Journal of Machine Learning Research 26 (2025) 1-55 Submitted 7/24; Revised 5/25; Published 7/25

Hierarchical Decision Making Based on Structural Information Principles

Xianghua Zeng ZENGXIANGHUA@BUAA.EDU.CN
School of Computer Science and Engineering
Beihang University
Beijing, 100191, China

Hao Peng PENGHAO@BUAA.EDU.CN
School of Cyber Science and Technology
Beihang University
Beijing, 100191, China

Dingli Su SUDINGLI@BUAA.EDU.CN
School of Computer Science and Engineering
Beihang University
Beijing, 100191, China

Angsheng Li ANGSHENG@BUAA.EDU.CN
School of Computer Science and Engineering
Beihang University
Beijing, 100191, China

Editor: George Konidaris

Abstract

Hierarchical Reinforcement Learning (HRL) is a promising approach for managing task complexity across multiple levels of abstraction and accelerating long-horizon agent exploration. However, the effectiveness of hierarchical policies heavily depends on prior knowledge and manual assumptions about skill definitions and task decomposition. In this paper, we propose a novel **S**tructural **I**nformation principles-based framework, namely **SIDM**, for hierarchical **D**ecision **M**aking in both single-agent and multi-agent scenarios. Central to our work is the utilization of structural information embedded in the decision-making process to adaptively and dynamically discover and learn hierarchical policies through environmental abstractions. Specifically, we present an abstraction mechanism that processes historical state-action trajectories to construct abstract representations of states and actions. We define and optimize directed structural entropy—a metric quantifying the uncertainty in transition dynamics between abstract states—to discover skills that capture key transition patterns in RL environments. Building on these findings, we develop a skill-based learning method for single-agent scenarios and a role-based collaboration method for multi-agent scenarios, both of which can flexibly integrate various underlying algorithms for enhanced performance. Extensive evaluations on challenging benchmarks demonstrate that our framework significantly and consistently outperforms state-of-the-art baselines, improving the effectiveness, efficiency, and stability of policy learning by up to 32.70%, 64.86%, and 88.26%, respectively, as measured by average rewards, convergence timesteps, and standard deviations.

Keywords: Hierarchical reinforcement learning, Structural information principles, State and action abstractions, Skill-based learning, Role-based learning

1. Introduction

Reinforcement learning (RL) (Sutton et al., 1998) enables agents to develop optimal decision-making strategies by interacting with their environment to solve sequential tasks with goal-oriented objectives. The integration of deep neural networks (LeCun et al., 2015; Schmidhuber, 2015) with RL has demonstrated remarkable success in diverse applications, including game intelligence (Vinyals et al., 2019; Zhou et al., 2023; Zhong et al., 2024), video acceleration (Ramos et al., 2022), and model fitting (Truong et al., 2022). However, RL algorithms face the critical challenge of requiring extensive interactions with complex environments to learn effective policies (Mattes et al., 2024).

Hierarchical reinforcement learning (HRL) offers a promising approach to improving sample efficiency by structuring agent exploration and decision-making across multiple levels of abstraction (Merel et al., 2019; Marino et al., 2018). HRL decomposes long-horizon tasks into subtasks, enabling hierarchical policies to operate at different temporal and abstraction levels, thereby accelerating policy learning (Hafner et al., 2022). However, many existing HRL approaches depend on prior knowledge, incorporating handcrafted assumptions about skill definitions (Lee et al., 2019, 2020b) or manually designed task decomposition heuristics (Tessler et al., 2017). The HSD-3 framework (Gehring et al., 2021) autonomously learns a skill hierarchy during a pre-training stage without reward supervision, but it still relies on manually selected features and predefined target objectives. Reskill (Rana et al., 2023) leverages state-conditioned generative models to construct a skill space but depends on expert-defined manipulation tasks for collecting demonstration data. The CEO framework (Machado et al., 2023), built on successor representation (Dayan, 1993), effectively discovers meaningful skills but introduces computational complexity and is sensitive to the skill scale. In multi-agent reinforcement learning (MARL), role-based task decomposition (Wang et al., 2020, 2021b) has proven effective in facilitating hierarchical collaborative strategies. However, its success heavily depends on domain-specific task knowledge and is highly sensitive to role discovery parameters. Therefore, developing an effective and stable hierarchical decision-making framework that operates without prior knowledge remains a critical challenge in advancing scalable and generalizable reinforcement learning.

To address this, we draw inspiration from structural information principles (Li and Pan, 2016), which provide a foundation for adaptive hierarchy discovery. Structural entropy measures the uncertainty in an undirected graph's dynamics by quantifying the number of bits required to encode a vertex transition during a single-step random walk. Minimizing this structural uncertainty yields a hierarchical partitioning of graph vertices[1], referred to as the encoding tree. Within this tree, each node corresponds to a subset of vertices, termed a 'community', where vertices exhibit stronger intraconnections than interconnections. In this work, we leverage the structural information embedded in the decision-making process to discover and learn hierarchical policies dynamically through environmental abstractions.

To this end, we propose a novel **S**tructural **I**nformation principles-based hierarchical **D**ecision **M**aking framework, called **SIDM**, to address the reliance on prior knowledge

1. A vertex is defined in the graph and a node in the tree.

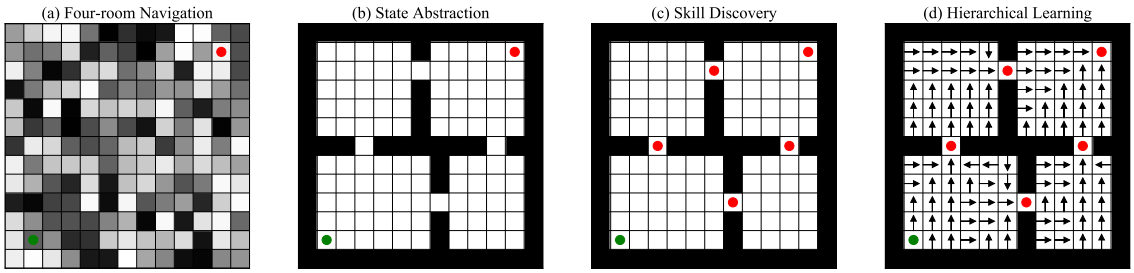


Figure 1: Illustrative single-agent navigation task within the gridworld benchmark, where the agent navigates from the start location (green) to the goal (red) while interacting with its environment.

and manual design in HRL. First, we introduce an adaptive abstraction mechanism that extracts meaningful structure from high-dimensional, noisy state-action information, producing compact abstract representations of states and actions. This mechanism leverages encoding trees to cluster states or actions with similar features into communities dynamically and applies an aggregation function to compute their abstract representations. Second, we formally define and optimize directed structural entropy to extend structural information principles beyond undirected graphs, enabling the modeling of asymmetric abstract state transitions. Using directed entropy, we quantify transition probabilities between abstract states and identify high-frequency transitions between abstract communities as skills, capturing key transition patterns in RL environments. Third, we build on the discovered skills and abstract actions to develop a skill-based method for single-agent learning and a role-based strategy for multi-agent collaboration. These methods operate independently of manual assistance and can flexibly integrate various underlying algorithms to enhance their performance. Finally, we conduct extensive experiments and comprehensive analysis on well-established benchmarks, including visual gridworld navigation, continuous robotic control, and StarCraft II micro-management. Comparative results demonstrate that, compared to state-of-the-art baselines, our framework improves average reward (effectiveness) and sampling efficiency (efficiency) by up to 32.70% and 64.86%, respectively, while reducing standard deviation (stability) by up to 88.26%.

Figure 1 shows a single-agent navigation task in the four-room domain, where the agent moves between rooms to reach a designated target. As shown in Figure 1(a), the agent receives high-dimensional, noisy visual inputs (e.g., from a camera or sensor) and navigates from the green starting position to the red target location. As illustrated in Figure 1(b), the SIDM framework extracts structural relationships from raw observations to generate abstract state representations in a lower-dimensional space, approximating the original 2D coordinates and thereby simplifying decision-making. By computing and optimizing structural entropy in directed abstract transitions, the SIDM framework adaptively identifies key abstract states—such as turning points between rooms in the gridworld (Figure 1(c)). By modeling navigation behaviors between these key states as skills, SIDM enables more efficient decision-making within each corresponding subspace, as shown in Figure 1(d).

This paper is organized as follows: Section 2 outlines the preliminaries and notations, Section 3 discusses the related work, Section 4 presents the detailed designs of SIDM framework, Sections 5 and 6 describe the experimental setups and evaluations, followed by the conclusion in Section 7.

2. Preliminaries and Notations

This section establishes the foundational concepts of reinforcement learning and structural information principles. We distinguish between "primitive states and actions," which represent the original variables provided by the environment, and "abstract states and actions," which represent higher-level abstractions derived from them. Spaces (denoted by calligraphic fonts, e.g., $\mathcal{S}, \mathcal{A}$) represent the complete sets of possible states or actions. Variables (denoted by capital letters, e.g., S, A) represent random variables over states or actions, often sampled from the replay buffer in practical implementations. Values (denoted by lowercase letters, e.g., s, a) represent specific instances of states or actions within their respective spaces. A summary of the primary notations and their detailed descriptions is provided in Appendix A.1.

2.1 Reinforcement Learning

Reinforcement learning (RL) is a learning paradigm in which one or more agents learn to make sequential decisions by interacting with an environment whose dynamics may be partially or fully unknown, with the objective of maximizing expected cumulative rewards.

2.1.1 Markov Decision Process

In RL, the single-agent decision-making problem is formulated as a **Markov decision process** (MDP) (Bellman, 1957), which is formally defined as a tuple:

$$\mathcal{M}_s = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle, \tag{1}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function mapping state-action pairs to expected rewards, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function defining the probability distribution over next states, and $\gamma \in [0, 1)$ is the discount factor that determines the weighting of future rewards.

The notation $\Delta(\mathcal{S})$ refers to the space of probability distributions over the state space $\mathcal{S}$. Here, the subscript s in $\mathcal{M}_s$ explicitly denotes a single-agent MDP, distinguishing it from the multi-agent Markov game $\mathcal{M}_m$ introduced later.

