Decoupled Hierarchical Reinforcement Learning with State Abstraction for Discrete Grids

Qingyu Xiao, Yuanlin Chang, Youtian Du*

School of Automation Science and Engineering, Xi'an Jiaotong University, Xi'an, China

arXiv:2506.02050v1 [cs.LG] 1 Jun 2025

***Abstract*—Effective agent exploration remains a core challenge in reinforcement learning (RL) for complex discrete state-space environments, particularly under partial observability. This paper presents a decoupled hierarchical RL framework integrating state abstraction (DcHRL-SA) to address this issue. The proposed method employs a dual-level architecture, consisting of a high-level RL-based actor and a low-level rule-based policy, to promote effective exploration. Additionally, state abstraction method is incorporated to cluster discrete states, effectively lowering state dimensionality. Experiments conducted in two discrete customized grid environments demonstrate that the proposed approach consistently outperforms PPO in terms of exploration efficiency, convergence speed, cumulative reward, and policy stability. These results demonstrate a practical approach for integrating decoupled hierarchical policies and state abstraction in discrete grids with large-scale exploration space. Code will be available at https://github.com/XQY169/DcHRL-SA.**

***Index Terms*—Reinforcement learning, decoupled hierarchical decision-making, state abstraction, discrete grid environment.**

I. Introduction

Reinforcement learning (RL) learns optimal policies through trial-and-error interactions, demonstrating strong adaptability and long-term optimization capabilities, with wide applications in robotics, path planning, and intelligent systems [1]–[3]. Despite these advances, practical RL applications often suffer from large-scale exploration spaces and partial observability, severely restricting the efficiency of exploration and the overall performance of the policy [4], [5]. In particular, discrete state-space environments and partially observable markov decision processes (POMDPs) present additional difficulties for conventional RL algorithms. In discrete state spaces, the lack of natural similarity metrics hinders policy generalization to unvisited states. Sparse rewards further exacerbate the difficulty in extracting effective learning signals. For POMDPs, partial observability leads to compromised decisions and challenges the efficiency and stability of standard Markov-based RL algorithms.

Multiple strategies have been proposed to address these challenges, including hierarchical reinforcement learning (HRL) and state abstraction. HRL aims to decompose complex tasks into multiple subtasks, where high-level and low-level policies are responsible for sub-goal planning and action execution, respectively, thus significantly improving exploration efficiency and learning performance [6], [7]. However, most existing methods rely on two coupled RL structures, making the engineering implementation complex. To address this, some frameworks combining RL with optimization-based controllers have been proposed. For example, Watanabe et al. [8] presented a hierarchical strategy that combines RL for high-level decision making with proportional integral derivative (PID) controllers for low-level motion execution, achieving sample-efficient locomotion adaptation for quadruped robots in complex environments. Similarly, Zhang and Ramirez-Amaro [9] proposed a framework that integrates symbolic planning with low-level RL, where the planner generates task sequences for high-level strategies while RL independently learns action sequences for each subtask. This approach improves efficiency and success rates in complex robotic manipulation tasks compared to coupled hierarchical methods. Nevertheless, such hierarchical frameworks often require manually specified high-level goals. Overly coarse goals (e.g. the final destination) compromise the purpose of hierarchical RL, while overly fine-grained goals demand extensive human prior knowledge.

State abstraction reduces exploration complexity by clustering the state space while preserving critical information [10]–[12]. Deep markov decision process state abstraction (DeepMDP) [11] strictly preserves the reward and transition equivalence within the abstracted state representation. This approach has demonstrated promising results in continuous-state RL tasks. In POMDPs, agents must learn policies based on historical observations or latent state estimations. By introducing belief distributions, a POMDP can theoretically be transformed into a Markov decision process (MDP) in the belief space [13]. However, the state space complexity grows exponentially with state dimensionality. Some works mitigate this by incorporating recurrent neural networks such as RNNs and LSTMs into deep reinforcement learning frameworks, encoding historical observations to improve performance in POMDPs [14], [15]. Moreover, [16] extended state abstraction techniques to historical observation sequences and achieved improved performance in continuous state-space environments. However, without sufficient exploration, state abstraction tends to degenerate into trivial representations [11].

