomous learning agents is a long-term goal in artificial intelligence, while the generation of intrinsic motivation agent is a pivotal step towards this goal for RL [6]. For example, the intrinsic motivation goal exploration process (IMGEP) allows the agent to explore and pursue goals without external rewards [7]. MACOB proposed by Forestier & Oudeyer employ population-based algorithms to achieve modular goals, making cross-goal learning in high-dimensional goal spaces possible [8]. CURIOUS, introduced by Cédric Colas et al., leverages intrinsic motivation and modular goal representation to acquire continuous and diverse sets of goals [9].

In complex tasks, the agent's state-action space becomes too large, leading to a rapid increase in parameters that hinders RL from achieving ideal results [10]. A key element in solving this type of problem is "combinability": decomposing complex tasks and solving the original problem by completing small goals, that is, hierarchical reinforcement learning (HRL) [11]. Tejas D. Kulkarni et al. introduced hierarchical DQN (h-DQN) with intrinsic motivation by integrating hierarchical value functions of different time scales [12]. HAC proposed by Andrew Levy et al. can make hierarchical agents learn together and overcome training process instability (non-stationarity) [13]. Ofir Nachum et al. employ the Off-policy model-free RL (MFRL) algorithm to learn hierarchical policies with less environment interaction, named as HIRO [14]. Tianren Zhang et al. restricted the high-level action space from the whole goal space to a k-step adjacent region of the current state using an adjacency constraint to address the training inefficiency [15]. Haotian Fu et al. proposed MGHRL, which learns to generate high-level meta strategies over subgoals given past experience and leaves the rest of how to achieve subgoals as independent RL subtasks [16].

Sparse reward is also one of the biggest challenges for RL. Hindsight Experience Replay (HER) addresses this by relabeling goals using failed experiences [17]. HER allows agents to learn from sparse and binary rewards without complex reward functions. In addition, there are some works [18], [19], [20], [21], [22] that enable agents to explore in a more efficient way by selecting suitable behavioral goals.

This work was supported in part by the National Natural Science Foundation of China (52374123, 51974144), Project of Liaoning Provincial Department of Education (LJKZ0340), and Project supported by discipline innovation team of Liaoning Technical University (LNTU20TD-01, LNTU20TD-07). *(Corresponding author: Wei Liu).*

All data generated or analysed during this study are included in this manuscript. If someone wants to request the data from this study, please contact the email: wangjx1392@163.com

However, current research still faces these issues: (1) Exploratory learning of complex tasks: Insufficient exploration in complex tasks can hinder robust value function acquisition, and dealing with large state-action spaces can lead to dimensionality challenges. (2) Goal diversity and sample efficiency: Goal relabeling methods often rely on past policy trajectories, which may not align well with the current policy. What's more, the goal diversity and sample efficiency will be poor since relabeling goals and updated tuples come from the same trajectory.

In the process of development, children will be driven by inner curiosity to explore, and then constantly improve and enrich the goals set with the guidance of parents / teachers to better plan in future [23]. Inspired by this concept, we introduce the Model-assisted Intrinsically Motivated Modular Hierarchical Reinforcement Learning (IM3HRL) framework. To address issue (1), we combine modular goal representation with hierarchical structure, while use the CLP task priority measure (CLP) and the intrinsic motivation reward function to guide the agent in complex environments for cross-module and cross-goal exploratory learning. For issue (2), a Future Goal Relabeling Strategy (FGRS) is proposed. FGRS leverages model-based RL (MBRL) method with better sample efficiency to rollout from the current state, facilitating the relabeling of goals and further enhancing policy optimization.

The specific contributions of this paper are as follows:

1) Modular hierarchical reinforcement learning architecture: MHA. MHA enables learning of continuous sets of diverse goals within a single hierarchical policy structure, enhancing exploration and efficiency in complex tasks.

2) Exploration learning mechanism with intrinsic motivation. We propose CLP and the reward function of hierarchical policies to guide the selection of modules and goals for replay, which can induce the agent to conduct efficient exploration.

3) A goal relabeling method combined with MBRL: FGRS. Leveraging a dynamics model for diverse goal generation, while optimizing policies through these relabeled goals enhances sample efficiency.

4) Provide an in-depth understanding of IM3HRL components, which offers new inspiration for the further development of GCRL.

The remainder of this article is structured as follows. Section II provides a summary of the relevant background knowledge, including GCRL, HER, and DDPG. Section III presents the main work of this paper, which focuses on MHA, CLP and FGRS. Section IV describes the overall framework of the proposed IM3HRL algorithm. In Section V, we demonstrate the robustness and advancement of IM3HRL through various types of experiments. Finally, the paper concludes with a summary and outlook.

II. BACKGROUND

A. Goal-Conditioned Reinforcement Learning

Markov decision process (MDP) is the basic core framework of RL, represented by the tuple $\langle S, A, P, r, g \rangle$, where $S \in \mathbb{R}^{d_s}$, $A \in \mathbb{R}^{d_a}$, $P(s_{t+1}|s_t, a_t)$, $r(s_t, a_t) \in \mathbb{R}$ and g denote the state space, action space, state transition function, reward function and discount factor respectively [24]. Different from general RL, GCRL uses the tuple $\langle S, A, P, \mathrm{G}, p_g, f, r_g, g \rangle$ augmented $\langle \mathrm{G}, p_g, f \rangle$ by to define MDP, where $\mathrm{G} \in \mathbb{R}^{d_g}$ is the goal space, p_g is the desired goal distribution of the environment, and $f : S \to \mathrm{G}$ is a tractable mapping function that maps state s to goal $g = f(s) \in \mathrm{G}$ [25]. In GCRL, the task is defined by the goal, and the goal itself is related to the state s. The reward function $r_g : S \times A \times \mathrm{G} \to \mathbb{R}$ is used to determine whether the goal is reached:

$$r_g(s_t, a_t, g) = \begin{cases} 1, \|f(s_{t+1}) - g\| \le e \\ 0, otherwise \end{cases} \tag{1}$$

The goal of GCRL is to find a goal-based policy $p(a|s, g)$ to maximize the expectation of discounted cumulative rewards over the goal distribution:

$$J(p) = \mathrm{E}_{a_t \sim p, g \sim p_g, s_{t+1} \sim P} \left[\sum_{t=0}^{\infty} g^t r(s_t, a_t, g) \right] \tag{2}$$

In GCRL, the desired goal is a required task to solve which can be provided by the environment or generated intrinsically, and the behavioral goal refers to the goal for sampling. If there is no well-designed goal selection mechanism, these two goals are usually the same.

B. Hindsight Experience Replay

Hindsight Experience Replay (HER) [17] is a data augmentation technique for Off-policy GCRL, which enables agents to train policy $p(a|s, g)$ in a sparse reward environment by using failure experience. In simple terms, the agent interacts with the environment to collect transitions $(s_t, a_t, r_g, g, s_{t+1})$ according to the behavior goal g, and uses a certain method (*final*, *future*, *random*, *episode*) to relabel g to generate new transitions $(s_t, a_t, r_g, g', s_{t+1})$. The most commonly used *future* strategy with the highest success rate is to randomly select a state s_{t+k} behind the same trajectory of the transition $(s_t, a_t, r_g, g, s_{t+1})$ for mapping to get $g' = f(s_{t+k})$. The reward r_g will also change, making the original sparse reward signal denser. This technique can be seen as a hidden curriculum of increasing difficulty for the agent generation, as it speeds up the efficient interaction between the agent and the environment.

C. Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) is a classic Off-policy RL algorithm in the field of continuous control [26], [27]. DDPG uses Actor-Critic neural networks to learn policy (Actor) and Q-function (Critic). Actor implements the controller and maps the current state s_t to the next action: $p : S \to A$. Critic approximates the optimal action-value function: Q^*, $Q : S \times A \to \mathbb{R}$. To enhance exploration,

Gaussian noise can be added on the actions to sample transitions.

DDPG stores the collected transitions into the replay buffer D, and randomly samples a batch of data from D in each iteration to complete the update of Actor and Critic. The action-value function Q is updated by minimizing the mean-squared Bellman error:

$$L(w,\mathrm{D}) = \mathop{\mathrm{E}}_{t:\mathrm{D}}[(g Q_w(s_{t+1},a_{t+1}) + r - Q_w(s_t,a_t))^2] \quad (3)$$

Where, w is the parameter of Critic, and $t = (s_t, a_t, r, s_{t+1})$ is a transition.