At each timestep t, the agent observes the current state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ according to its policy, i.e., $a_t \sim \pi(s_t)$, where $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ defines a probability distribution over actions conditioned on the current state. The action a_t leads to a new state s_{t+1}, sampled from the transition distribution, i.e., $s_{t+1} \sim \mathcal{P}(s_t, a_t)$, and the agent receives a reward $r_t \sim \mathcal{R}(s_t, a_t) \in \mathbb{R}$.

The agent aims to learn an optimal policy $\pi^* : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected cumulative discounted reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\mathcal{P},\pi} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \right]. \tag{2}$$

2.1.2 Markov Game

A fully cooperative multi-agent task, where all agents aim to maximize a shared reward, is modeled as a **Markov game** (Littman, 1994). It is formally defined as a tuple:

$$\mathcal{M}_m = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle, \tag{3}$$

where $\mathcal{N} \equiv \{n_1, n_2, \ldots, n_{|\mathcal{N}|}\}$ denotes the finite set of cooperative agents, with $|\mathcal{N}|$ representing the total number of agents. $\mathcal{S}$ is the global state space, and $\mathcal{A} \equiv \mathcal{A}_1 \times \cdots \times \mathcal{A}_{|\mathcal{N}|}$ is the joint action space composed of individual agent action spaces. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the shared reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the global transition function, and $\gamma \in [0, 1)$ is the discount factor.

At each timestep t, each agent $n_i \in \mathcal{N}$ selects an action $a_t^i \in \mathcal{A}_i$ based on the global state $s_t \in \mathcal{S}$. The set of individual actions forms a joint action $\boldsymbol{a}_t \equiv [a_t^i]_{i=1}^{|\mathcal{N}|}$. This joint action induces a transition to the next global state $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, \boldsymbol{a}_t)$ and yields a shared reward $r_t = \mathcal{R}(s_t, \boldsymbol{a}_t)$.

Each agent n_i maintains a local trajectory τ_i, which consists of its sequence of observations, actions, and rewards over time. It optimizes a local policy $\pi_i(a_t^i \mid \tau_i)$ to maximize the overall team performance, defined by the expected cumulative discounted reward under the joint policy $\boldsymbol{\pi} \equiv \{\pi_1, \pi_2, \ldots, \pi_{|\mathcal{N}|}\}$:

$$\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} \mathbb{E}_{\mathcal{P}, \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \boldsymbol{a}_t) \right]. \tag{4}$$

2.1.3 State or Action Abstraction

State or action abstraction (Abel, 2022) aims to simplify decision-making by designing a parameterized abstraction function f_ϕ with the trainable parameter ϕ, which maps primitive states and actions to their abstract counterparts.

Specifically, f_ϕ maps each primitive state $s \in \mathcal{S}$ to an abstract state $z_s \in \mathcal{Z}_s$, i.e., $f_\phi : \mathcal{S} \to \mathcal{Z}_s$, or each primitive action $a \in \mathcal{A}$ to an abstract action $z_a \in \mathcal{Z}_a$, i.e., $f_\phi : \mathcal{A} \to \mathcal{Z}_a$. This abstraction reduces the complexity of the original decision process by compressing state and action representations, enabling more efficient policy learning through reduced dimensionality and improved generalization. The resulting abstract MDP is formally defined as a tuple:

$$\mathcal{M}_\phi = \langle \mathcal{Z}_s, \mathcal{Z}_a, \mathcal{R}_\phi, \mathcal{P}_\phi, \gamma \rangle, \tag{5}$$

where $\mathcal{Z}_s$ and $\mathcal{Z}_a$ are the abstract state and action spaces, $\mathcal{P}_\phi : \mathcal{Z}_s \times \mathcal{Z}_a \to \Delta(\mathcal{Z}_s)$ is the abstract transition function, $\mathcal{R}_\phi : \mathcal{Z}_s \times \mathcal{Z}_a \to \mathbb{R}$ is the abstract reward function, and $\gamma \in [0, 1)$ is the discount factor.

The abstract transition and reward functions, $\mathcal{P}_\phi$ and $\mathcal{R}_\phi$, are constructed by aggregating the transition dynamics and reward distributions of the underlying MDP over the pre-image sets induced by f_ϕ. Specifically, they are defined as follows:

$$\mathcal{P}_\phi(z_k^s | z_i^s, z_j^a) = \sum_{s_t \in z_i^s} \sum_{a_t \in z_j^a} \sum_{s_{t+1} \in z_k^s} \mathcal{P}(s_{t+1} | s_t, a_t), \quad \mathcal{R}_\phi(z_i^s, z_j^a) = \sum_{s_t \in z_i^s} \sum_{a_t \in z_j^a} \mathcal{R}(s_t, a_t), \tag{6}$$

where $z_i^s \in \mathcal{Z}_s$ and $z_j^a \in \mathcal{Z}_a$ denote an abstract state and abstract action, respectively.

2.1.4 Skill-based Learning

In **skill-based learning**, we use the term 'skill' to refer broadly to the general concept of reusable behaviors or abilities, while an 'option' refers to a specific, parameterized instance of a skill, defined by an initiation set, an option policy, and a termination condition, as established in prior work (Sutton et al., 1999; Machado et al., 2023).

Following the option framework (Precup, 2000), an option $\kappa \in \mathcal{K}$, representing a learned skill, is formally defined as a tuple:

$$\kappa = \langle \mathcal{I}_\kappa, \pi_\kappa, \mathcal{T}_\kappa \rangle, \tag{7}$$

where $\mathcal{I}_\kappa \subseteq \mathcal{S}$ is the initiation set in which the option κ can be executed, $\pi_\kappa : \mathcal{S} \to \Delta(\mathcal{A})$ is is the option policy mapping states to a probability distribution over primitive actions, and $\mathcal{T}_\kappa : \mathcal{S} \to [0, 1]$ is the termination function specifying the probability of terminating the option at a given state.

Incorporating the skill space $\mathcal{K}$ within an MDP gives rise to a hierarchical two-level policy structure. The high-level policy π_k^h selects an option $\kappa \in \mathcal{K}$, while the low-level policy π_k^l governs primitive action execution under the chosen option until the termination function $\mathcal{T}_\kappa$ signals the end of the option.

2.1.5 Role-based Learning

In **role-based learning** (Wilson et al., 2010; Wang et al., 2021b), the goal is to improve agent coordination and enhance scalability in complex multi-agent cooperative tasks by decomposing $\mathcal{M}_m$ into subtasks through the assignment of specialized roles. This decomposition is guided by a predefined role space Ψ, where each role imposes structural constraints on agent behavior by limiting its available actions. By reducing ambiguity in role assignments, this approach facilitates more efficient learning and execution of cooperative behaviors.

Each role $\rho_j \in \Psi$ encapsulates a subtask and an associated policy, formally defined as:

$$\rho_j = \left\langle t_j, \pi_{\rho_j} \right\rangle, \tag{8}$$

where the subtask $t_j = \langle \mathcal{N}_j, \mathcal{S}, \mathcal{A}_j, \mathcal{R}, \mathcal{P}, \gamma \rangle$ is derived from the global task but restricted to a subset of agents $\mathcal{N}_i \subseteq \mathcal{N}$. The role policy $\pi_{\rho_j} : \mathcal{S} \to \Delta(\mathcal{A}_j)$ governs action selection by assigning a probability distribution over the actions available within subtask t_j.

Within each subtask t_j, agents in $\mathcal{N}_j$ operate in a constrained action subspace $\mathcal{A}_j \subseteq \mathcal{A}$, which minimizes action overlap across roles and promotes more structured and efficient role-based coordination.

2.2 Structural Information Principles

Efficient and robust decision-making requires abstracting raw state and action information by eliminating irrelevant details while preserving essential features. To model structural relationships among states or actions, we construct a weighted, undirected graph $G = (V, E, W)$, built separately for each entity type. In this graph, all vertices V represent either states S or actions A, and edges E connect vertices that exhibit functional similarity, typically computed via feature-based similarity metrics. The edge weights $W : E \to \mathbb{R}_{\geq 0}$

quantify the degree of similarity between connected vertices, and the degree of a vertex $v \in V$, denoted d_v, is defined as the sum of the weights of all its incident edges.

Building on this graphical representation, we introduce encoding trees (Li and Pan, 2016) to represent a hierarchical partitioning of the vertices based on their feature similarities. Each level of the tree reflects a progressive grouping of elements, where lower levels correspond to fine-grained partitions that capture detailed distinctions, and higher levels represent broader aggregations that highlight overarching structural patterns. This hierarchical structure helps capture both local and global structural relationships among states or actions, enabling the model to recognize patterns of similarity and connectivity that influence decision-making processes.

We then leverage structural entropy—a measure of uncertainty associated with hierarchical partitioning—to quantitatively assess the quality of the induced hierarchy. Minimizing structural entropy guides partitioning to capture the intrinsic organization of the state or action space more faithfully.

2.2.1 Encoding Tree

Given the weighted, undirected graph G, we define an **encoding tree** T that hierarchically partitions the set of states or actions according to their feature-driven similarities. This tree is rooted and satisfies the following properties:

• The root node $\lambda \in T$ corresponds to the entire vertex set V, meaning that $V_\lambda = V$, and it serves as the starting point for the hierarchical partitioning.

• Each leaf node $\nu \in T$ corresponds to a singleton set containing a single vertex $v \in V$, i.e., $V_\nu = v$, representing the finest level of partitioning in the tree.

• Each internal node $\alpha \in T$ (i.e., neither a root nor a leaf) corresponds to a vertex subset $V_\alpha \subseteq V$, grouping states or actions that are further partitioned in the tree.

• Each non-root node $\alpha \in T$ has a unique parent node, denoted as α^-, from which α is directly descended in the tree structure.

• Each non-leaf node $\alpha \in T$ has exactly $L_\alpha \geq 2$ children, indexed from left to right as $\alpha_1, \alpha_2, \ldots, \alpha_{L_\alpha}$, where the index i increases sequentially.

• For each non-leaf node $\alpha \in T$, the vertex subsets of its children are pairwise disjoint, satisfying $V_\alpha = \bigcup_{i=0}^{L_\alpha - 1} V_{\alpha_i}$ and $V_{\alpha_i} \cap V_{\alpha_j} = \emptyset$ for all $i \neq j$.