This paper proposes a decoupled hierarchical RL framework integrating state abstraction (DcHRL-SA) for discrete grid environments to address the challenges posed by large-scale exploration spaces and partial observability. The framework designs reasonable high-level goals while employing simple rule-based control for low-level execution. An action mask

* Corresponding author: duyt@xjtu.edu.cn

This work was supported by the National Key Research and Development Program of China under Grant 2021YFB2400800.

mechanism is incorporated to restrict invalid actions and improve exploration efficiency [17]. Furthermore, it is theoretically proven that the decoupled hierarchical mechanism preserves the existence of an optimal policy when the discount factor γ is set to 1. The framework further integrates DeepMDP abstraction to compress the state space and enhance both training efficiency and policy robustness. Experimental results demonstrate that the proposed approach consistently outperforms the baseline PPO algorithm in cumulative rewards and convergence efficiency in MDPs and POMDPs, providing a practical and effective solution for RL in complex discrete grid environments.

II. Methodology

A. DcHRL-SA Framework

The decision-making process of the proposed framework in POMDP environments is illustrated in Fig. 1. First, the environment continuously provides a historical observation-goal sequence, denoted as h_t, which is formally defined as:

$$h_t = \{o_{t-l}, g_{t-l}, \ldots, o_{t-1}, g_{t-1}, o_t\} \tag{1}$$

where o_i and g_i denote the observation and sub-goal at time i, respectively, and l is the history length. This sequence is processed by the state abstraction module through the abstraction mapping function $\phi(\cdot)$, producing the latent state representation z_t at the current timestep. Subsequently, a high-level actor, implemented based on the PPO algorithm, takes z_t as input and generates a new goal g_t for the current decision step. The generated goal g_t, together with the current observation o_t, is then passed to a rule-based low-level policy, which outputs an action sequence $\{a_t^0, a_t^1, \ldots, a_t^m\}$ required to achieve the goal. The environment executes this action sequence until completion and transitions to a new observation state. Afterward, the environment provides updated observation information, and the decision-making process proceeds to the next iteration. In MDP environments, since the agent has full observability, the state s_t is directly passed into the state abstraction function to obtain the abstract state z_t, without additional inference steps.

B. DeepMDP Abstraction in POMDPs

State abstraction methods aim to reduce the dimensionality of state spaces by mapping states with similar dynamic behaviors into a unified abstract space. The DeepMDP framework leverages end-to-end neural network models to learn an abstract mapping function $z = \phi(s)$.

A MDP typically consists of a state set S, an action set A, a state transition function $P : S \times A \times S \to [0, 1]$, and a reward function $R : S \times A \to \mathbb{R}$. A POMDP additionally includes an observation space X and an observation function $O : S \times A \times X \to [0, 1]$, where $O(s', a, x) = P(x \mid a, s')$ denotes the probability of receiving an observation x after executing action a and transitioning to state s'.

Under the assumption of deterministic transitions, where each state-action pair (s_t, a_t) uniquely determines the next state $s_{t+1} = P(s_t, a_t)$, the DeepMDP constraint is formally expressed as:

$$\begin{aligned}\forall s_1, s_2,\ \phi(s_1) = \phi(s_2),\ \forall a \in A, , \\ R(s_1, a) = R(s_2, a), \phi(P(s_1, a)) = \phi(P(s_2, a))\end{aligned} \tag{2}$$

During training, DeepMDP jointly optimizes the abstract mapping, reward prediction, and transition prediction networks by minimizing the following modeling loss:

$$\begin{aligned}L_{\text{bisim}}(\theta_1, \theta_2, \theta_3) = \mathbb{E}_{s,s',a,r}\Big[\|R_{\theta_1}(\phi_{\theta_3}(s), a) - r\|_2^2 \\ + \lambda \|P_{\theta_2}(\phi_{\theta_3}(s), a) - \phi_{\theta_3}(s')\|_2^2\Big]\end{aligned} \tag{3}$$

An observation-action history sequence for POMDPs is defined as $h_t = \{o_{t-l}, a_{t-l}, \ldots, o_{t-1}, a_{t-1}, o_t\}$. Based on this, we construct an extended state space $\mathcal{H}$.