The gradient of the policy and Q-function is:

$$\tilde{N}_J J(p_J) = \mathop{\mathrm{E}}_{s_t:\, r^b}\left[\tilde{N}_a Q_w^p(s_t,a_t)\tilde{N}_J p_J(s)\big|_{a=p_J(s)}\right] \quad (4)$$

$$\tilde{N}_w J(Q_w) = \mathop{\mathrm{E}}_{s_t:\, r^b}[\tilde{N}_w (Q_w(s_t,a_t) - r - g Q_w(s_{t+1},a_{t+1}))^2] \quad (5)$$

Where, r^b is the action strategy used to collect data.

III. METHODOLOGY

This section proposes a modular 2-layer hierarchical policy framework, mapping "tasks" to "modules". Each task includes multiple goals, serving as a link between high-level and low-level policies, facilitating efficient learning of multi-task and multiple-goal scenarios. Section 3.1 presents a modular approach to representing tasks and goals, enabling agents to learn continuous sets of diverse goals within a single hierarchical policy structure. Section 3.2 explores the hierarchical policy structure, encompassing high-level and low-level policies. The intrinsic motivation reward function of low-level policy can enhance the agent's comprehension of goal space, while the hierarchical structure can accelerate learning. The CLP introduced in Section 3.3 can not only promote exploration ability, but also achieve generalization learning across tasks. Section 3.4 presents FGRS with MBRL, which can diversify goal sets, and policy optimization based on relabeled goals can further improve sample efficiency. These methods encourage efficient exploration, expedite learning, and provide faster solutions to multi-task and multiple goal challenges.

A. Modular task and goal representation

In most current works [28], [29], [30] on solving multi-task reinforcement learning (RL) problems, the agent doesn't explicitly state its current task. Instead, it aims to maximize the overall reward through a high-dimensional goal space, resulting in low learning efficiency.

In environment E with N tasks, the agent may want to "move the cube to the target position x" (task: "*Reach*") or "pick up and place the cube" (task: "*Pick & Place*"), which requires modular goal representation to describe the full goal space, that is, goals are organized by modules. Multiple goal tasks correspond to individual modules, forming the module set $\mathrm{M} = \{M_i\}_{i=1,\mathrm{L},N}$. Each module $M_i = (\mathrm{R}_{M_i, g_i \in \mathrm{G}_{M_i}}, \mathrm{G}_{M_i})$ is the pair of a reward function and a goal space. The reward function is parameterized by the module and the goal, specifying a set of constraints that must be satisfied by the agent's state (e.g., reaching a certain state for "Reach" (M_i)), and the goal (e.g., $g_i = x$) continuously evolves in the associated goal space G_{M_i} of M_i (e.g., 3D Euclidean space).

We utilize a module descriptor m_d of size N (one-hot encoding) to encode the current module (e.g., for an environment with three modules M_1, M_2, M_3, if the agent is currently trying M_3, then $m_d = (0,0,1)$). Given G_{M_i}, define the current goal g as a vector of dimension $|\mathrm{G}| = \sum_{i=1}^{N} |\mathrm{G}_{M_i}|$, where different modules may have varying goal space dimensions. After selecting M_i and the corresponding goal $g_i \in \mathrm{G}_{M_i}$, g is set to 0 everywhere except for g_i. To ensure this, we freeze the weights related to unselected goals during the backpropagation of neural network.

B. Modular Hierarchical Architecture

To efficiently tackle modular multiple goal learning challenges, we propose the Modular Hierarchical Architecture (MHA), uniting modular goal representation with a hierarchical framework. In the 2-layer hierarchical structure, the high-level policy $p_h(g|s_t)$ extracts goals $g_i \in \mathrm{G}_{M_i}$ of the current module M_i to guide the low-level policy $p_l(a_t|s_t, g, m_d)$ in performing actions. Both high-level and low-level policies conduct self-assessment based on the feedback of reward signals and continuously improve to optimize overall performance. The reward signals can be either raw rewards from the environment or from engineered goals. We train p_h and p_l separately, so p_l is agnostic to p_h. While, the policies share the same G, enabling the reuse of the low-level policy for different high-level policies. Modular Hierarchical Architecture is shown in Fig.1.

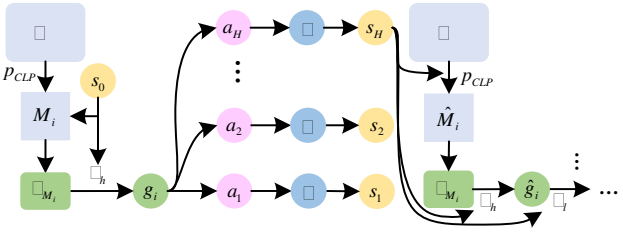


Fig. 1. MHA: The high-level policy p_h selects the goal g_i for the low-level policy p_l from the goal space G_{M_i} of module M_i.

1) High-level policy

p_h explore and act in the goal space by providing instructions to p_l, and is trained to solve the standard MDP: $\langle S, \mathrm{G}_{M_i}, P_h, R_{env}, g_h \rangle$. Where, the action space is G_{M_i}, the state space aligns with p_l, and the reward function $R_{env}: S \times \{0,1\}$ is supervision from the environment to judge whether the goal is achieved. To achieve a goal usually requires a series of agent actions, so we execute a fixed number (H) of actions with p_l for each high-level instruction and store the results in D_{env}. In

principle, any RL algorithm can train the high-level policy, we use the transitions in D_{env} for training via DDPG.

2) Low-level policy

p_l follows the instructions of p_h, observes environmental feedback (including new states and rewards), then passes it to p_h.

p_l is used to solve the augmented MDP: $\langle S, A, P_l, R_{m_d,g}, g_l, M, G_{M_i} \rangle$. Where, the reward function $R_{m_d,g}$ is parameterized by m_d and g. A simple way to represent $R_{m_d,g}$ is to map the state, module, and goal to a single Boolean value $R_{m_d,g}: S \times M \times G_M \to \{0,1\}$, which describes whether s satisfies the constraints described by g in m_d. If satisfied, the reward is 1; otherwise, it's 0. To ensure that the actions of the agent can induce rewards, we define the intrinsic motivation reward function as:

$$R_{m_d,g}(s_t, a_t, m_d, g, s_{t+1}) = \begin{cases} 0, & if\ Y(s_{t+1}, g) = 0 \\ Y(s_{t+1}, g) \wedge Y(s_t, g), & if\ Y(s_{t+1}, g) = 1 \end{cases} \quad (6)$$

Where, $Y: S \times G \to \{0,1\}$ indicates whether s satisfies g. It is worth noting that in the state space S, any goal can be simply represented by a Boolean function of the above form by checking the proximity of two states up to a threshold parameter.

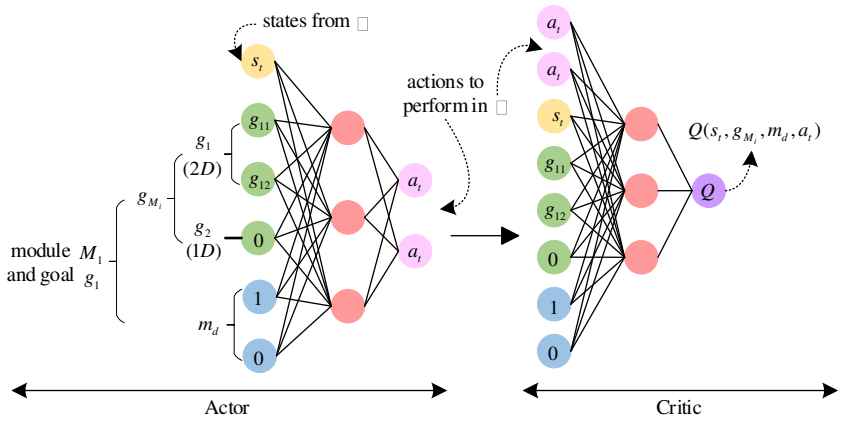


Fig.2. Low-level policy network structure parameterized by modular objectives. In this example, 2 modules are parameterized by g_1 (2D) and g_2 (1D) respectively. The agent is trying goal $g_1 = [g_{11}, g_{12}]$ in module M_1, which is specified by the one-hot module descriptor $m_d = (1,0)$. Actor (left) is responsible for computing a_t, and critic (right) is responsible for calculating the Q-value.