2.2.2 Structural Entropy

The **one-dimensional structural entropy** measures the dynamical uncertainty in single-step random walks within the state or action graph G, without applying any partitioning strategy, where all states or actions belong to a single community. Mathematically, it is equivalent to the Shannon entropy of the stationary distribution induced by vertex degrees, reflecting the local similarity structure of each state or action with its neighbors. It is defined as follows:

$$H^1(G) = -\sum_{v \in V} \frac{d_v}{\mathrm{vol}(G)} \cdot \log_2 \frac{d_v}{\mathrm{vol}(G)}, \tag{9}$$

where $\mathrm{vol}(G) = \sum_{v \in V} d_v$ is the total volume of the graph, representing the sum of all vertex degrees. An encoding tree T introduces a hierarchical partitioning strategy that refines this

entropy by clustering strongly connected vertices, thereby revealing the inherent hierarchical community structure of the graph.

The ***K*-dimensional structural entropy** quantifies the residual structural entropy in the graph G after applying a hierarchical partitioning strategy represented by an encoding tree T with height at most K. For each non-root node α in T, the assigned structural entropy is defined as follows:

$$H^T(G;\alpha) = -\frac{g_\alpha}{\mathrm{vol}(G)} \log_2 \frac{\mathcal{V}_\alpha}{\mathcal{V}_{\alpha^-}}, \tag{10}$$

where g_α denotes the total edge weight of all edges crossing the boundary of V_α, and $\mathcal{V}_\alpha$ denotes the volume of the subgraph induced by V_α, i.e., the sum of degrees of its constituent vertices. A higher value of $H^T(G;\alpha)$ indicates greater uncertainty in single-step transitions from the parent community V_{α^-} to the sub-community V_α.

The overall K-dimensional structural entropy is defined as the minimum total entropy over all encoding trees of height at most K, formally expressed as follows:

$$H^K(G) = \min_T \left\{ H^T(G) \right\}, \quad H^T(G) = \sum_{\alpha \in T, \alpha \neq \lambda} H^T(G;\alpha). \tag{11}$$

A structural entropy optimization algorithm guides the construction of such an encoding tree, and the tree that minimizes this entropy is referred to as the optimal encoding tree. This tree effectively captures the hierarchical community structure among states or actions based on feature similarity. In each sampled batch, states or actions with similar features are adaptively grouped into communities, forming the foundation for subsequent state and action abstraction mechanisms.

3. Related Work

This section reviews related work on structural information principles, state abstraction, and hierarchical decision-making, highlighting the motivation behind our proposed SIDM framework and its primary advantages over existing approaches.

3.1 Structural Information Principle

Structural information (Li and Pan, 2016) was introduced in 2016 to quantify the dynamic uncertainty in complex graph structures. Specifically, structural entropy measures the minimum number of bits required to identify an accessible vertex in a single-step random walk. The principles of structural entropy minimization (Li et al., 2016, 2018) were later introduced to automatically identify the optimal partitioning strategy, known as the encoding tree, which reveals the hierarchical community structure of vertices.

Since then, structural information principles have been applied across various fields, including graph learning (Liu et al., 2019; Wu et al., 2022), network analysis (Zeng et al., 2024; Cao et al., 2024), and reinforcement learning (Zeng et al., 2023a,b). However, existing research on structural information has primarily focused on undirected graphs, which are unable to capture irreversible directional relationships between graph vertices, leading to inevitable information loss.

In this work, we extend structural information principles by proposing a formal definition and an optimization algorithm for directed graphs. Furthermore, we leverage structural information from historical state-action trajectories to develop a novel hierarchical decision-making framework based on state and action abstractions applicable to both single-agent and multi-agent scenarios.

3.2 State Abstraction

Value function approximation (Mahadevan and Maggioni, 2005, 2007) utilizes spectral analysis of the state space to derive low-dimensional, compact representations of the Markov Decision Process, thereby improving policy learning. The successor representation characterizes states by their discounted occupancy and visitation density, facilitating knowledge transfer across tasks (Barreto et al., 2017) and enabling learning conditioned on specific goals (Hoang et al., 2021). However, these methods often suffer from instability and poor generalization in noisy, high-dimensional environments, limiting their effectiveness.

Previous studies have investigated aggregation functions that cluster similar states to reduce task complexity (Hutter, 2016; Abel et al., 2018). Abstract State Transition Graphs (STGs) (de Mendonça et al., 2018) create compact state representations by identifying structurally similar states through encoded features. The DSAA method (Attali et al., 2022) achieves end-to-end state-action abstraction based on successor features and max-entropy regularization. Nevertheless, these methods rely on prior knowledge about underlying tasks, such as the distance threshold parameter in the DBSCAN algorithm and the assumed N-simplex distribution over abstract states, which limits their stability and effectiveness across diverse scenarios. Recent studies (Zang et al., 2022; Zhu et al., 2022) have explored different learning objectives to refine state representations from environmental observations. While these methods yield strong representational power, they often prioritize invariance, potentially discarding essential environmental details—such as context-specific features or dynamic variations—leading to inaccurate characterizations of the original decision process (Abel et al., 2019).

In this work, we leverage similarity-based structural relationships between states to achieve adaptive state abstraction, balancing the removal of irrelevant information with the preservation of critical decision-making features for robust policy learning.

3.3 Hierarchical Decision Making

3.3.1 Skill-based Learning

Skill-based learning utilizes previously acquired behaviors or skills to facilitate efficient exploration in complex decision-making tasks (Merel et al., 2018, 2020). The DHRL method (Lee et al., 2022) separates the time horizons of low-level and high-level policies using a graph structure, improving training efficiency in standard environments. Meanwhile, Gaussian prior (Pertsch et al., 2021) and conditional generative models (Singh et al., 2020) are introduced to approximate the distribution of relevant skills for a given state and to guide exploration of the skill space, respectively.

Despite these benefits, skill discovery remains a fundamental challenge due to the difficulty of identifying reusable and transferable behaviors across diverse tasks. To address

this challenge, researchers have explored structured approaches for constructing meaningful skills. One approach (Jinnai et al., 2019) constructs skills by minimizing cover time—a metric derived from the state transition graph—to accelerate learning in environments with sparse rewards. Reward-respecting subtasks (Sutton et al., 2023) incorporate environmental rewards associated with terminating states, ensuring alignment with the overall reward structure. The ROD algorithm (Machado et al., 2023) employs spectral analysis and clustering on successor representations to identify skills that facilitate temporally extended exploration. However, these methods typically rely on a fixed skill set, limiting adaptability in dynamic environments. Furthermore, their dependence on the original state space constrains scalability, particularly in complex environments with high-dimensional and noisy state spaces.

In this work, we define and optimize the structural entropy of directed transitions between abstract states to construct a skill hierarchy with an adaptive scale. The DSAA algorithm (Attali et al., 2022) and the Louvain algorithm (Evans and Şimşek, 2023) were recently introduced to similarly compute skills between abstract states and to construct multi-level skill hierarchies, respectively. However, their limited consideration of transition-based relationships between abstract states, along with their reliance on a fixed-resolution partitioning parameter, constrains their effectiveness and flexibility in decision-making across diverse evaluation scenarios, as discussed in Section 6.2.

3.3.2 Role-based Learning

In natural systems (Gordon, 1996; Jeanson et al., 2005; Butler, 2012), individuals assume specialized roles that enhance labor division and improve operational or cognitive efficiency. Inspired by this, multi-agent systems decompose tasks and assign specialized agents to subtasks, thereby reducing design complexity and improving performance (Wooldridge et al., 2000; Bonjean et al., 2014).

However, the practical implementation of these approaches is often constrained, as predefined task decompositions and roles are not always available in real-world environments, making their application challenging (Lhaksmana et al., 2018; Sun et al., 2020). To address this, Bayesian inference (Wilson et al., 2010) has been incorporated into MARL algorithms to infer roles dynamically, while the ROMA methodology (Wang et al., 2020) fosters role emergence by designing a specialization objective that encourages agents to differentiate their tasks. Despite these advances, exhaustive exploration of the entire state-action space remains computationally prohibitive. The RODE method alleviates this issue by decomposing joint action spaces (Wang et al., 2021b), facilitating more efficient exploration and coordination among agents through a focus on relevant action subspaces. Nevertheless, its effectiveness is strongly influenced by domain-specific knowledge, as it remains highly sensitive to parameters such as the number of clusters and the choice of distance metrics in clustering algorithms.

In this work, we leverage structural similarities between actions to achieve hierarchical action abstraction, enabling an adaptive, role-based learning method that requires no prior knowledge. By dynamically structuring the action space, our approach improves scalability and generalization across diverse multi-agent collaboration scenarios.

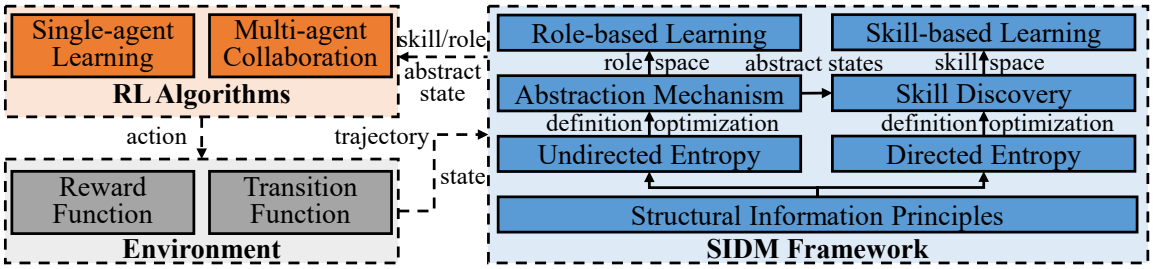


Figure 2: The overall decision-making process, including the environment, the proposed SIDM framework, and various downstream RL algorithms.