To effectively capture temporal dependencies in the observation sequences, a LSTM module is incorporated into the abstraction mapping function $\phi(\cdot)$ to encode the dynamic information within the history sequence through its hidden state.

Building upon this encoding, the DeepMDP constraint is applied to the observation–action histories, with the corresponding loss function defined as:

$$\begin{aligned}L_{\text{bisim}}(\theta_1, \theta_2, \theta_3) = \mathbb{E}_{h,h',a,r}\Big[\|R_{\theta_1}(\phi_{\theta_3}(h), a) - r\|_2^2 \\ + \lambda \|P_{\theta_2}(\phi_{\theta_3}(h), a) - \phi_{\theta_3}(h')\|_2^2\Big]\end{aligned} \tag{4}$$

This approach enables the model to extract DeepMDP-consistent structural information from historical observation sequences while incorporating data augmentation strategies inspired by [18] to improve the robustness of the abstraction model. As a result, it enhances the state abstraction capability under partial observability and improves both policy convergence speed and training stability.

C. Decoupled Hierarchical Strategy

The proposed DcHRL-SA framework follows the goal-based HRL paradigm proposed in [6], where a sub-goal set G is defined to guide hierarchical decision-making. The high-level actor $\pi^H : S \to G$ selects a sub-goal $g \in G$ based on the current state $s \in S$, while the low-level policy $\pi^L : S \times G \to A$ determines the action $a \in A$ conditioned on both the current state s and the selected sub-goal g. In this framework, sub-goals typically correspond to sub-regions of the state space S or specific target states. The low-level policy generates action sequences to accomplish the given sub-goal. The objective of HRL is to maximize the cumulative reward under the joint policy:

$$J(\pi^H, \pi^L) = \mathbb{E}_{\pi^H, \pi^L}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{5}$$

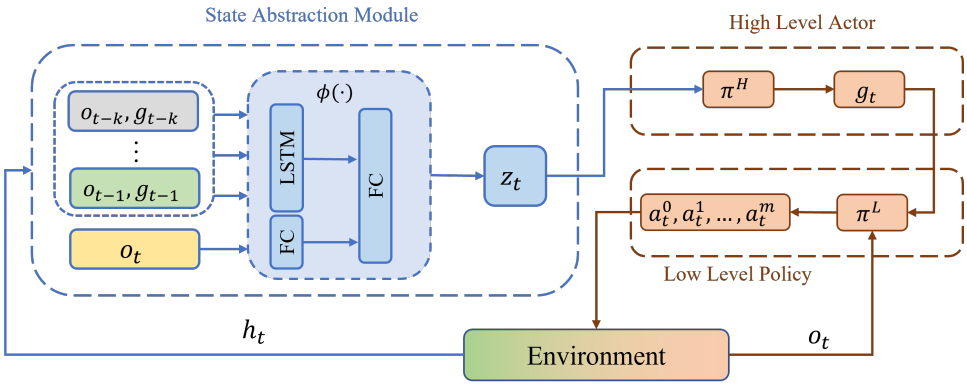


Fig. 1. Overview of the Combined Framework for Inference in POMDPs.

Considering the characteristics of discrete grid environments in terms of observations and action spaces, this work defines sub-goals as composite actions, which are formed by combining primitive actions from the action set. This design ensures the stability and reliability of the low-level policy's execution. For instance, in a modified *DoorKey* environment, the agent perceives its surroundings through a $\mathcal{W} \times \mathcal{W}$ local observation window centered on itself, where $\mathcal{W}$ represents the size of the observation window. The basic action space consists of five discrete actions: moving up, down, left, right, and an interaction action. Composite actions can be categorized into two types: (1) *Move*, which involves a sequence of movement actions, and (2) *Move and interact*, which consists of a sequence of movements followed by performing the interaction at the final location. This work extends the goal space into position–action type pairs, including:

- Target position: one of the $\mathcal{W} \times \mathcal{W}$ discrete positions within the local observation.
- Action type: either *Move* or *Move and interact*.

Thus, the goal space for the high-level actor is expanded to $2 \times \mathcal{W} \times \mathcal{W}$ dimensions, specifically defined as:

- The first $\mathcal{W} \times \mathcal{W}$ actions: selecting a target position and moving to that location.
- The remaining $\mathcal{W} \times \mathcal{W}$ actions: selecting a target position and executing an interaction upon arrival.