Like CURIOUS, we use UVFA to link the agent's modular goal with current state to form the input ($[s_t, g, m_d]$) of the policy network and the value function ($Q(s_t, g, m_d, a_t)$) realized by the deep neural network. The Actor-Critic network of p_l is shown in Fig. 2. Actor maps $[s_t, g, m_d]$ to the next action by executing the action policy, and Critic uses the concatenation $[s_t, g, m_d, a_t]$ of a_t and $[s_t, g, m_d]$ as input to calculate an estimate of Q-value. p_l can be trained with any Off-policy RL algorithm, and we also choose DDPG.

C. Module selection and cross-module learning

To timely assess task completion and make informed choices, we implement an automatic curriculum strategy inspired by CURIOUS. We track both the agent's *competence* (C) and *learning progress* (LP) to measure task priority. This method is denoted as CLP task priority measurement (abbreviated as CLP).

Like the training of p_h, the agent periodically assesses its goal achievement using transitions from D_{env}. For each module, the result (1 for success, 0 for failure) is stored in competence queues $results^{(i)}$. The agent's competence $C_{M_i}: t \to p_{success}(t)$ to complete M_i is the probability of success at time *t*, and the calculation formula is:

$$C_{M_i} = \frac{1}{l} \sum_{j=0}^{l-1} results^{(i)}(n^{(i)} - j) \quad (7)$$

Where, $n^{(i)}$ is the number of transitions used to evaluate M_i.

LP is defined as the derivative of the competence: $LP_{M_i} = \frac{dC_{M_i}}{dt}$, signifying the recent success rate change in M_i. A faster change implies a higher LP. For simplicity, we approximate LP as the difference between the success rate in the most recent half and the first half of the history, namely:

$$LP_{M_i} = C_{M_i}(recent(n^{(i)}/2)) - C_{M_i}(earlier(n^{(i)}/2)) \quad (8)$$

The agent primarily focuses on modules with the largest absolute LP, while giving less attention to modules that are already solved or deemed unsolvable, characterized by lower $|LP_M|$. Utilizing $|LP_M|$ allows the agent to prioritize modules, addressing the forgetting issue, allocating resources sensibly, and enhancing the learning efficiency of the entire multi-task system. A high $|LP_M|$ represents two significant changes: (1)

The success rate of the module increases rapidly, that is, the task has made good progress, and priority is given. (2) The success rate of the module drops rapidly, indicating potential task forgetfulness and the need for relearning with higher probability.

CURIOUS [9] uses proportional probability matching in combination with the ϵ-greedy strategy to select modules, that is, the calculation of the LP probability $p_{LP}(M_i)$ is:

$$p_{LP}(M_i) = \epsilon' \frac{1}{N} + (1-\epsilon)' \frac{\left|LP_{M_i}\right|}{\sum_{j=1}^{N}\left|LP_{M_j}\right|} \tag{9}$$

The left term of (9) is called the random exploration term, which allows for module sampling irrespective of CLP, with a probability of ϵ (whether the module is solved, too difficult, or in the platform period). Conversely, the right term biases module selection with the probability of $(1-\epsilon)$. This way, the agent can assess whether it can handle modules it has already learned or choose to persist with modules currently considered too difficult.

While some papers [8], [9] have used LP for module selection, we further consider the achievement of goals within M_i, allowing the agent to explore more comprehensively. The enhanced CLP probability $p_{CLP}(M_i)$ is:

$$p_{CLP}(M_i) = \epsilon' \frac{1}{N} + (1-\epsilon)' \frac{(1-C_{M_i})*\left|LP_{M_i}\right|}{\sum_{j=1}^{N}(1-C_{M_j})*\left|LP_{M_j}\right|} \tag{10}$$

$(1-C_{M_i})$ is added to the right term of (10) to promote the agent's preference for selecting modules with high $|\mathrm{LP}|$ and low C (easier to learn) during data collection and module substitution. This encourages the agent to tackle modules with higher failure rates, ultimately enhancing its exploratory learning capabilities. For the update of $p_{CLP}(M_i)$, we set up self-evaluation rollouts to separate it from the training process of the agent.

CLP is mainly used for module selection and cross-module learning: (1) Before interacting with the environment, the initial $p_{M_i}(CLP)$ is used for module selection to complete data collection. (2) During the agent training process, transitions are sampled from the replay buffer, and the new module $\hat{m}_d$ is sampled from $\mathbf{M}$ based on the updated $p_{M_i}(CLP)$ to substitute m_d.

As an example, in an environment with four tasks, the calculated probability $p_{M_i}(CLP) = [0.5, 0.2, 0.2, 0.1]$ is used to guide the agent to learn towards modules with high $|\mathrm{LP}|$ and low C. If the size of the mini-batch is n_p, the agent will sample $\lfloor n_p \times 0.5 \rfloor$ transitions associated with M_1, $\lfloor n_p \times 0.2 \rfloor$ transitions associated with M_2, $\lfloor n_p \times 0.2 \rfloor$ transitions associated with module M_3, and $\lfloor n_p \times 0.1 \rfloor$ transitions associated with module M_4. In this mini-batch, all transitions are sampled for training a particular module (e.g., $\hat{m}_d$), although they could be collected when targeting another module (e.g., m_d). To perform this cross-module learning, simply substitute the latter (m_d) with the former ($\hat{m}_d$).

D. Future goal relabeling strategy

FGRS aims to diversify goals for low-level policies based on the agent's historical experience. After completing the module substitution, FGRS selects transitions from the mini-batch data with ratio x, then performs short-rollout from the current state s_t using the MBRL method to generate future goals.

The environmental dynamics model F of RL is usually unknown. In MBRL, the agent interacts with the environment by executing the current policy $p(a_t|s_t)$ and collects data to model F [31], [32]:

$$F(s_{t+1}|s_t, a_t) = \mathrm{E}_{(s_t, a_t, s_{t+1})\sim \mathrm{D}}[P(s_{t+1}|s_t, a_t)] \tag{11}$$

We use the ensemble of bootstrapped probabilistic dynamics models proposed by Chua et al. as the dynamics model. Previous research has shown that this approach substantially reduces the amount of data needed for task learning and yields improved asymptotic performance [33].

Define $F_q(s_{t+1}|s_t, a_t)$ as a dynamics model parameterized by q, its bootstrap ensemble is $\{f_q^1, \mathrm{L}, f_q^B\}$, and each member is a probabilistic neural network. Given inputs s_t and a_t, the output of each probabilistic neural network is a Gaussian distribution whose diagonal covariance parameterized by q, that is:

$$f_q^k(s_{t+1}|s_t, a_t) = \mathrm{N}(m_q^k(s_t, a_t), \mathrm{S}_q^k(s_t, a_t)) \tag{12}$$

We learn the dynamics model through the maximum likelihood method, and the negative log-likelihood loss (NLL) [34] is:

$$\mathrm{L}_{NLL}(q) = \sum_{t=1}^{T} [m_q(s_t, a_t) - s_{t+1}]^{\mathrm{T}} \mathrm{S}_q^{-1}(s_t, a_t)[m_q(s_t, a_t) - s_{t+1}] + \log\det \mathrm{S}_q(s_t, a_t) \tag{13}$$

The dynamic model will continuously compound the errors with the real environment model during multi-step rollout, resulting in the compounding error [35], [36]. Therefore, after learning the dynamics model, the agent foresees future goals through short-rollout. This can not only improve the diversity of goals, but also not lead to excessive errors [37].

Firstly, mini-batch transitions are selected from the data that has completed the module substitution with ratio x, and one f_q^k is randomly selected from F_q. Then, for the transition $(s_t, a_t, r_{\hat{m}_d, g_t}, \hat{m}_d, g_t, s_{t+1})$, f_q^k generates the simulation trajectory from state s_t and p_l selects the action according to the state, so as to perform h-step forward to obtain $\hat{s}_{t+h}$. Thirdly, get the future goal according to the mapping function f:

$$\hat{g}_t = \mathrm{E}_{\hat{a}_{t+n}\sim p_l(\cdot|\hat{s}_{t+n}, g_t, m_d), \hat{s}_{t+n+1}\sim f_q^k(\cdot|\hat{s}_{t+n}, g_t, \hat{a}_{t+n})}[f(\hat{s}_{t+h})] \quad 0 \le n \le h-1 \tag{14}$$

Substitute the original goal g_t with $\hat{g}_t$. Finally, $r_{\hat{m}_d,\hat{g}_t}$ is calculated using the reward function $R_{m_d,g}$ and the transition $(s_t, a_t, r_{\hat{m}_d,\hat{g}_t}, \hat{m}_d, \hat{g}_t, s_{t+1})$ is stored in D_{model}. It should be noted that since the tuple used for training is $(s_t, a_t, r_{\hat{m}_d,\hat{g}_t}, \hat{m}_d, \hat{g}_t, s_{t+1})$, the first step rollout needs to be performed from s_{t+1} when relabeling s_t. The pseudo code is shown in Algorithm 1.