4. The Proposed SIDM Framework

This section provides an overview of our proposed SIDM framework, which takes as input the original states and historical trajectories from the environment and outputs the discovered skills, roles, and corresponding abstract states, which can be utilized by downstream RL algorithms, as shown in Figure 2. Specifically, our framework introduces an adaptive abstraction mechanism to derive abstract representations of states and actions, defines and optimizes directed structural entropy in abstract transitions to facilitate hierarchical skill discovery, and develops skill-based and role-based learning methods for single-agent and multi-agent decision-making, respectively.

4.1 Abstraction Mechanism

To improve learning efficiency and reduce the complexity of high-dimensional and noisy environmental information, we present an adaptive abstraction mechanism based on structural information principles. This mechanism partitions similar states and actions into distinct communities, producing compact and meaningful abstract state and action representations.

At each timestep t, we sample a batch of size n from the replay buffer $\mathcal{B}$, which includes the current state variable S_t, action variable A_t, subsequent state variable S_{t+1}, and reward variable R_t. Each of these denotes a batch of random variables drawn from the replay buffer. We first employ an encoder-decoder architecture (Cho et al., 2014) to transform the original high-dimensional variables S_t, S_{t+1}, and A_t into their low-dimensional representations, denoted as S'_t, S'_{t+1}, and A'_t, respectively. The transformation process is formally defined as follows:

$$S'_t = f^s_{\phi_s}(S_t), \quad S'_{t+1} = f^s_{\phi_s}(S_{t+1}), \quad A'_t = f^a_{\phi_a}(A_t), \tag{12}$$

where $f^s_{\phi_s}$ and $f^a_{\phi_a}$ denote the state and action encoders with parameters ϕ_s and ϕ_a, respectively. To highlight similarities between action functionalities and state transitions, we introduce two decoders, each with a distinct objective: the state decoder d_s predicts actions based on state transitions, while the action decoder d_a reconstructs the next state based on actions and current states. Specifically, the state decoder $d^s_{\theta_s}(a_t|s'_t, s'_{t+1}; \theta_s)$ employs a cross-entropy loss for inverse dynamics modeling (Allen et al., 2021) to predict the current action $a_t \in A_t$ given the adjacent state representations $s'_t \in S'_t$ and $s'_{t+1} \in S'_{t+1}$. The action decoder $d^a_{\theta_a}(s_{t+1}|s'_t, a'_t; \theta_a)$ reconstructs the next state $s_{t+1} \in S_{t+1}$ using the action representation $a'_t \in A'_t$ and the current state representation $s'_t \in S'_t$. The decoder loss $\mathcal{L}_{de}$

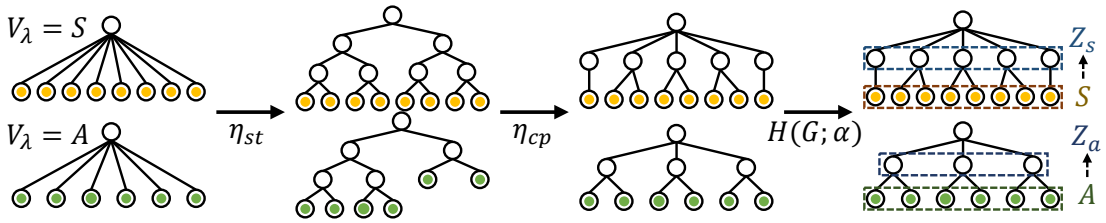


Figure 3: The abstraction mechanism for states and actions.

is computed as follows:

$$\mathcal{L}_{de} = -\frac{1}{n} \cdot \sum_{i=1}^{n} \left[||d^s_{\theta_s}(s'_t, s'_{t+1}) - a_t||_2^2 + ||d^a_{\theta_a}(s'_t, a'_t) - s_{t+1}||_2^2 \right]. \tag{13}$$

For simplicity, we denote the combined set of abstract state representations, S'_t and S'_{t+1} as S, and refer to A'_t as A. In the following discussion, we use state abstraction as an illustrative example, while applying the same methodology to action abstraction.

Although the bisimulation metric (Castro, 2020) effectively captures behavioral equivalence between states, its computational complexity is prohibitively high. Therefore, we second adopt a more computationally efficient similarity metric based on Pearson Correlation Analysis. For each pair of states $s_i, s_j \in S$ with $i \neq j$, we compute their similarity score $\mathcal{C}(s_i, s_j)$ using Pearson correlation over their feature dimensions:

$$\mathcal{C}(s_i, s_j) = \frac{\mathbb{E}\left((s_i - \mu_{s_i})(s_j - \mu_{s_j})\right)}{\sigma_{s_i}\sigma_{s_j}}, \tag{14}$$

where μ_{s_i} and μ_{s_j} denote the means, and σ_{s_i} and σ_{s_j} denote the variances of state representations s_i and s_j, respectively. A higher absolute value of $\mathcal{C}(s_i, s_j)$ indicates greater similarity between states s_i and s_j, guiding the state abstraction process by grouping similar states together.

Third, we construct a complete, weighted, and undirected state graph G_s, where each state $s \in S$ is a vertex and each edge (s_i, s_j) is weighted by the metric $\mathcal{C}(s_i, s_j)$. To eliminate the interference of insignificant connections, particularly those with absolute values near zero, we apply edge filtration to the complete state graph G_s. Following the prior study (Li et al., 2016), we simplify G_s into a k-nearest neighbor (kNN) graph by minimizing its one-dimensional structural entropy. The filtration procedure, summarized in Algorithm 1, involves treating each state $s \in S$ as a center vertex and retaining only its k edges with the highest absolute weights to construct the kNN graph G_{k} (line 4 in Algorithm 1). We then compute the one-dimensional structural entropy $H^1(G_{\mathrm{k}})$ for the resulting graph G_{k} (line 5 in Algorithm 1). To determine the optimal parameter k^*, we evaluate $H^1(G_{\mathrm{k}})$ across a range of plausible k values and select the one that minimizes entropy (lines 3 and 8 in Algorithm 1). The optimized graph G_{k^*} serves as the final sparse state graph G^*_s (lines 9 and 10 in Algorithm 1).

Fourth, as illustrated in Figure 3, we determine the optimal partitioning structure of the sparse state graph G^*_s and design an aggregation function derive abstract states within each community. We initialize a one-layer partitioning structure for G^*_s, represented as the

Algorithm 1: The Edge Filtration Algorithm

Input: a weighted, undirected, and complete graph $G = (V, E, W)$
Output: a sparsified graph G^*
for k = 1 to $n - 1$ **do**
  if the kNN graph G_{k} exists **then**
    $H^1(G_{\mathrm{k}}) \leftarrow$ calculate the one-dimensional structural entropy of G_{k}
  end if
end for
$\mathrm{k}^* \leftarrow \arg\min_k\{H^1(G_{\mathrm{k}})\}$
$G^* \leftarrow G_{\mathrm{k}^*}$
return G^*

initial encoding tree T_s, where each community initially consists of a single state. To further optimize this partitioning structure, we introduce two stable-enhancing operators from the HSCE algorithm (Pan et al., 2021): *stretch* (η_{st}) and *compress* (η_{cp}), which iteratively reduce the structural entropy of G_s^* under T_s, increasing the tree height from 1 to 2. The optimization process is summarized in Algorithm 2. During a "*stretch-compress*" cycle, we operate on the set U_i^s of tree nodes at layer i in T_s and quantify the average variation in structural entropy as $\overline{\mathrm{Spar}}_i(T_s)$. In each iteration, we traverse all sets of tree nodes at the same level and select the set that results in the greatest reduction in structural entropy (line 4 in Algorithm 2). These selected nodes then undergo a "*stretch-compress*" cycle (lines 8 - 13 in Algorithm 2). The iteration terminates when either the tree height reaches $\underline{h_{T_s}} = 2$ (line 3 in Algorithm 2) or no node set exhibits a positive entropy reduction, i.e., $\overline{\mathrm{Spar}}_{i^*}(T_s) > 0$ (lines 5 and 6 in Algorithm 2). The process outputs the optimal encoding tree T_s^* (lines 19 and 20 in Algorithm 2).

In the optimal encoding tree T_s^*, the structural entropy assigned to each node, as specified in Equation 10, quantifies the uncertainty of a single-step random walk reaching its associated vertex community from its parent's community. Treating this uncertainty as the node weight, we design an aggregation function to compute the node representations in T_s^*. For each leaf node $\nu \in T_s^*$ with a corresponding state set $V_\nu = s$, its representation is defined as $h_\nu = s$. For each non-leaf node α, we normalize the weights of its children using the softmax function to ensure a probabilistic distribution, then compute its node representation as follows:

$$h_\alpha = \sum_{i=1}^{L_\alpha} \frac{\exp\left(H^{T_s^*}\left(G_s^*; \alpha_i\right)\right)}{\sum_{j=1}^{L_\alpha} \exp\left(H^{T_s^*}\left(G_s^*; \alpha_j\right)\right)} \cdot h_{\alpha_i}. \tag{15}$$

The representation of each child node h_{λ_i} of the root node λ is defined as an abstract state z_i^s, where its corresponding state set is denoted as $S_i \subset S$. Thus, the set of abstract states is defined as $Z_s = \{z_1^s, z_2^s, \ldots, z_{L_\lambda}^s\}$.

At timestep t, the original environmental state $s_t \in \mathcal{S}$ is embedded into a low-dimensional vector and mapped to an abstract state z_t^s based on its dot products with the abstract representations in Z_s, as follows:

$$z_t^s = \arg\max_{z_i^s \in Z_s} \langle f_{\phi_s}^s(s_t), z_i^s \rangle, \tag{16}$$

Algorithm 2: The Undirected Optimization Algorithm

Input: a one-layer initial encoding tree T
Output: an optimized two-layer encoding tree T^*
while the tree height $h_T < 2$ **do**
  $i^* \leftarrow \arg\max_i\{\overline{\text{Spar}}_i(T)\}$
  # determine whether the iteration is terminated
  if $\overline{\text{Spar}}_{i^*}(T) = 0$ **then**
    break
  end if
  # execute optimizations on selected nodes
  for $\alpha \in U_{i^*}$ **do**
    $T \leftarrow \eta_{st}(T_\alpha)$
    $T \leftarrow \eta_{cp}(T_\alpha)$
    $h_T \leftarrow h_T + 1$
  end for
  # adjust and update tree structure
  for $i = i^* + 1$ to h_T **do**
    update U_i
  end for
end while
$T^* \leftarrow T$
return T^*

where $f^s_{\phi_s}$ is the encoding function for states. Similarly, the abstract action variable Z_a is defined as $Z_a = \{z^a_1, z^a_2, \ldots, z^a_{L_\lambda}\}$, where each abstract action z^a_i corresponds to a subset $A_i \subset A$. In the discrete action space $\mathcal{A}$, the original actions associated with the embedded actions in A_i collectively form an action subspace, denoted as $\mathcal{A}_i \subset \mathcal{A}$.