The low-level policy dynamically generates the optimal movement action sequence from the current position to the specified target position based on the current observation o_t and the sub-goal provided by the high-level actor, and performs the corresponding operation if required at the destination.

Formally, the high-level action space is defined as:

$$G = \left\{ g_{i,j,k} \;\middle|\; (i,j) \in \mathcal{O},\ k \in \{0,1\} \right\} \tag{6}$$

where (i,j) denotes the target position within the observation space $\mathcal{O}$, $k = 0$ represents *Move*, and $k = 1$ denotes *Move and interact*.

To further enhance decision-making efficiency, this work introduces an action mask mechanism to filter out invalid sub-goals based on environmental constraints, thereby avoiding ineffective exploration. Specifically, a validity-checking function $\mathcal{F}(o, g)$ is defined to determine whether a sub-goal g is valid given the current observation state o. The validity rules are as follows:

- For *Move*: if the target position $(i,j) \in \mathcal{O}$ is unreachable in the current observation, then: $\mathcal{F}(o, g_{i,j,0}) = 0$.
- For *Move and interact*: if the target position $(i,j) \in \mathcal{O}$ does not contain any interactive object, then: $\mathcal{F}(o, g_{i,j,1}) = 0$.

The set of valid sub-goals is defined as:

$$G_{\text{valid}}(o) = \left\{ g \in G \;\middle|\; \mathcal{F}(o,g) = 1 \right\} \tag{7}$$

Finally, the agent samples actions only within $G_{\text{valid}}(o)$, thereby improving exploration efficiency and accelerating policy convergence. Moreover, since the hierarchical goal space fully covers the basic action space, it can be theoretically shown that the existence of an optimal policy is preserved when the discount factor γ is set to 1, as formally proven in Appendix.

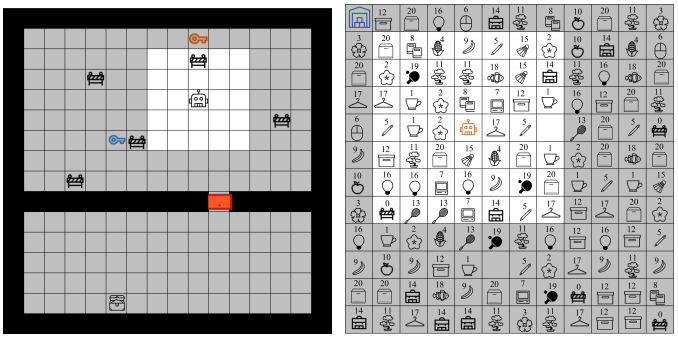
III. Experimental Results

This study conducts experiments on two discrete grid environments to evaluate the performance of the proposed method under varying task structures and observation conditions. The layouts of two environments are illustrated in Fig. 2.

A. Environment

1) Customized MiniGrid-DoorKey-16x16: This environment is based on the DoorKey scenario from the MiniGrid platform [19], extended to a 16×16 grid. The map is divided by a wall into Room A and Room B. The agent starts in Room A, where two randomly placed keys are available, but only one can unlock the door to Room B. The agent needs to pick up the correct key, open the door, and finally reach the treasure chest to receive rewards.

- Observation Space: A 5×5 egocentric view and inventory information.
- Action Space: Five discrete actions: {*up, down, left, right, interact*}.



(a) Customized DoorKey (b) Multi-Item collection grid

Fig. 2. Visualizations of the discrete grid environments used in experiments.

- Reward Structure:
  - +1 for opening the door.
  - -0.01 penalty for colliding with walls.
  - $5 - 3 \times (\frac{\text{total_steps}}{\text{max_steps}})$ for reaching the treasure chest.

2) Multi-Item Collection Grid: This environment is designed as a 12×12 grid-world with 21 types of randomly distributed items. There is one warehouse in the position (0,0). The agent collects items into a backpack with unlimited capacity, and can submit four identical items at the warehouse to gain rewards.