Algorithm 1: Future Goal Relabeling Strategy (FGRS)

Input: Dynamics model F_q, Current low-level policy p_l, Mapping function f, Transitions of module substitution is completed Mem, Reward function $R_{m_d,g}$, Maximum rollout length h, Ratio x.

Output: Relabeled replay buffer D_{model}.

Use ratio x to sample a mini-batch of transitions $(s_t, a_t, r_{m_d,g_t}, \hat{m}_d, g_t, s_{t+1})$ from Mem;

for *every tuple* $(s_t^j, a_t^j, r_{m_d,g_t}^j, \hat{m}_d, g_t^j, s_{t+1}^j)$ **do**

Randomly choose a rollout step h uniformly in $(1, h]$;

Choose a new behavioral goal through the *future* strategy HER;

Randomly select a f_q^k from F_q;

Use f_q^k and $\hat{a}_{t+1}^j \sim p_l(\cdot|\hat{s}_{t+1}^j, g_{new}^j, \hat{m}_d)$ to take h-step rollout on s_{t+1}^j and get $\hat{s}_{t+h+1}^j$, where $t+1 \le t+h$;

Relabel g_t^j with $\hat{g}_t^j = f(\hat{s}_{t+h+1}^j)$;

Calculate new reward $r_{\hat{m}_d,\hat{g}_t^j}^j = R_{m_d,g}(s_t^j, a_t^j, \hat{m}_d, \hat{g}_t^j, s_{t+1}^j)$;

end for

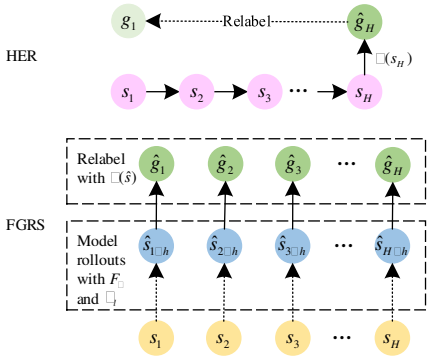


Fig. 3. The working principle of FGRS and HER

The working principle of FGRS and HER is shown in Fig.3. Compared with HER, FGRS offers the following advantages: (1) By utilizing MBRL for short-rollouts to generate various future goals, FGRS enhances sample diversity in the optimization process of low-level policies, leading to improved algorithm performance. (2) FGRS regenerates goals based on the current model and policy, whereas HER relies on the previous ones, potentially leading to significant deviations from the training of the current policy.

IV. FRAMEWORK

Fig.4 shows the process structure of IM3HRL. The specific steps are as follows:

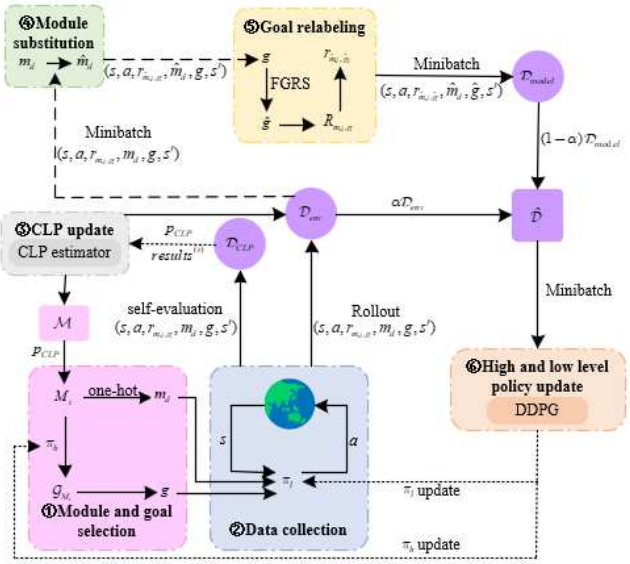


Fig. 4. Schematic view of IM3HRL

1) Module and goal selection: The agent selects modules and goals to interact with the environment. In this part, a random number is first generated in each episode, and we decide whether to perform self-evaluation rollout by judging if this random number is less than the probability p_{eval}. If so, randomly choose a model M_i uniformly in M. Otherwise, M_i is selected from M according to p_{CLP}, and g_i is selected from the corresponding goal space G_{M_i} using $p_h(g|s_0)$, where $\mathrm{G} \subseteq S$.

2) Data collection: The agent interacts with E and uses the current $p_l(a_t|s_t, g_i, m_d)$ to collect transitions. Then, if it's a self-evaluation rollout, the data is stored in D_{CLP}, otherwise stored in D_{env}. In particular, self-evaluation rollout does not use exploratory noise.

3) CLP update: If the agent is performing a self-evaluation rollout, p_{CLP} is updated using Equation (10) based on the results collected in D_{CLP}.

4) Model substitution: The agent decides which modules to train. To update the high-level and low-level policies, the agent first uses the updated p_{CLP} to sample mini-batch data from D_{env}. If you want to train on the module M_j, substitute the original module descriptor m_d with $\hat{m}_d$ corresponding to M_j, and sample the mini-batch transitions from D_{env} according to the updated p_{CLP}, thereby achieving cross-module learning.

5) Goal relabeling: According to the mini-batch transitions after module substitution, FGRS use f_q^k and p_l to perform short-rollout with ratio x. The original goal g_t in these transitions is relabeled as $\hat{g}_t$, and the reward for each transition is calculated by $r_{\hat{m}_d,\hat{g}_t}$. After that, $(s_t, a_t, r_{\hat{m}_d,\hat{g}_t}, \hat{m}_d, \hat{g}_t, s_{t+1})$ is stored in D_{model}.

Algorithm 2: IM3HRL

Input: Environment E, Model set M, Goal space $G_{1:N}$, Noise d, Mapping function f, Reward function $R_{m_d,g}$, Maximum rollout length h, Probability p_{CLP_eval}, x, a.

Initialize: Replay buffer D_{CLP}, D_{env}, D_{model}, Mem, Dynamics model F_q, High-level policy p_h, Low-level policy p_l, CLP probability p_{CLP}.

for *episodes* = 1: E **do**

$s_0 \leftarrow E.reset()$;

$CLP\_eavl \leftarrow random() < p_{CLP\_eval}$; // If Ture, the agent will perform a self-evaluation rollout

$md(M_i) \leftarrow ModuleSelector()$; // $M_i \sim p_{CLP}$ (If it is self-evaluation, then uniform and random selection)

$goal \leftarrow p_h(g|s_t)$;

for t = 0: H **do**

$a_t \leftarrow p_l(a_t|s_t, m_d, goal)$;

if *not* CLP_eavl **then**

$a_t \leftarrow a_t + d$;

end if

$s_{t+1} \leftarrow E.step(a_t)$;

$r_{m_d,goal} \leftarrow R_{m_d,g}(s_t, a_t, m_d, goal, s_{t+1})$;

end for

if CLP_eavl

$D_{CLP}.add(s_t, a_t, r_{m_d,goal}, m_d, goal, s_{t+1})$;

$p_{CLP} \leftarrow CLP\_Update(D_{CLP})$;

else

$D_{env}.add(s_t, a_t, r_{m_d,goal}, m_d, goal, s_{t+1})$;

$Mem \leftarrow ModuleReplacing(p_{CLP}, D_{env})$;

According to Eq. (13), use NLL loss to update F_q;

$D_{model} \leftarrow FGRS(F_q, p_l f, Mem, R_{m_d,g}, h, x)$;

$\hat{D} \leftarrow a D_{env} + (1-a) D_{model}$;

$p_l \leftarrow RL\_Update(A_l, \hat{D})$;

$p_h \leftarrow RL\_Update(A_h, \hat{D})$;

end for

6) RL update: Use the mini-batch data in D_{env} and D_{model} to update p_l and p_h through the RL algorithm A_l and A_h. For $p_l(a_t|s_t, g, m_d)$, we extend $Q(s_t, a_t)$ to a modular general Q-function $Q(s_t, g, m_d, a_t)$, then use the transitions generated by the dynamics model and DDPG[25] for optimization. The update gradient of the parameterization policy and Q-function are:

$$\tilde{N}_l J(p_l) = \mathop{E}_{\substack{(s_t,g,m_d,a_t)\sim \hat{D} \\ a_t \sim p_l}} \left[\tilde{N}_{a_t} Q(s_t, g, m_d, a_t) \tilde{N}_l p_l(a_t|s_t, g, m_d) \right] \tag{15}$$

$$\tilde{N}_w J(Q_w) = \mathop{E}_{\substack{(s_t,g,m_d,a_t,s_{t+1})\sim \hat{D} \\ a_t \sim p}} [\tilde{N}_w (Q_w(s_t, g, m_d, a_t) - r - gQ_w(s_{t+1}, g, m_d, a_{t+1}))^2] \tag{16}$$

Where, Q_w is the objective Q function. $\hat{D} \leftarrow a D_{env} + (1-a) D_{model}$ is a mixture of the real experience D_{env} obtained by interacting with the environment and the short-rollout experience D_{model} generated by the dynamics model.