4.2 Directed Structural Entropy

To address the limitations posed by undirected constraints in existing structural information principles (Li and Pan, 2016; Zeng et al., 2023a,b), we define and optimize high-dimensional structural entropy for directed graphs, enabling subsequent accurate representation of key transition dynamics between abstract states in RL problems.

Given a directed graph $G_{dir} = (V, E_{dir}, W_{dir})$ with non-negative edge weights, we introduce Algorithm 3 to modify its structure with two primary objectives: (i) to ensure its strong connectivity, meaning there exists a directed path between any pair of vertices, and (ii) to make it more conducive to a random walk process by normalizing edge weights such that the weighted out-degree sum of each vertex is 1. To achieve this, we first identify the strongly connected components of G_{dir} (lines 3 and 4 in Algorithm 3) and introduce directed edges with minimal weights to establish connectivity between these components (lines 5 - 9 in Algorithm 3). Then, for each vertex $v \in V$, we normalize the weights of all outgoing edges by dividing each weight by the vertex's weighted out-degree sum (lines 11 - 13 in Algorithm 3). The following proposition confirms the existence and uniqueness of the

Algorithm 3: The Directed Graph Adjustion Algorithm

Input: a directed, weighted graph with non-negative edge weights $G_{dir} = (V, E_{dir}, W_{dir})$
Output: a strongly connected, directed graph G'_{dir} with normalized edge weights
 # identify strongly connected components (SCCs)
$SCCs \leftarrow$ Kosaraju-SCC(G)
$G_{SCC} = (V_{SCC}, E_{SCC}) \leftarrow$ create a meta-graph of connected components
 # ensure strong connectivity between SCCs
for (SCC_i, SCC_j) that lack a directed path in G_{SCC} **do**
  $G_{dir} \leftarrow$ introduce a minimal-weight edge from SCC_i to SCC_j
end for
 # normalize edge weights to weighted out-degree sum to 1
for $(v_i, v_j) \in E_{dir}$ **do**
  $W_{dir}(v_i, v_j) \leftarrow W_{dir}(v_i, v_j) / \sum_{(v_i, v_k) \in E} W_{dir}(v_i, v_k)$
end for
$G'_{dir} \leftarrow G_{dir}$
return G'_{dir}

stationary distribution π_s over the vertices of the adjusted graph G'_{dir}, forming the basis for the definition and optimization of directed structural entropy.

Proposition 1 *Given a strongly connected, directed graph $G'_{dir} = (V, E'_{dir}, W'_{dir})$ with non-negative edge weights, where the weighted out-degree of each vertex sums to* 1*, its stationary distribution π_s exists and is unique. This distribution corresponds to the unique eigenvector associated with the dominant eigenvalue* 1 *of the adjacency matrix A'_{dir}.*

Two distinct proofs of this proposition, one based on the Perron-Frobenius theorem and the other on the properties of Markov chains, are provided in Appendix C.1.

For the strongly connected graph G'_{dir}, we compute the stationary distribution π_s over all vertices and define the one-dimensional directed structural entropy as follows:

$$H^1(G'_{dir}) = -\sum_{v \in V} \pi_s(v) \cdot \log \pi_s(v), \tag{17}$$

where $\pi_s(v)$ denotes the stationary probability of vertex v in G'_{dir}. Using the stationary distribution π_s, we refine the terms g_α and $\mathcal{V}_\alpha$ for each non-root node α in the encoding tree T_{dir} of G'_{dir}, following Equation 10, and redefine its assigned entropy as follows:

$$\mathcal{V}_\alpha = \sum_{v_i \in V} \sum_{v_j \in V_\alpha} \left[\pi_s(v_i) \cdot W'_{dir}(v_i, v_j)\right], \tag{18}$$

$$g_\alpha = \sum_{v_i \notin V_\alpha} \sum_{v_j \in V_\alpha} \left[\pi_s(v_i) \cdot W'_{dir}(v_i, v_j)\right], \tag{19}$$

$$H^{T_{dir}}(G'_{dir}; \alpha) = -\frac{g_\alpha}{\mathrm{vol}(G'_{dir})} \cdot \log_2 \frac{\mathcal{V}_\alpha}{\mathcal{V}_{\alpha^-}}, \tag{20}$$

where the volume of G'_{dir}, $\mathrm{vol}(G'_{dir})$, is defined as the sum of in-degrees and out-degrees of all vertices:

$$\mathrm{vol}(G'_{dir}) = \sum_{v \in V} (d_v^+ + d_v^-). \tag{21}$$

Thus, the K-dimensional directed structural entropy of G'_{dir} is redefined as follows:

$$H^{T_{dir}}(G'_{dir}) = \sum_{\alpha \in T_{dir}, \alpha \neq \lambda} H^{T_{dir}}(G'_{dir}; \alpha), \quad H^K(G'_{dir}) = \min_{T_{dir}} \left\{ H^{T_{dir}}(G'_{dir}) \right\}, \tag{22}$$

where T_{dir} ranges over all encoding trees with a maximal height of K.

Expanding upon the *merge* (η_{mg}) and *combine* (η_{cb}) operators introduced by deDoc (Li et al., 2018), we develop an optimization approach for directed structural entropy $H^{T_{dir}}(G'_{dir})$ to determine the optimal tree-structure partitioning strategy T^*_{dir}. To calculate the entropy variation caused by a single *merge* or *combine* operation on sibling nodes $\alpha, \beta \in T_{dir}$, we introduce $g_{\alpha,\beta}$ defined as the total weight of edges connecting vertices in V_α to vertices in V_β as follows:

$$g_{\alpha,\beta} = \sum_{v_i \in V_\alpha} \sum_{v_j \in V_\beta} \pi_s(v_i) \cdot W'_{dir}(v_i, v_j). \tag{23}$$

The entropy variation caused by a single *merge* operation on sibling nodes $\alpha, \beta \in T_{dir}$ is denoted as $\Delta_{mg}(T_{dir}, \alpha, \beta)$ and is calculated as follows:

$$\Delta_{mg}(T_{dir}, \alpha, \beta) = \frac{g_{\alpha,\beta} + g_{\beta,\alpha}}{\mathrm{vol}(G'_{dir})} \cdot \log_2 \frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\mu_{mg}}} - \frac{\sum_{i \neq j}^{L_\alpha} g_{\alpha_i,\alpha_j}}{\mathrm{vol}(G'_{dir})} \cdot \log_2 \frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\alpha} - \frac{\sum_{i \neq j}^{L_\beta} g_{\beta_i,\beta_j}}{\mathrm{vol}(G'_{dir})} \cdot \log_2 \frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\beta}, \tag{24}$$

where μ_{mg} is the newly added node via the *merge* operation. The entropy variation caused by a single *combine* on sibling nodes $\alpha, \beta \in T_{dir}$ is denoted as $\Delta_{cb}(T_{dir}, \alpha, \beta)$ and is calculated as follows:

$$\Delta_{cb}(T_{dir}, \alpha, \beta) = \frac{g_{\alpha,\beta} + g_{\beta,\alpha}}{\mathrm{vol}(G'_{dir})} \cdot \log_2 \frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\mu_{cb}}}, \tag{25}$$

where μ_{cb} is the newly added node via the *combine* operation. Detailed deviation of these entropy variations is provided in Appendix B.1.

We summarize the iterative optimization process in Algorithm 4. At each iteration, we traverse all pairs of sibling nodes (lines 5 and 11 in Algorithm 4) and selectively execute either the *merge* or *combine* operator (lines 7 and 13 in Algorithm 4), choosing the operation that maximally reduces structural entropy while ensuring the tree height remains below K (line 3 in Algorithm 4). When no node pair satisfies $\Delta_{mg}(T_{dir}, \alpha, \beta) > 0$ or $\Delta_{cb}(T_{dir}, \alpha, \beta) > 0$ (lines 6 and 12 in Algorithm 4), we terminate the iteration and output T as the optimal encoding tree T^* (lines 18 and 19 in Algorithm 4).

Adding directed edges between strongly connected components in the directed graph G_{dir} may alter the original topological relationships, potentially disrupting irreversible environmental constraints among vertices. To mitigate this issue, Algorithm 3 assigns significantly smaller weights to these additional edges compared to the original transition edges. This guarantees minimal interference with environmental constraints. Specifically, Algorithm 3 carefully ensures the stationary distribution π_s exists and remains unique, while

Algorithm 4: The Directed Optimization Algorithm

Input: an one-layer initial encoding tree T_{dir}, $K \in \mathbb{Z}^+$
Output: the K-layer optimal encoding tree T^*_{dir}
while tree height $h_{T_{dir}} < K$ **do**
  # execute *merge* optimization
  $(\alpha^*, \beta^*) \leftarrow \arg\max\{\Delta_{mg}(T_{dir}, \alpha, \beta)\}$ via Equation 24
  if $\Delta_{mg}(T_{dir}, \alpha^*, \beta^*) > 0$ **then**
    $T_{dir} \leftarrow \eta_{mg}(T_{dir}, \alpha^*, \beta^*)$
    continue
  end if
  # execute *combine* optimization
  $(\alpha^*, \beta^*) \leftarrow \arg\max\{\Delta_{cb}(T_{dir}, \alpha, \beta)\}$ via Equation 25
  if $\Delta_{cb}(T_{dir}, \alpha^*, \beta^*) > 0$ **then**
    $T_{dir} \leftarrow \eta_{cb}(T_{dir}, \alpha^*, \beta^*)$
    continue
  end if
  break
end while
$T^*_{dir} \leftarrow T_{dir}$
return T^*_{dir}

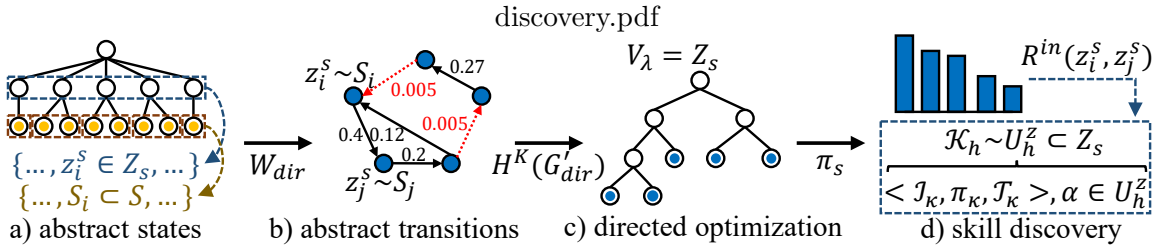


Figure 4: The skill discovery based on the directed structural information.