- Observation Space:
  - *MDP mode:* Full map view and inventory.
  - *POMDP mode:* A 7×7 egocentric view and inventory information.
- Action Space: Five discrete actions: {*up, down, left, right, interact*}.
- Reward Structure:
  - +3 for successfully submitting a group of four identical items.
  - $-(0.1 + \frac{\text{carry_number}}{144})$ movement penalty based on the number of items carried.
  - +100 bonus for completing all submission tasks within the step limit.
  - $-(140 - \text{submit_number})$ for incomplete tasks.

B. Architecture and Training

Table I details the key hyperparameter settings adopted in different environments. These hyperparameters were tuned through preliminary experiments to ensure the stability of the policy learning process and the effectiveness of state abstraction representations.

The hyperparameters used in the experiments are described as follows. lr denotes the learning rate. γ is the discount factor for future rewards. λ represents the coefficient used in Generalized Advantage Estimation (GAE). Bs indicates the training batch size. l refers to the length of the historical observation sequence employed in POMDPs. $dim(Z)$ specifies the dimensionality of the abstract state space after state abstraction. TF represents the training frequency per training step. The parameters $P_{agent}, P_{door}, P_{key}, P_{wall}$, and $P_{treasure}$ denote the initial position of the agent, door, key, wall, and treasure, respectively.

TABLE I
KEY HYPERPARAMETER SETTINGS

Module	Hyperparameter	DoorKey-16x16	Multi-Item Grid
PPO&LSTM	Optimizer	Adam	Adam
	lr	0.0001	0.0001
	γ	0.997	0.997
	λ	0.95	0.95
	ϵ	0.2	0.2
	Bs	256	256
	l	15	15
State Abstraction	$dim(Z)$	60	25(MDP) 40(POMDP)
	Bs	384	384
	TF	30	30
Randomness	Environment	P_{agent}, P_{door} P_{key}, P_{wall} $P_{treasure}$	P_{agent}

C. Results

To verify the effectiveness of the proposed method, comparative experiments were conducted in the customized DoorKey environment and multi-item collection grid environment. The experiments compared three methods: the baseline PPO, decoupled hierarchical RL (DcHRL), and DcHRL-SA. All experiments were performed on an NVIDIA RTX 4090 GPU platform under consistent hyperparameter settings to ensure fairness.

The score curve for the customized DoorKey environment are illustrated in Fig. 3. Quantitative results are summarized in Table II, confirming the advantages of hierarchical decision-making and the potential of state abstraction in discrete grid environments. Compared with the baseline PPO, the proposed DcHRL framework significantly improves the final score and accelerates policy convergence by leveraging hierarchical exploration strategies. Moreover, the DcHRL-SA method, which integrates DeepMDP abstraction, slightly improves the final

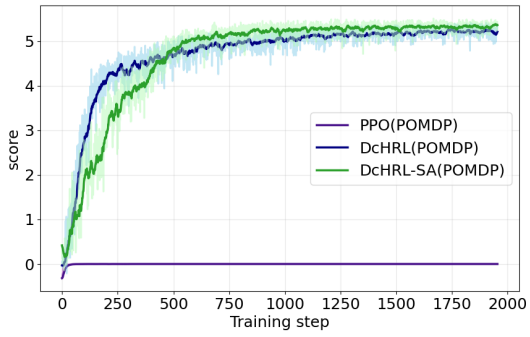


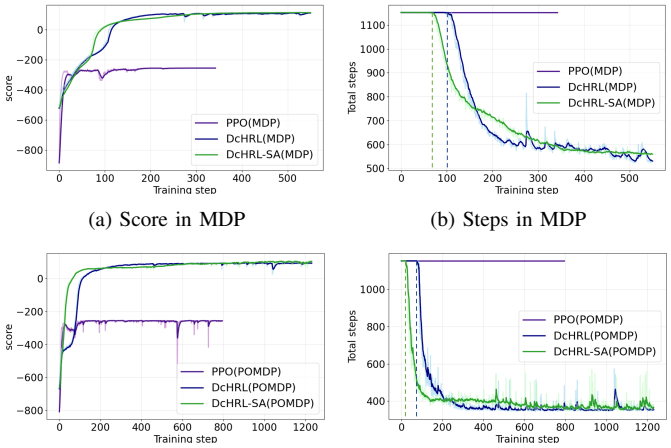
Fig. 3. Score curves of different methods in the MiniGrid DoorKey environment.