For $p_h(g|s_t)$, we also use DDPG[25] for optimization. The update gradient of the parameterization policy and Q-function are:

$$\tilde{N}_h J(p_h) = \mathop{E}_{\substack{(s_t,g)\sim \hat{D} \\ g \sim p_h}} \left[\tilde{N}_g Q(s_t, g) \tilde{N}_h p_h(g|s) \right] \tag{17}$$

$$\tilde{N}_w J(Q_w) = \mathop{E}_{\substack{(s_t,g,s_{t+1})\sim \hat{D} \\ g \sim p_h}} [\tilde{N}_w (Q_w(s_t, g) - r - gQ_w(s_{t+1}, g))^2] \tag{18}$$

The pseudocode of IM3HRL is shown in Algorithm 2.

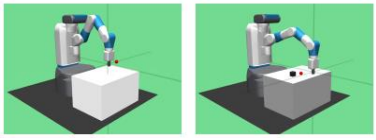
V. EXPERIMENT

This section answers the following questions through different types of experiments: (1) Is the combination of modular task and goal representation with hierarchical structure effective? (2) How does the module selection mechanism of intrinsic motivation CLP perform? (3) Can relabel goals using FGRS achieve superior results compared to HER? (4) Does IM3HRL outperform previous methods in complex tasks?

To address the aforementioned questions, Section 5.1 introduces the simulation environment essential for the experiments. Section 5.2 presents the success rate of the MHA architecture in tackling complex tasks compared with other baseline architectures. Section 5.3 visualizes the agent's learning process and validates CLP's robustness against forgetting and sensor perturbations. Section 5.4 compares the learning curves of FGRS and HER to verify the effectiveness of FGRS. Then, Section 5.5 compares the experimental results of IM3HRL with other baseline methods to test whether its learning performance is better. Finally, Section 5.6 sets up ablation experiments with different hyperparameters to obtain the optimal value of each hyperparameter.

A. Experimental environment and related settings

We use the Modular Goal Fetch Arm proposed by Cédric Colas et al. based on the Fetch environment of the OpenAI Gym as the simulation environment, shown in Fig.5. The agent is a 7-DoF Fetch robotics arm with a two-fingered parallel gripper, which faces 2 cubes randomly placed on the table and can target many diverse goals. The action space is 4-dimensional: 3-dimensions specify the desired gripper in Cartesian coordinates, and 1-dimension controls opening and closing of the gripper. The observation space has 40 dimensions (Additional details can be found in the original paper presenting the Fetch environment [38]).



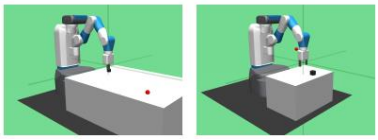


Fig. 5. The Modular Goal Fetch Arm test environment.

In this environment, agents can target a different set of modular goals: (M_1 : Reach) Reach a 3D target with the gripper; (M_2 : Push) Push cube 1 onto a 2D target on the table; (M_3 : Pick & Place) Pick and place cube 1 on a 3D target; (M_4 : Stack) Stack cube 1 over cube 2. Additionally, additional Push modules can be defined as distractor modules, containing more impossible or distracting goals, namely: (M_5 -) Push a randomly moving cube to a 2D target on the table, which is out of reach. With each additional cube added, the observation space increases by 3 dimensions. All simulation experimental environments in this paper include 4 achievable modules and 4 distracting modules.

In all Fetch tasks, rewards are sparse and binary. The internal reward function $R_{m_d,g}$ of p_l is parameterized by m_d and g. When the constraints defined by the module satisfy the current goal and result, the reward function is 1. For the Push and Pick & Place modules, the reward is 1 when the distance between the cube and the target position is within $e_{reach} = 0.05$ (simulation units), and 0 otherwise. For the Reach module, the same criteria apply to the distance between the gripper and the target. Finally, for the Stack module, in addition to the limitation of the target-cube distance, the target-grabber distance must be greater than $1.2' e_{reach}$ to ensure that two cubes are stacked together instead of being gripped by the gripper.

In all experiments, the episode is set to $30' 10^4$, the maximum capacity of D_{env}, D_{model} and $\hat{D}$ is 10^6, the number of probabilistic neural networks in the dynamics model bootstrap ensemble is 5, and the maximum rollout length is 30.

We used the same settings as CURIOUS [9] for the policy network parameters, the probability $p_{CLP\_eval} = 0.1$ that the agent performs self-evaluation rollout and the length $l = 300 episode$ of the windows considered for computing C and LP. In addition, we set two extra hyperparameters: ①Set the exploration parameter controlling the mixture of random module selection and CLP-based active module selection to e; ②Set the ratio of selecting mini-batch transitions for short-rollout in FGRS to x.The ablation experiments in section 5.6 demonstrated that IM3HRL achieves optimal performance when $e = 0.4$ and $x = 0.8$.

B. Impact of Modular Hierarchical Architecture

This section aims to explore the impact of MHA on agent performance and compare it with M-UVFA from CURIOUS [9]. The modular multiple goal architecture using universal approximators (M-UVFA) represents the goal in a modular manner and uses UVFA to concatenate the agent's goal with the current state to form the input of the policy and the value function implemented by the deep neural network.

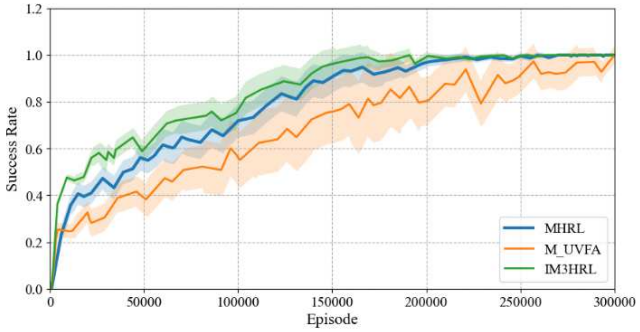


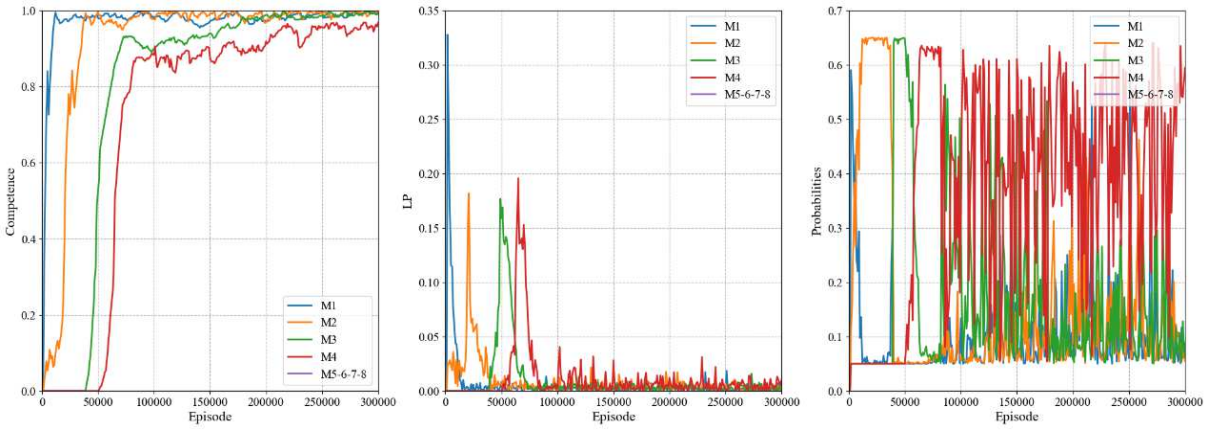
Fig. 6. The impact of MHA architecture on agent performance. The solid line and the shade represent the mean and standard deviation of the success rate respectively, which is calculated over 10 different random seeds.