Algorithm 4 employs a greedy strategy prioritizing edges with substantial original weights. As a result, our approach preserves the structural consistency of the encoding tree before and after directed graph adjustment. Consequently, our graph adjustment method establishes an effective and principled framework for computing the structural entropy of directed graphs, preserving essential environmental constraints of the original task. This ensures the rationality and consistency of subsequent skill-based and role-based learning processes.

4.3 Skill Discovery

In this subsection, we construct an abstract state transition graph to model the decision-making process and minimize directed structural entropy to obtain the optimal hierarchical partitioning of abstract states. Within this tree structure, we facilitate skill discovery across hierarchical communities to capture abstract transition dynamics, accounting for the different temporal properties of the RL environment, as illustrated in Figure 4.

Taking the abstract states in Z_s as vertices, we construct a weighted, directed graph $G_{dir} = (Z_s, E_{dir}, W_{dir})$, where each vertex represents an abstract state with a corresponding subset of primitive states. A directed edge $(z_i^s, z_j^s) \in E_{dir}$ is established if there exists an original transition between the corresponding state subsets S_i and S_j. The edge weight $W_{dir}(z_i^s, z_j^s)$ is determined based on the occurrence frequency of these transitions in the sampled batch. To ensure strong connectivity and normalization, we refine the graph G_{dir} using Algorithm 3, resulting in the adjusted graph G'_{dir}. We apply Algorithm 4 to derive the optimal encoding tree T^*_{dir}, constrained to a maximum height of K.

For any integer $0 \leq h \leq K$, representing a hierarchical level in T^*_{dir}, the discovered options associated with the node set U_h^z at layer h in T^*_{dir} are defined as follows:

$$\mathcal{K}_h = \{\langle \mathcal{I}_{\kappa_i}, \pi_{\kappa_i}, \mathcal{T}_{\kappa_i} \rangle \mid \alpha_i \in U_h^z\}. \tag{26}$$

Each option $\kappa_i \in \mathcal{K}_h$ is characterized by its initiation set $\mathcal{I}_{\kappa_i}$, policy function π_{κ_i}, and termination condition $\mathcal{T}_{\kappa_i}$.

In the stationary distribution π_s, the distribution probability $\pi_s(z_j^s)$ assigned to each abstract state z_j^s reflects its criticality in the historical transition trajectory. Abstract states with higher probabilities are more frequently visited and can be considered critical for task completion. To leverage this notion of state criticality, we define an intrinsic reward function $\mathcal{R}^{in}$ based on the stationary distribution π_s over abstract states in G'_{dir}. The reward for transitioning between abstract states z_j^s and z_k^s is computed as follows:

$$\mathcal{R}^{in}(z_j^s, z_k^s) = \pi_s(z_k^s) - \pi_s(z_j^s). \tag{27}$$

For each option $\kappa_i \in \mathcal{K}_h$, the termination condition $\mathcal{T}_{\kappa_i}$ consists of abstract states where no further positive intrinsic reward is accumulated, given by:

$$\mathcal{T}_{\kappa_i}(z_j^s) = \begin{cases} 1 & \text{if } \arg\max_{z_k^s \in Z^s_{\alpha_i}} \pi_s(z_k^s) = z_j^s, \\ 0 & \text{otherwise,} \end{cases} \tag{28}$$

where $Z^s_{\alpha_i} \subset Z_s$ denotes the subset of abstract states corresponding to the node α_i in T^*_{dir}. The initiation set $\mathcal{I}_{\kappa_i}$ is defined as the set of states not included in the termination condition of κ_i, given by:

$$\mathcal{I}_{\kappa_i} = \{z_j^s \mid z_j^s \neq \arg\max_{z_k^s \in Z^s_{\alpha_i}} \pi_s(z_k^s)\}. \tag{29}$$

The option policy π_{κ_i} is trained by maximizing the expected long-term cumulative intrinsic reward, given by:

$$\pi^*_{\kappa_i} = \arg\max_{\pi_{\kappa_i}} \mathbb{E}_{\mathcal{P}} \left[\sum_{t=0}^{\mathcal{T}_{\kappa_i}(z_j^s)} \gamma^t \mathcal{R}^{in}(z_j^s, z_k^s) \mathcal{T}_{\kappa_i}(z_j^s) \right], \tag{30}$$

where z_j^s and z_k^s correspond to the abstract states associated with the environmental states s_t and s_{t+1} at timestep t, respectively.

Classical skill discovery approaches (Jinnai et al., 2019; Machado et al., 2023) employ eigenvalue decomposition to derive the eigenoption κ_e, which corresponds to the principal eigenvector associated with the largest eigenvalue of the transition matrix A'_{dir} of graph G'_{dir}. The following theorem establishes the relationship between our skill discovery method and these classical approaches to highlight the theoretical properties of our framework.

skills.pdf

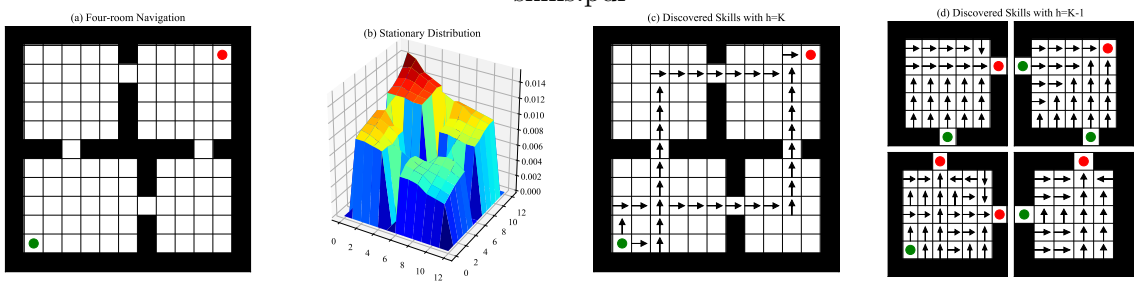


Figure 5: The identified skills under different hierarchy levels (h values) in the four-room navigation task.

learning.pdf

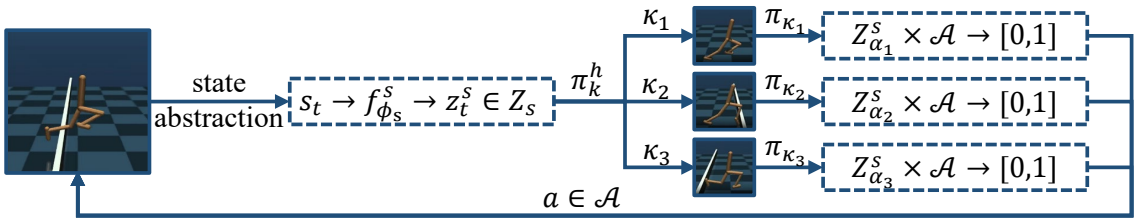


Figure 6: The single-agent skill-based learning in SIDM framework.

Theorem 2 *For $h = K$, the discovered set $\mathcal{K}_h$ consists of a single option κ_1, which is equivalent to the eigenoption κ_e associated with the principal eigenvector of the adjusted transition matrix A'_{dir}.*

The detailed proof is provided in Appendix C.2.

By adjusting the parameter h, our skill discovery method captures the temporal structure of the environment across different time scales. Figure 5 illustrates the discovered options in the four-room domain for varying values of h. For $h = K$, the sole option in $\mathcal{K}_h$ motivates the agent toward the state with the highest stationary probability in π_s, which lies along the environment's diagonal. For smaller values of h, the skill set enables finer-grained navigation over shorter time scales within a subspace of abstract states, where state transitions cover shorter distances, leading to more localized and refined options.

4.4 Skill-based Learning

In the single-agent decision-making environment, we apply the previously introduced state abstraction and skill discovery mechanisms to construct a hierarchical two-layer skill-based learning framework, as illustrated in Figure 6. To model the multi-scale temporal structure of the environment, we define the overall skill set $\mathcal{K}$ as the union of the identified skill sets obtained at hierarchical levels $h = K$ and $h = K - 1$. The complete procedure for the proposed two-layer skill-based learning framework is outlined in Algorithm 5.