TABLE II
PERFORMANCE COMPARISON IN MINIGRID DOORKEY ENVIRONMENT

Method	Final Reward	State Dimension
PPO	0.0000 ± 0.0000	15×26
DcHRL	5.2158 ± 0.0981	15×26
DcHRL-SA	5.3207 ± 0.0849	60

TABLE III
PERFORMANCE COMPARISON IN CUSTOM GRID ENVIRONMENTS (MDP & POMDP MODES)

Modes	Method	Converged Score	Total steps
MDP	PPO	-263.81 ± 0.00	1152
	DcHRL	108.40 ± 3.41	547.62 ± 14.38
	DcHRL-SA	114.10 ± 0.80	561.87 ± 4.74
POMDP	PPO	-267.18 ± 33.27	1152
	DcHRL	90.33 ± 5.49	363.13 ± 16.96
	DcHRL-SA	100.95 ± 4.61	369.70 ± 24.17



(a) Score in MDP (b) Steps in MDP

(c) Score in POMDP (d) Steps in POMDP

Fig. 4. Training performance under different methods in MDP and POMDP environments.

score while effectively reducing the state space dimension from 15×26 to 60.

In the multi-item collection grid environment, similar performance trends are observed. The score curves and task completion step curves for both MDP and POMDP modes are illustrated in Fig. 4, and the corresponding quantitative results are summarized in Table III. In terms of score performance, the baseline PPO algorithm demonstrates poor exploration capability in both environments, quickly converging to local optima. In contrast, the DcHRL framework substantially improves exploration efficiency and final performance in both MDP and POMDP modes. Furthermore, the DcHRL-SA framework achieves additional gains in final converged scores while effectively reducing the state dimensionality.

Regarding the task completion steps, the step curves of DcHRL-SA start to decline earlier than those of DcHRL as additionally marked in Fig. 4. DcHRL-SA exhibits smoother and more stable convergence curves than DcHRL in the MDP mode, indicating enhanced optimization stability. In the POMDP mode, while DcHRL-SA converges slightly faster than DcHRL, its step curves show increased oscillation near convergence, where the incompleteness of information makes it difficult to converge to a stable abstraction function.

Overall, these results validate the effectiveness of DcHRL-SA framework, enhancing both exploration efficiency and convergence speed in discrete grid environments under varying observability conditions.

IV. CONCLUSION

This paper proposes a decoupled hierarchical reinforcement learning framework combined with a DeepMDP abstraction method, aiming to improve decision-making efficiency and exploration capability in sparse-reward discrete grid environments. Experimental results on the customized DoorKey environment and a multi-item collection grid environment demonstrate that the decoupled hierarchical policy outperforms the PPO baseline in both convergence speed and final performance, validating the effectiveness of hierarchical decision structures in improving exploration in complex discrete environments. Although the introduction of the DeepMDP abstraction mechanism results in limited improvement in final scores, experiments show that it successfully captures essential decision-related information and exhibits a clear advantage in convergence speed in both MDPs and POMDPs. This confirms the potential of state abstraction methods in enhancing training efficiency and model scalability. Overall, this work provides practical guidance and methodological references for future research on optimizing hierarchical policies and state abstraction techniques.

REFERENCES

[1] G. Sartoretti et al., "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019. doi: 10.1109/LRA.2019.2903261.

[2] A. Agrawal, S. J. Won, T. Sharma, M. Deshpande, and C. McComb, "A multi-agent reinforcement learning framework for intelligent manufacturing with autonomous mobile robots," *Proc. Design Society*, vol. 1, pp. 161–170, Jul. 2021. doi: 10.1017/pds.2021.17.

[3] Z. Yan and Y. Xu, "A multi-agent deep reinforcement learning method for cooperative load frequency control of multi-area power systems," *IEEE Trans. Power Syst.*, early access, 2020. doi: 10.1109/TPWRS.2020.2999890.

[4] D. Ghosh, J. Rahme, A. Kumar, A. Zhang, R. P. Adams, and S. Levine, "Why generalization in RL is difficult: Epistemic POMDPs and implicit partial observability," in *Proc. NeurIPS*, 2021.

[5] C. P. Andriotis and K. G. Papakonstantinou, "Managing engineering systems with large state and action spaces through deep reinforcement learning," *Reliab. Eng. Syst. Saf.*, Apr. 2019. doi: 10.1016/j.ress.2019.04.036.