Fig.6 shows the ablation experiment results of MHA, M-UVFA and IM3HRL in Modular Goal Fetch Arm test environment. The solid line represents the average of the success rates over 10 different random seeds, and the shaded area is the standard deviation of these 10 trajectories. According to the experimental results, compared with M-UVFA, MHA can learn reachable goals faster ($20' 10^4$ *vs.* $25' 10^4$), signifying that the combination of modular goal representation with hierarchical structure effectively enhances the agent's exploration and learning capabilities. This is because modularity enables agent to pursue a diverse range of modular goals within a single hierarchical network, while hierarchical structures partially mitigate the curse of dimensionality by breaking down tasks. However, there is still a certain gap between the results of MHA and IM3HRL, indicating that solely introducing MHA is insufficient for significantly improving the agent's ability to excel in solving complex tasks.

C. Impact of CLP

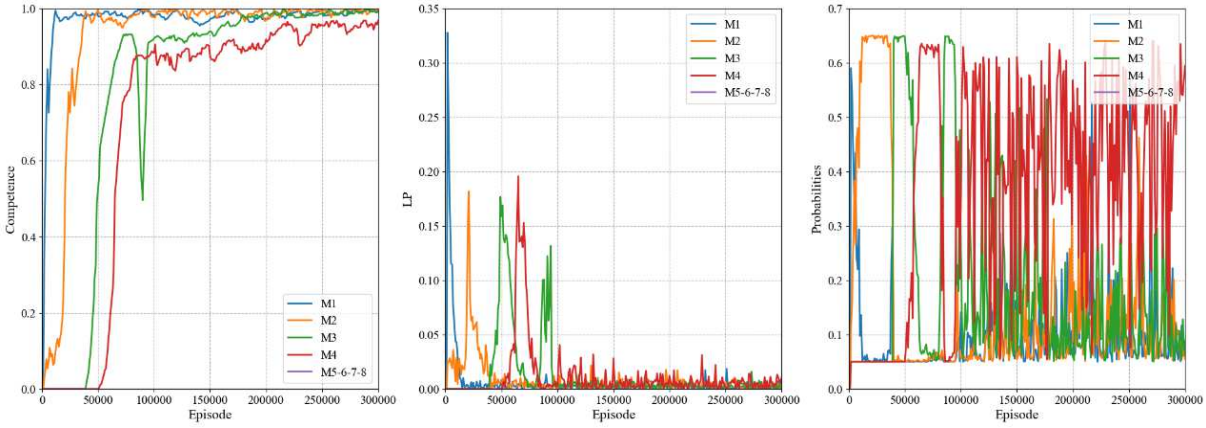
1) Visualizing the Intrinsic Motivation towards CLP

This section aims to elucidate the inner workings of the module selection mechanism driven by the intrinsic motivation towards CLP. Fig.7(a), (b) and (c) respectively show the evolution process of the agent's subjectively perceived module-dependent competence C, the corresponding module learning progress $|\mathrm{LP}|$ and module selection probability p_{CLP} in a single running experiment. The results demonstrate that modules are learned in a progressive manner, indicating the existence of successive learning phases.



(a) Module-dependent competence C (b) Model learning progress $|\mathrm{LP}|$ (c) Module selection probability p_{CLP}

Fig. 7. Visualization of intrinsic motivation towards.



(a) Module-dependent competence C (b) Model learning progress $|\mathrm{LP}|$ (c) Module selection probability p_{CLP}

Fig. 8. The resilience of intrinsic motivation towards CLP to forgetting

The agent initially focuses on mastering simpler modules, such as controlling the gripper (M_1), followed by learning to move cube 1 to the target position (M_2),then picks cube1 up (M_3) and stacks over cube 2 (M_4). As depicted in Fig.7, the agent's attention shifts from easier modules (low $|\mathrm{LP}|$, high C) to more challenging ones. When a module is successfully learned, its $|\mathrm{LP}|$ gradually increases, and p_{CLP} of the corresponding module being selected also increases.

Similar to human developmental learning behavior, in the early phases of learning, when some modules are simpler, the agent learns the module faster. The consistent rise of LP allows the agent to concentrate on these modules, leading to increased success. Subsequently, with the proactive guidance of CLP, the agent's learning trajectory shifts, and it starts to tackle modules that were either forgotten or considered too challenging to solve.

2) Resilience to Forgetting and Sensor Perturbations

a) Resilience to Forgetting

During the learning process, the agent may forget previously mastered modules for various reasons (like catastrophic forgetting, environmental alterations, or internal changes). Ideally, the CLP mechanism should promptly detect such situations and respond immediately. This section will investigate CLP's adaptability to such perturbations and compare it with a modular layered architecture that does not use CLP.

Fig.8 shows a case of forgetting and explains how the CLP detects and reacts in a timely manner. Since forgetting cannot be triggered, we set up an experiment to simulate time-locked sensory failure: triggering a sensory perturbation at an accurate time (epoch=250, episode= $25' \ 10^4$), so that the perception of cube 2 gets shifted by 0.05 (simulation units) until the end of the run.

Observing Fig.8(a), we notice a decline in the agent's ability around episode= $9' \ 10^4$, indicating a case of catastrophic forgetting. This is due to the fact that the agent is trained on other modules at this time, causing it to inexplicably forget previously mastered ones (M_3). From the corresponding moments of Fig.8(b) and (c), we observe a gradual increase in both $|\mathrm{LP}|$ and p_{CLP} of M_3 , which in turn triggers the extra attention of the agent to M_3 . From the above experimental results, it's evident that CLP proves valuable in addressing the forgetting issue and facilitating parallel learning of multiple modules. CLP enables the agent to monitor its proficiency in each module and adjust its focus in a timely manner to counteract any forgetting of previously mastered modules.

b) Resilience to Sensor Perturbations

To prove the robustness of CLP to sensor perturbations, we use the above time-locked simulation experiment to compare IM3HRL with MHA.

In Fig.9, it can be observed that after the episode= $25' 10^4$ perturbation, the average success rate of IM3HRL and MHA both decreased by about $1/4$, which is due to the fact that the set perturbation only affects M_3. The experimental results show that IM3HRL can respond more quickly after detecting disturbances and use more transitions to learn M_3. Compared with MHA needs $9.5' 10^4$ episode, IM3HRL only needs $5.5' 10^4$ episode to restore the agent to its pre-interference performance. This implies that IM3HRL's recovery speed is increased by about 42 %, that is, it has stronger robustness.
Define abbreviations and acronyms the first time they are used in the text, even after they have already been defined in the abstract. Abbreviations such as IEEE, SI, ac, and dc do not have to be defined. Abbreviations that incorporate periods should not have spaces: write "C.N.R.S.," not "C. N. R. S." Do not use abbreviations in the title unless they are unavoidable (for example, "IEEE" in the title of this article).

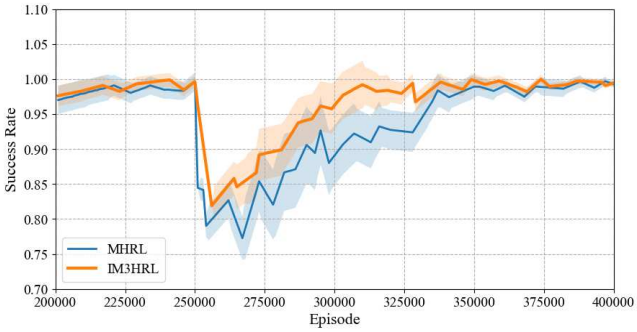


Fig. 9. Resilience of intrinsic motivation towards CLP to sensor perturbations. The average success rate +/-std of 10 experiments was plotted, and IM3HRL is superior to MHA in testing.

D. Impact of FGRS

This section aims to confirm whether FGRS can lead to improved outcomes for agents compared to HER. Figure 9 shows the ablation results of IM3HRL with FGRS (IM3HRL-FGRS), IM3HRL with HER (IM3HRL-HER) and CURIOUS in the modular test environment.

According to the experimental results of Fig.10, FGRS can achieve the highest success rate in $19' 10^4$ episode, while HER and CURIOUS need $26' 10^4$ and $30' 10^4$ episode respectively. In addition, FGRS can stabilize at the highest success rate after $25' 10^4$ episode, while HER remains stable after the $28' 10^4$ episode. The above results show the advantages and effectiveness of FGRS. This is because in the design of this paper, FGRS can provide more diverse goals for agents from the current policy. Consequently, agents can access more valuable information, effectively enhancing their learning efficiency.