At each timestep t, the agent observes the current environmental state s_t and applies the state abstraction function $f^s_{\phi_s}$ to map s_t to an abstract representation $z^s_t \in Z_s$ (line 5 in Algorithm 5). Based on the abstract state z^s_t, the high-level policy $\pi^h_k : Z_s \times \mathcal{K} \to [0, 1]$

Algorithm 5: The Single-Agent Skill-Based Learning Method

1: **Input:** batch size n, update interval t_{up}, replay buffer $\mathcal{B}$
2: **Output:** hierarchical skill policies π_κ^h and $\{\pi_{\kappa_1}, \pi_{\kappa_2}, \dots\}$
3: **for** each episode **do**
4: **for** each timestep t **do**
5: $z_t^s \leftarrow$ obtain an abstract representation of s_t via Equation 16
6: $\kappa_i \leftarrow$ select a skill from the skill space $\mathcal{K}$ based on the high-level policy π_k^h
7: $\tau_t = (s_t, a_t, s_{t+1}, r_t) \leftarrow$ collect transition based on the low-level policy π_{κ_i} and environmental state s_t
8: $\mathcal{B} \leftarrow \mathcal{B} \bigcup \tau_t$
9: **if** $t \mod t_{up} == 0$ **then**
10: $S_t, A_t, S_{t+1}, R_t \leftarrow$ sample a batch from $\mathcal{B}$
11: $G_s \leftarrow$ construct the complete state graph from variables S_t and S_{t+1}
12: $G_s^* \leftarrow$ filter out trivial edges from G_s
13: $T_s^* \leftarrow$ generate the optimal encoding tree
14: $Z_s \leftarrow$ aggregate states to obtain abstract embeddings via Equation 15
15: $\mathcal{K} \leftarrow$ update the skill set via the skill discovery method
16: $\mathcal{L}_{de}, \mathcal{L}_{rl} \leftarrow$ calculate the decoding loss and training loss
17: optimize the abstraction function and hierarchical policies by minimizing the loss $\mathcal{L}_{de}$ and $\mathcal{L}_{rl}$
18: **end if**
19: **end for**
20: **end for**

selects an option κ_i from the discovered set $\mathcal{K}$ (line 6 in Algorithm 5). The low-level option policy $\pi_{\kappa_i} : Z_{\alpha_i}^s \times \mathcal{A} \to [0, 1]$ maps the abstract state z_t^s to an action distribution over $\mathcal{A}$, from which it samples and executes an action a_t. This results in a transition to a new environmental state s_{t+1} and a received reward r_t (line 7 in Algorithm 5). Low-level policy training is performed using a single-agent underlying RL algorithm that optimizes both environmental and intrinsic rewards. The associated training loss is denoted as $\mathcal{L}_{rl}$.

At regular update intervals t_{up}, we retrieve samples containing state-action transitions (S_t, A_t, S_{t+1}, R_t) of size n from the replay buffer $\mathcal{B}$ (line 10 in Algorithm 5). Next, the state abstraction mechanism (Subsection 4.1) and skill discovery method (Subsection 4.3) are used to refine the abstract state set Z_s and update the skill set $\mathcal{K}$ (lines 11 - 15 in Algorithm 5). Finally, we calculate the training losses $\mathcal{L}_{de}$ and $\mathcal{L}_{rl}$ to optimize the state abstraction function $f_{\phi_s}^s$ and enhance the hierarchical policy learning (lines 16 and 17 in Algorithm 5).

4.5 Role-based Learning

In fully cooperative multi-agent settings with discrete action spaces, we leverage the state and action abstraction mechanisms described earlier to construct a hierarchical two-layer role-based learning framework, as illustrated in Figure 7. In this work, we define the set of abstract actions Z_a as the role set Ψ, where each role $\rho_i \in \Psi$ corresponds to an action

learning.pdf

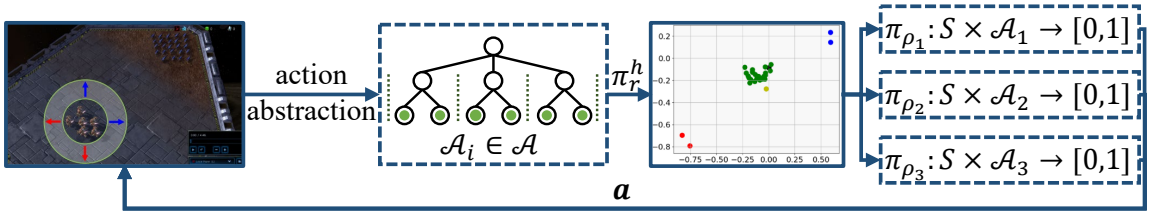


Figure 7: The multi-agent role-based learning in SIDM framework.

Algorithm 6: The Multi-Agent Role-Based Learning Method

1: **Input:** batch size n, update interval t_{up}, replay buffer $\mathcal{B}$
2: **Output:** hierarchical role policies π_h^r and $\{\pi_{\rho_1}, \pi_{\rho_2}, \dots\}$
3: **for** each episode **do**
4: **for** each timestep t **do**
5: **for** each agent $n_i \in \mathcal{N}$ **do**
6: $\rho_j \leftarrow$ select a role from the role space Ψ based on the high-level policy π_r^h
7: $a_i^t \leftarrow$ select an action based on the role policy π_{ρ_j} and environmental state s_t
8: **end for**
9: $\mathcal{B} \leftarrow \mathcal{B} \bigcup (s_t, \boldsymbol{a}_t, s_{t+1}, r_t)$
10: **if** $t \mod t_{up} == 0$ **then**
11: $S_t, A_t, S_{t+1}, R_t \leftarrow$ sample a batch from $\mathcal{B}$
12: $G_a \leftarrow$ construct the complete action graph from the variable A_t
13: $G_a^* \leftarrow$ filter out trivial edges from G_a
14: $T_a^* \leftarrow$ generate the optimal encoding tree
15: $Z_a \leftarrow$ aggregate actions to obtain abstract embeddings via Equation 15
16: $\Psi \leftarrow$ update the role set
17: $\mathcal{L}_{de}, \mathcal{L}_{marl} \leftarrow$ calculate the decoding loss and training loss
18: optimize the abstraction function and hierarchical policies by minimizing the losses $\mathcal{L}_{de}$ and $\mathcal{L}_{marl}$
19: **end if**
20: **end for**
21: **end for**

subspace $\mathcal{A}_i \subset \mathcal{A}$. The complete procedure for the proposed two-layer role-based learning framework is outlined in Algorithm 6.

At each timestep t, for each agent $n_i \in \mathcal{N}$, the high-level policy $\pi_r^h : \tau_i \times \Psi \to [0, 1]$ selects a role ρ_j and its corresponding action subspace $\mathcal{A}_j$ from the role set Ψ to complete the role assignment (line 6 in Algorithm 6). The low-level role policy $\pi_{\rho_j} : \mathcal{S} \times \mathcal{A}_j \to [0, 1]$ determines an action distribution over $\mathcal{A}_j$, from which it samples and executes an action $a_i^t \in \mathcal{A}_j$ given the global state $s_t \in \mathcal{S}$ (line 7 in Algorithm 6). The individual actions of all agents collectively form a joint action $\boldsymbol{a}_t$, which produces a joint reward r_t and leads to the subsequent global state s_{t+1}. To train the low-level role policies, we employ an existing MARL algorithm in which agents sharing the same role utilize a shared policy network. The associated training loss is denoted as $\mathcal{L}_{marl}$.

At regular update intervals t_{up}, we retrieve samples containing state-action transitions (S_t, A_t, S_{t+1}, R_t) of size n from the replay buffer $\mathcal{B}$ (line 11 in Algorithm 6). Subsequently, we apply the action abstraction mechanism to refine the sets Z_s and Ψ (lines 12 - 16 in Algorithm 6). Finally, we calculate the training losses $\mathcal{L}_{de}$ and $\mathcal{L}_{marl}$ to optimize the hierarchical policies (lines 17 and 18 in Algorithm 6).

4.6 Time Complexity Analysis

This subsection analyzes the time complexity of the abstraction mechanism (Subsection 4.1) and skill discovery (Subsection 4.3) in our SIDM framework to evaluate its computational feasibility. The total time complexity of the abstraction mechanism is given by $O(n^2 + n + m \cdot \log^2 n)$, where $n = |V|$ and $m = |E|$ denote the numbers of vertices and edges, respectively, in the state or action graph. During algorithm execution, n represents the batch size sampled from the replay buffer, while m denotes the number of edges retained after the sparse operation in Algorithm 1, which is approximately $\frac{kn}{2}$. Specifically, graph construction has a complexity of $O(n^2 + n)$, attributable to $O(n^2)$ for complete graph construction and $O(n)$ for filtering insignificant edges. According to the analysis (Pan et al., 2021), the process of optimizing a high-dimensional encoding tree through the *stretch* and *compress* operators incurs a time complexity of $O(m \cdot \log^2 n)$. The aggregation of a K-layer encoding tree with n leaves has a proven upper bound of $O(n)$. Since the number of abstract states in the transition graph does not exceed n, the time complexity of skill discovery is capped at $O(m \cdot \log^2 n + n)$.

5. Experimental Setup

To validate the performance advantage of our SIDM framework, we have conducted extensive comparative experiments in both single-agent decision-making and multi-agent collaboration scenarios. Specifically, we evaluate three key components of the SIDM framework: the state abstraction mechanism (SISA), the skill-based learning method (SISL), and the role-based learning method (SIRD). Each method is compared against state-of-the-art baselines using established and challenging benchmarks. Each experiment consists of ten independent runs using different random seeds to ensure fairness. We report the average reward to assess effectiveness and the standard deviation to measure stability over all episodes after convergence. Additionally, we measure efficiency by counting the environmental timesteps needed to reach specific rewards.

5.1 Benchmarks

5.1.1 State Abstraction

We evaluate the SISA mechanism for offline state abstraction in a visual Gridworld environment. Following the approach in Allen et al. (2021), each (x, y) coordinate in the 6×6 Gridworld is linked to a high-dimensional noisy image. During offline training of the state abstraction mechanism, the agent explores the environment by interacting with these images using a random policy with four directional actions, without access to actual grid positions. During DQN policy training (Mnih et al., 2015), the abstraction function, which maps original images to abstract state representations, remains fixed.

Following the offline evaluation of the SISA mechanism in Gridworld, we apply it in an online setting to various image-based continuous control tasks from the DeepMind Control Suite (DMControl) (Tunyasuvunakool et al., 2020), where both the abstraction mechanism and policy network are trained simultaneously. Specifically, our experiments focus on nine DMControl tasks: `ball_in_cup-catch`, `cartpole-swingup`, `cheetah-catch`, `finger-spin`, `reacher-easy`, `walker-walk`, `hopper-hop`, `hopper-stand`, and `pendulum-swingup`.

5.1.2 Skill-based Learning

We evaluate the skill-based learning method, SISL, in robotic control environments using the MuJoCo physics simulator (Todorov et al., 2012), including a bipedal robot (Gehring et al., 2021) and a 7-DoF fetch arm (Silver et al., 2018). For the bipedal robot experiments, we select six tasks requiring diverse skills: `Hurdles` (jumping), `Limbo` (torso control), `Stairs` (intricate foot manipulation), and `PoleBalance` (body balance). For the 7-DoF Fetch arm experiments, we select four downstream tasks: `Table Cleanup`, `Slippery Push`, `Pyramid Stack`, and `Complex Hook`. To emphasize efficient exploration, all tasks are designed with sparse rewards, which are awarded only when a goal or subgoal is achieved.