[6] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. NeurIPS*, vol. 29, pp. 3682–3690, Dec. 2016.

[7] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, Feb. 2017. doi: 10.1609/aaai.v31i1.10916.

[8] T. Watanabe et al., "Hierarchical reinforcement learning with central pattern generator for enabling a quadruped robot simulator to walk on a variety of terrains," *Sci. Rep.*, vol. 15, no. 1, Apr. 2025. doi: 10.1038/s41598-025-94163-2.

[9] J. Zhang and K. Ramirez-Amaro, "Hierarchical reinforcement learning based on planning operators," *arXiv preprint arXiv:2309.14237*, Jan. 2023. doi: 10.48550/arXiv.2309.14237.
[10] L. Li, T. J. Walsh, and M. L. Littman, "Towards a unified theory of state abstraction for MDPs," in *Proc. ISAIM*, Jan. 2006.
[11] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare, "DeepMDP: Learning continuous latent space models for representation learning," in *Proc. ICML*, pp. 2170–2179, May 2019.
[12] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," *arXiv preprint arXiv:2006.10742*, 2020.
[13] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Proc. NeurIPS*, vol. 23, pp. 2164–2172, Dec. 2010.
[14] L. Meng, R. Gorbet, and D. Kulic, "Memory-based deep reinforcement learning for POMDPs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 5619–5626, Sep. 2021. doi: 10.1109/IROS51168.2021.9636140.
[15] T. Ni, B. Eysenbach, and R. Salakhutdinov, "Recurrent model-free RL can be a strong baseline for many POMDPs," *arXiv preprint arXiv:2110.05038*, 2021.
[16] J. Subramanian, A. Sinha, R. Seraj, and A. Mahajan, "Approximate information state for approximate planning and reinforcement learning in partially observed systems," *arXiv preprint arXiv:2010.08843*, 2020. doi: 10.48550/arXiv.2010.08843.
[17] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," in *Proc. Int. FLAIRS Conf.*, vol. 35, May 2022. doi: 10.32473/flairs.v35i.130584.
[18] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 19884–19895, Dec. 2020.
[19] M. Chevalier-Boisvert et al., "Minigrid and Miniworld: Modular and customizable reinforcement learning environments for goal-oriented tasks," *arXiv preprint arXiv:2306.13831*, Jun. 2023. Accessed: Jul. 8, 2024. [Online]. Available: https://arxiv.org/abs/2306.13831.

APPENDIX

Proof. The notations used in this proof are summarized as follows:

- $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$: the finite set of n discrete environment states.
- $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$: the finite set of m possible actions.
- $\mathbf{u}_t^\pi \in \mathbb{R}^{n\times 1}$: the state distribution vector at time t under policy π, where the i-th entry $u_t^\pi(s_i)$ represents the probability of being in state s_i at time t.
- $\mathbf{P}^\pi \in \mathbb{R}^{n\times n}$: the state transition probability matrix under policy π, where the (i,j)-th entry $p^\pi(s_i|s_j)$ represents the probability of transitioning from state s_j to state s_i.
- $\pi(a|s)$: the probability of selecting action a when in state s under policy π.
- $R(s,a)$: the immediate reward obtained by taking action a in state s.
- $\mathbf{R}^\pi \in \mathbb{R}^{n\times 1}$: the expected immediate reward vector under policy π.
- $\mathbf{V}^\pi \in \mathbb{R}^{n\times 1}$: the state value function vector under policy π, where the i-th entry $V^\pi(s_i)$ represents the expected cumulative discounted reward starting from state s_i.
- $\gamma \in [0,1]$: the discount factor for future rewards.
- $J^\pi(\mathbf{u}_t^\pi)$: the expected cumulative reward at time t under policy π.

Accordingly, for a given policy π, the corresponding state probability transition trajectory can be denoted as $\mathbf{u}_0 \to \mathbf{u}_1^\pi \to \mathbf{u}_2^\pi \to \cdots \to \mathbf{u}_e$, where $\mathbf{u}_e$ represents the desired target distribution, which is independent of the policy π.