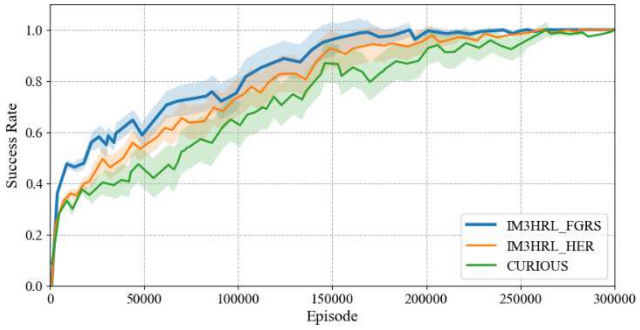


Fig. 10. The impact of FGRS on the performance of the agent. The average success rate +/-std of 10 experiments was plotted. According to the observation, compared with HER, FGI helps agents learn faster.

E. IM3HRL comparative experiment and results

This section aims to understand the impact of the proposed IM3HRL algorithm on the performance of the agent, and selects the following baseline methods for comparative testing:

Intrinsically motivated modular multiple goal reinforcement learning (CURIOUS) [9]**:** The algorithm uses a modular universal value function approximator and an absolute learning progress to efficiently learn a continuous diversity goal set.

A flat multiple goal architecture (HER+DDPG) [17]**:** The goal is represented in a linear way, and the corresponding goal is selected in a holistic goal space G. The agent needs to satisfy the constraints described by all modules at the same time to obtain the reward. This goal-parameterized architecture is equivalent to UVFA.

A multiple goal module-experts architecture (MG-ME) [39]**:** A multi-objective expert policy is trained for each of the N modules. Each policy is trained one epoch every N on its specified module and the collected transitions are shared with other experts. When evaluating a specific module, the algorithm uses the corresponding module-expert.

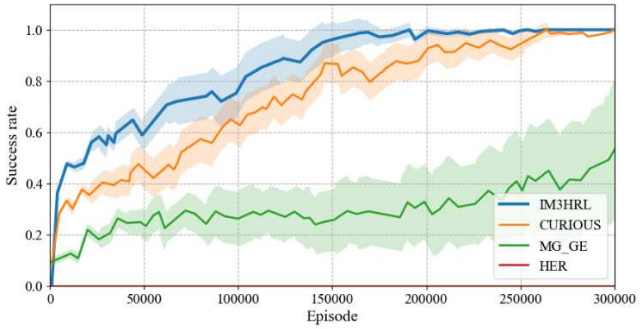


Figure. 11. Comparative experimental results of IM3HRL. The average success rate +/-std of 10 experiments was plotted, and the significance of IM3HRL compared with other baseline methods can be seen.

In Fig.11, the average success rate of HER+DDPG remains consistently at 0. This can be attributed to the agent's inability to acquire any modules. In other words, none of the goals in G correspond to the real experimental situation, and the agent cannot satisfy the constraints of any modules simultaneously, resulting in no rewards.

By comparing IM3HRL with CURIOUS and MG-GE, it's evident that in the multi-module goal methods, IM3HRL can achieve the highest success rate in $19' 10^4$ episode and gradually stabilize, while CURIOUS and MG-GE need $30' 10^4$ episode or even longer. And IM3HRL exhibiting a learning speed approximately 15% faster than the optimal CURIOUS in the baseline method.

The above results highlight that IM3HRL exhibits the best performance. This can be attributed to several factors. Firstly, the CLP-guided MHA architecture enables agents to simultaneously learn multiple tasks, thereby enhancing their generalization abilities across different tasks and goals. Moreover, CLP facilitates real-time monitoring of task learning progress, promoting efficient exploration, and bolstering robustness. Furthermore, FGRS consistently employs a dynamics model to conduct rollouts based on the current policy. This allows the agent to adapt according to the current policy, diversify its goals, and naturally facilitate policy optimization, ultimately leading to a quicker attainment of the highest success rate.

F. Ablation experiments on hyperparameters and results

In order to ensure that the agent can not only achieve a certain degree of random exploration, but also actively select the module guided by CLP, we tested different values of e at 0.2, 0.3, 0.4, 0.5 to determine the optimal e.

The comparison results in Fig.12 show that increasing e will improve performance up to a point. However, with $e = 0.5$, although the agent's early training success rate improves, the curve is unstable, and overall performance remains weaker than when $e = 0.4$. This may be because increasing the probability of random exploration can enhance the learning ability of the agent to different modules. Yet, overly high e values make the agent more prone to random selection over the module selection guided by its intrinsic learning, causing training instability, and the agent cannot select the appropriate module for learning in the later stage.

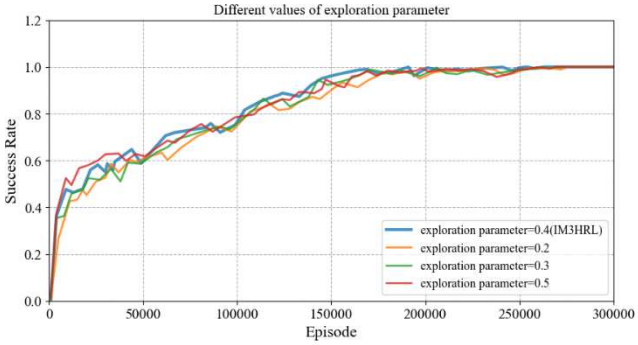


Fig. 12. The impact of different e on the performance of the agent.

For the ratio parameter x in FGRS, in order to strike a balance between preserving a certain number of original transitions and enhancing goal diversity, we select 0.6,0.7,0.8, 0.9 for ablation experiments to obtain the optimal x.

The experimental results, as shown in Fig.13, indicate that the agent's performance is optimal when $x = 0.8$. This may be because increasing x leads to more transitions for goal substitution, thus enhancing the diversity of goals. However, setting x too high can result in a shortage of original transition data, reducing the amount of raw experiential information available to the agent. This limitation can hinder the agent's ability to learn from failed experiences, ultimately leading to suboptimal performance.

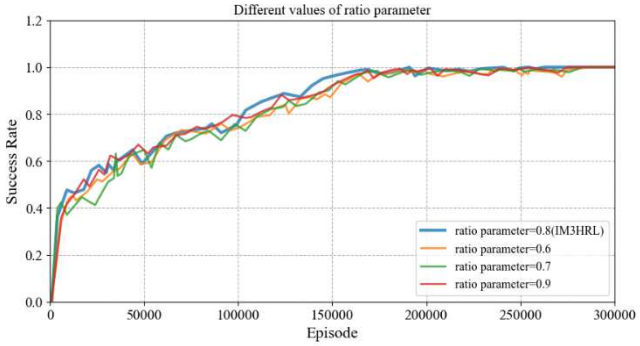


Fig. 13. The impact of different x on the performance of the agent.

VI. CONCLUSION

In our quest to further improve the performance of GCRL in complex decision-making tasks, we have introduced IM3HRL, which significantly enhances the agent's exploratory learning capabilities. This algorithm boasts several key advantages:

First, the modular hierarchical reinforcement learning architecture can simplify complex tasks and improve the exploration and learning efficiency of agents. Secondly, using the exploration learning mechanism with intrinsic motivation, the attention of the agent is biased towards the modules with high and low C. This enables the agent to prioritize modules that are feasible while also safeguarding against forgetting and perturbations. Thirdly, the integration of MBRL during the goal relabeling phase provides the agent with a more diverse set of goals. Fourthly, FGRS-driven goal relabeling is employed to optimize policies, thereby further improving the algorithm's sample efficiency.

The experimental results conducted in the modular test environment confirm the superiority of IM3HRL. In comparison to various baseline algorithms, IM3HRL has the best performance, with a learning speed at least 15% faster than other algorithms. Forgetting and perturbation experiments demonstrate that the CLP mechanism has strong robustness to forgetting and perturbation phenomena. The ablation experiments of MHA and FGRS further validate the effectiveness of the proposed improvement scheme.

For future works, Laversanne-Finot, etc. showed that disentangled representations learned by β-VAEs can be leveraged by modular curiosity-driven IMGEPs to explore as efficiently as using handcrafted low-dimensional scene features in experiments that include both controllable and distractor objects [40]. Combining IM3HRL to this unsupervised disentangled goal space learning approach is an interesting avenue to seek additional forms of intrinsic motivation to drive exploration.