If the state distributions at two time steps along the trajectory, $\mathbf{u}_i^\pi$ and $\mathbf{u}_j^\pi$ $(i \le j)$, are known, the expected cumulative reward when transitioning from $\mathbf{u}_i^\pi$ to $\mathbf{u}_j^\pi$ under policy π is given by:

$$d_\pi(\mathbf{u}_i^\pi, \mathbf{u}_j^\pi) = \sum_{t=i}^{j-1} \gamma^{t-i} \left(\mathbf{u}_t^\pi\right)^T \mathbf{R}^\pi \tag{A.1}$$

which depends only on the policy π and the two state distributions $\mathbf{u}_i^\pi$ and $\mathbf{u}_j^\pi$.

The optimal policy π^* is then defined as:

$$\pi^* = \arg\max_\pi d_\pi(\mathbf{u}_0, \mathbf{u}_e) \tag{A.2}$$

Notably, given an intermediate point $\mathbf{u}_m$ on the optimal trajectory, it holds that:

$$d_{\pi^*}(\mathbf{u}_0, \mathbf{u}_e) = d_{\pi^*}(\mathbf{u}_0, \mathbf{u}_m) + \gamma^m d_{\pi^*}(\mathbf{u}_m, \mathbf{u}_e) \tag{A.3}$$

Particularly when $\gamma = 1$, it simplifies to:

$$d_{\pi^*}(\mathbf{u}_0, \mathbf{u}_e) = d_{\pi^*}(\mathbf{u}_0, \mathbf{u}_m) + d_{\pi^*}(\mathbf{u}_m, \mathbf{u}_e) \tag{A.4}$$

Therefore, the optimal policy π^* is also the optimal policy for both sub-problems $\mathbf{u}_0 \to \mathbf{u}_m$ and $\mathbf{u}_m \to \mathbf{u}_e$ independently.

When the precise intermediate point on the optimal path is not known, but only an abstract set G containing possible intermediate states $\mathbf{g}$, the problem can be expressed as:

$$\begin{aligned}\max_\pi d_\pi(\mathbf{u}_0, \mathbf{u}_e) = &\max_{\pi_1,\pi_2,\mathbf{g}} \Big(d_{\pi_1}(\mathbf{u}_0, \mathbf{g}) + d_{\pi_2}(\mathbf{g}, \mathbf{u}_e)\Big) \\ = &\max_{\mathbf{g}_1} \Big(\max_{\pi_1} d_{\pi_1}(\mathbf{u}_0, \mathbf{g}_1) + \max_{\mathbf{g}_2} \big(\\ &\max_{\pi_2} d_{\pi_2}(\mathbf{g}_1, \mathbf{g}_2) + \max_{\pi_3} d_{\pi_3}(\mathbf{g}_2, \mathbf{u}_e)\big)\Big)\end{aligned} \tag{A.5}$$

When $\mathbf{g}_1$ equals to an optimal point u_1^* on the optimal path:

$$\begin{aligned}\max_\pi d_\pi(\mathbf{u}_0, \mathbf{u}_e) = \max_{\pi_1} d_{\pi_1}(\mathbf{u}_0, \mathbf{u}_1^*) + \\ \max_{\mathbf{g}_2} \Big(\max_{\pi_2} d_{\pi_2}(\mathbf{u}_1^*, \mathbf{g}_2) + \max_{\pi_3} d_{\pi_3}(\mathbf{g}_2, \mathbf{u}_e)\Big)\end{aligned} \tag{A.6}$$

Recursively, when each $\mathbf{g}_i$ equals to the corresponding optimal point u_i^*, we obtain:

$$\begin{aligned}d_{\pi^*}(\mathbf{u}_0, \mathbf{u}_e) = &\max_{\pi_1} d_{\pi_1}(\mathbf{u}_0, \mathbf{u}_1^*) + \max_{\pi_2} d_{\pi_2}(\mathbf{u}_1^*, \mathbf{u}_2^*) \\ &+ \cdots + \max_{\pi_m} d_{\pi_m}(\mathbf{u}_m^*, \mathbf{u}_e)\end{aligned} \tag{A.7}$$

Finally, since the proposed method's goal space fully covers the base action space, it can degenerate into a flat decision-making process under worst-case conditions, thereby ensuring policy optimality. □