In addition, we acquaint that using language to represent goals can reduce redundant information in states, and flexibly describe goals in tasks and environments, which has strong

applicability [41]. The combinatorial nature of language provides transferable features for cross-module and cross-goal generalization [42]. Therefore, a natural extension of this work can be using natural language (NL) interactions with a descriptive social partner (SP) to explore environment and interact with objects. The agent learns to represent goals by jointly training a language encoder mapping NL provided by its own exploration and SP to goal embeddings and a goal-achievement reward function, thereby extending the capabilities of IM3HRL.

REFERENCES

[1] Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research, 17(1), 1334-1373.

[2] Liu, M., Zhu, M., & Zhang, W. (2022). Goal-conditioned reinforcement learning: Problems and solutions. arXiv preprint arXiv:2201.08299.

[3] Roberts, M., Borrajo, D., Cox, M., & Yorke-Smith, N. (2018). Special issue on goal reasoning. AI Communications, 31(2), 115-116.

[4] Rolf, M., & Steil, J. J. (2013). Efficient exploratory learning of inverse kinematics on a bionic elephant trunk. IEEE transactions on neural networks and learning systems, 25(6), 1147-1160.

[5] Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015, June). Universal value function approximators. In International conference on machine learning (pp. 1312-1320). PMLR.

[6] Colas, C., Karch, T., Sigaud, O., & Oudeyer, P. Y. Intrinsically Motivated Goal-Conditioned Reinforcement Learning: A Short Survey. arXiv 2020. arXiv preprint arXiv:2012.09830.

[7] Forestier, S., Mollard, Y., and Oudeyer, P.-Y. Intrinsically motivated goal exploration processes with automatic curriculum learning. arXiv preprint arXiv:1708.02190,2017.

[8] Forestier, S., & Oudeyer, P. Y. (2016, October). Modular active curiosity-driven discovery of tool use. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 3965-3972). IEEE.

[9] Colas, C., Fournier, P., Chetouani, M., Sigaud, O., & Oudeyer, P. Y. (2019, May). Curious: intrinsically motivated modular multiple goal reinforcement learning. In International conference on machine learning (pp. 1331-1340). PMLR.

[10] Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial intelligence, 112(1-2), 181-211.

[11] Eppe, M., Gumbsch, C., Kerzel, M., Nguyen, P. D., Butz, M. V., & Wermter, S. (2022). Intelligent problem-solving as integrated hierarchical reinforcement learning. Nature Machine Intelligence, 4(1), 11-20.

[12] Kulkarni, T. D., Narasimhan, K., Saeedi, A., & Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. Advances in neural information processing systems, 29.

[13] Levy, A., Konidaris, G., Platt, R., & Saenko, K. (2017). Learning multi-level hierarchies with hindsight. arXiv preprint arXiv:1712.00948.

[14] Nachum, O., Gu, S. S., Lee, H., & Levine, S. (2018). Data-efficient hierarchical reinforcement learning. Advances in neural information processing systems, 31.

[15] Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In Advances in Neural Information Processing Systems, 2020.https://doi.org/10.48550/arXiv.2006.11485.

[16] Fu, Haotian, et al. "MGHRL: Meta goal-generation for hierarchical reinforcement learning." Distributed Artificial Intelligence: Second International Conference, DAI 2020, Nanjing, China, October 24–27, 2020, Proceedings 2. Springer International Publishing, 2020.https://doi.org/10.1007/978-3-030-64096-5_3.

[17] Hindsight experience replay. (2017). Hindsight experience replay. Advances in neural information processing systems, 30.

[18] Fang, M., Zhou, C., Shi, B., Gong, B., Xu, J., & Zhang, T. (2018, September). DHER: Hindsight experience replay for dynamic goals. In International Conference on Learning Representations.

[19] Fang, M., Zhou, T., Du, Y., Han, L., & Zhang, Z. (2019). Curriculum-guided hindsight experience replay. Advances in neural information processing systems, 32.

[20] Florensa, C., Held, D., Geng, X., & Abbeel, P. (2018, July). Automatic goal generation for reinforcement learning agents. In International conference on machine learning (pp. 1515-1528). PMLR.

[21] Ren, Z., Dong, K., Zhou, Y., Liu, Q., & Peng, J. (2019). Exploration via hindsight goal generation. Advances in Neural Information Processing Systems, 32.

[22] Sahni, H., Buckley, T., Abbeel, P., & Kuzovkin, I. (2019). Addressing sample complexity in visual tasks using her and hallucinatory gans. Advances in Neural Information Processing Systems, 32.

[23] Cangelosi, A., & Schlesinger, M. (2015). Developmental robotics: From babies to robots. MIT press.

[24] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[25] Pitis, S., Chan, H., Zhao, S., Stadie, B., & Ba, J. (2020, November). Maximum entropy gain exploration for long horizon multiple goal reinforcement learning. In International Conference on Machine Learning (pp. 7750-7761). PMLR.

[26] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014, January). Deterministic policy gradient algorithms. In International conference on machine learning (pp. 387-395). Pmlr.

[27] Lillicrap T P, Hunt J J, Pritzel A, et al. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[28] Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Hess, N., & Pascanu, R. (2017). Distral: Robust multitask reinforcement learning. Advances in neural information processing systems, 30.

[29] Espeholt, L., Soyer, H., Munos, R., et al. (2018, July). Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In International conference on machine learning (pp. 1407-1416). PMLR.

[30] Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., & Van Hasselt, H. (2019, July). Multi-task deep reinforcement learning with popart. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 3796-3803).

[31] Moerland, T. M., Broekens, J., & Jonker, C. M. (2020). Model-based reinforcement learning: A survey. arXiv preprint arXiv:2006.16712.

[32] Zhao Tingting, Kong Le, Han Yajie, Ren Dehua, Chen Yarui. Review of model-based reinforcement learning [J]. Journal of Frontiers of Computer Science and Technology, 2020,14 (06): 918-927.

[33] Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. Advances in neural information processing systems, 31.

[34] Pineda, L., Amos, B., Zhang, A., Lambert, N. O., & Calandra, R. (2021). Mbrl-lib: A modular library for model-based reinforcement learning. arXiv preprint arXiv:2104.10159.

[35] Talvitie, E. (2014, July). Model Regularization for Stable Sample Rollouts. In UAI (pp. 780-789).

[36] Asadi, K., Misra, D., & Littman, M. (2018, July). Lipschitz continuity in model-based reinforcement learning. In International Conference on Machine Learning (pp. 264-273). PMLR.

[37] Janner, M., Fu, J., Zhang, M., & Levine, S. (2019). When to trust your model: Model-based policy optimization. Advances in Neural Information Processing Systems, 32.

[38] Plappert M, Andrychowicz M, Ray A, et al. (2018). Multiple goal reinforcement learning: Challenging robotics environments and request for research. arXiv preprint arXiv:1802.09464.

[39] Nguyen, S. M., & Oudeyer, P. Y. (2012). Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner. Paladyn, 3(3), 136-146.

[40] Laversanne-Finot, A., Pere, A., & Oudeyer, P. Y. (2018, October). Curiosity driven exploration of learned disentangled goal spaces. In Conference on Robot Learning (pp. 487-504). PMLR.

[41] Luketina J, Nardelli N, Farquhar G, et al. (2019). A survey of reinforcement learning informed by natural language. arXiv preprint arXiv:1906.03926.

[42] Colas, C., Karch, T., Lair, N., Dussoux, J. M., Moulin-Frier, C., Dominey, P., & Oudeyer, P. Y. (2020). Language as a cognitive tool to imagine goals in curiosity driven exploration. Advances in Neural Information Processing Systems, 33, 3761-3774.

Wei Liu Ph.D. ,associate professor at the College of Science, Liaoning Technical University, Director of the Institute of Mathematics and System Science of the College of Science, a member of the Mathematical Planning Branch of the Chinese Operations Research Association, Senior member of the Chinese Computer Society, and member of the Chinese Artificial Intelligence Society. His research interest is the research work of machine learning, deep neural network, unmanned truck intelligent dispatching mining system engineering and other directions.

Jiaxiang Wang, received the B.S. degree in mathematics and applied mathematics from Shangqiu Normal University, Henan, China, in 2021. He is currently pursuing the M.S. degree in Liaoning Technical University. His research interests are machine learning and reinforcement learning.

Guangwei Liu, is now a professor at the College of Mines, Liaoning Technical University, Liaoning, China.