5.1.3 Role-based Learning

For multi-agent role-based learning, we evaluate the role-based method, SIRD, using the standard Centralized Training with Decentralized Execution (CTDE) benchmark in complex, high-control environments: the StarCraft II micromanagement (SMAC) suite (Samvelyan et al., 2019). In these micromanagement scenarios, each agent autonomously controls an allied unit based on local observations, while a built-in AI controls all enemy units. At each timestep, each agent selects an action from a discrete action space, including four-directional movement, stopping, executing a no-op, and selecting an enemy or ally unit to attack or heal. The more challenging maps, classified as hard and super-hard, present significant exploration challenges that require intricate collaborative strategies among the agents. Thus, we primarily concentrate on the performance of the SIRD method and other baseline approaches in the hard and super-hard maps. In summary, the multi-agent collaborative SMAC tasks consist of three difficulty levels: easy (`1c3s5z`, `2s3z`, `2c_vs_1sc`, and `10m_vs_11m`), hard (`2c_vs_64zg`, `3s_vs_5z`, `5m_vs_6m`, and `bane_vs_bane`), and super hard (`3s5z_vs_3s6z`, `corridor`, `MMM2`, and `27m_vs_30m`).

For each benchmark, we provide the task description along with detailed settings for observations, actions, and rewards, as presented in Appendix D.

5.2 Baselines

5.2.1 State Abstraction

For offline state abstraction, we select several baseline algorithms from the visual Gridworld environment, each designed to optimize distinct learning objectives. These baselines are detailed as follows:

• **PixelPred (Kaiser et al., 2019)**: A model-based RL approach that leverages video prediction to reduce environment interactions.

• **Autoenc (Lee et al., 2020a)**: An algorithm that decouples representation learning from policy learning by constructing a compact latent space.

• **Markov (Allen et al., 2021)**: A state abstraction method that preserves the Markov property using inverse model estimation and temporal contrastive learning.

• **IAEM (Zhu et al., 2022)**: A representation model that captures the invariance in action effects across states by explicitly utilizing action-effect relations.

For online state abstraction, we adopt state representation and data augmentation algorithms that have demonstrated superior performance in the DMControl Suite, as described in detail below:

• **RAD (Laskin et al., 2020b)**: A data augmentation approach that applies transformations such as random cropping and color jittering to visual inputs.

• **CURL (Laskin et al., 2020a)**: A contrastive learning-based method that extracts high-level features from raw pixels to enhance pixel-based control performance.

• **DBC (Zhang et al., 2020)**: A representation learning technique that utilizes bisimulation metrics to learn task-relevant state representations.

• **SAC-AE (Yarats et al., 2021b)**: A model-free RL approach that integrates auxiliary autoencoder-based losses to enhance representation learning.

• **DrQv2 (Yarats et al., 2021a)**: A model-free RL algorithm that leverages data augmentation to improve efficiency in visual continuous control.

• **Simsr (Zang et al., 2022)**: A method that learns robust state representations from image-based observations to achieve policy robustness.

• **CMID (Dunion et al., 2024)**: A method that learns disentangled representations by minimizing conditional mutual information between features.

5.2.2 Skill-based Learning

For skill-based learning, we integrate both skill-based and non-skill-based control methods, each demonstrating strong performance in robotic manipulation tasks. The baselines for the bipedal robot are detailed as follows:

• **SAC (Haarnoja et al., 2018)**: A model-free algorithm using the maximum entropy framework that combines off-policy updates with a stochastic actor-critic formulation.

• **Switching Ensemble (Nachum et al., 2019)**: A hierarchical RL framework that leverages hierarchy-inspired techniques instead of relying on rigidly imposed structures.

• **HIRO (Nachum et al., 2018)**: A hierarchical RL method that leverages off-policy experience for both higher- and lower-level policies.

• **HIDIO (Zhang et al., 2021)**: A hierarchical RL algorithm that learns task-agnostic options through self-supervised entropy minimization.

• **HSD-3 (Gehring et al., 2021)**: A hierarchical skill learning framework that automatically balances general and specific skills in an unsupervised manner.

For the 7-DOF robotic arm control task, we select high-performing methods from this benchmark and introduce two additional baselines, DSAA (Attali et al., 2022) and Louvain (Evans and Şimşek, 2023), that closely resemble our approach. This allows for a more direct comparison, further emphasizing the advantages of our skill-based learning method. These baselines are detailed as follows:

• **SPiRL (Pertsch et al., 2021)**: An RL method that learns a skill prior to offline experience, guiding the exploration of transferable skills.

• **BC+Fine-Tuning (Beeson and Montana, 2022)**: An offline approach that mitigates overestimation bias by behavioral cloning for stable fine-tuning and refined policies.

• **PaRRot (Singh et al., 2020)**: A pre-training method for RL that learns behavioral priors from past tasks, enabling rapid adaptation to robotic manipulation tasks.

• **Reskill (Rana et al., 2023)**: A skill-based RL approach that accelerates exploration by using state-conditioned generative models and low-level residual policy.

• **DSAA (Attali et al., 2022)**: A method for learning sparse, discrete state-action abstractions through successor representations and max-entropy regularization.

• **Louvain (Evans and Şimşek, 2023)**: A hierarchical skill learning method that automatically generates skill hierarchies based on modularity maximization.

For offline pretraining-based baselines (e.g., SPiRL), we follow the warm-start method (Zhou et al., 2025), which consists of pretraining, warmup, and online update phases, to enable efficient fine-tuning of the RL agent without retaining or jointly training on the offline dataset. The Louvain baseline, which is limited to discrete state spaces, is integrated into our hierarchical framework by applying Louvain clustering to our similarity-guided state graph to achieve multi-level skill discovery and skill-based learning.

5.2.3 Role-based Learning

For role-based learning, we evaluate both role-based and non-role-based approaches for multi-agent coordination, all of which achieve state-of-the-art performance in the SMAC benchmark. These baselines are detailed as follows:

• **IQL (Tampuu et al., 2017)**: A multi-agent extension of Deep Q-Learning that studies cooperation and competition between autonomous agents learning from visual input.

• **VDN (Sunehag et al., 2018)**: A cooperative MARL method that decomposes a team's value function into individual agent values, addressing challenges in partially-observable environments.

• **QMIX (Rashid et al., 2018)**: A value-based multi-agent method that combines centralized training with decentralized execution by enforcing monotonicity in the joint action values using a mixing network.

• **QPLEX (Wang et al., 2021a)**: A MARL method that improves scalability and stability by using a duplex dueling network architecture to efficiently factorize the joint value function.

• **QTRAN (Son et al., 2019)**: A MARL approach that generalizes value decomposition by proposing a new factorization method for joint action-value functions without restrictive structural constraints.

• **COMA (Foerster et al., 2018)**: A counterfactual multi-agent actor-critic method that uses a centralized critic and decentralized actors, addressing credit assignment by marginalizing out a single agent's action while keeping others fixed.

• **ACE (Li et al., 2023)**: A MARL algorithm that solves the non-stationarity problem using bidirectional action-dependent Q-learning, turning multi-agent decision-making into a single-agent process.

• **RODE (Wang et al., 2021b)**: A role-based RL method that discovers roles by clustering actions based on their environmental and inter-agent effects, leading to more efficient learning and policy generalization.

5.3 Experimental Setting

In our proposed SIDM framework, the maximum heights of encoding trees are set to $K = 2$ for undirected optimization in state/action abstraction and $K = 5$ for directed optimization in skill discovery.

For the SISA mechanism, we set a latent dimension of 2, a batch size of 2048, a learning rate of 0.003, and the Adam optimizer for training the DQN. We set a maximum episodic step of 1000, a batch size of 16, and a discount factor γ for the offline abstraction. In the online abstraction, we set a latent dimension of 50, a replay buffer size of $1e5$, a batch size of 128, a discount factor of 0.99, and the Adam optimizer. We use the Soft Actor-Critic (SAC) algorithm (Haarnoja et al., 2018) as the underlying single-agent RL method, which is integrated with various state abstraction approaches. For the state graph, we set the number of vertices to twice the batch size and determine the number of edges according to Algorithm 1.

For the SISL method, we adopt the standardized SAC algorithm within the corresponding state subspace to train the low-level option policy for each discovered option. For the high-level policy, we extract the abstract state with the highest probability within each state subspace to construct the set of termination states. We extend the SAC algorithm to the discrete termination set by inputting its continuous output into a Softmax layer. The resulting output is a probability distribution over the termination states. For all experiments, we use neural networks with 4 hidden layers, skip connections, and ReLU activations. During the training process, we use the Adam optimizer with a replay buffer size of $1e6$, a mini-batch size of 256, a learning rate of 0.001, and a discount factor of 0.99. Similarly, for the state graph, we set the number of vertices to twice the batch size and determine the number of edges according to Algorithm 1.

For the SIRD method, we share a trajectory encoding network with two fully connected layers and a GRU layer for each agent, followed by a linear network without hidden layers or activation functions, which serves as the role policy. The outputs of the role policies are fed into separate QMIX-style mixing networks (Rashid et al., 2020), each containing a 32-dimensional hidden layer with ReLU activation, to estimate global action values. For all SMAC experiments, the dimension of action representations is set to 20, the optimizer is set to RMSprop with a learning rate of 0.0005, and the discount factor is set to 0.99. For the action graph, we set the number of vertices to the number of enemies plus six general discrete actions, and the number of edges is automatically determined according to Algorithm 1.

5.4 Implementation Details

We implement the SISA mechanism using Python 3.8.15 and PyTorch 1.13.0, the SISL method using Python 3.9.1 and PyTorch 1.9.0, and the SIRD method using Python 3.5.2 and PyTorch 1.5.1. All experiments are conducted on five Linux servers, each equipped with an NVIDIA RTX A6000 GPU and an Intel i9-10980XE CPU clocked at 3.00 GHz.

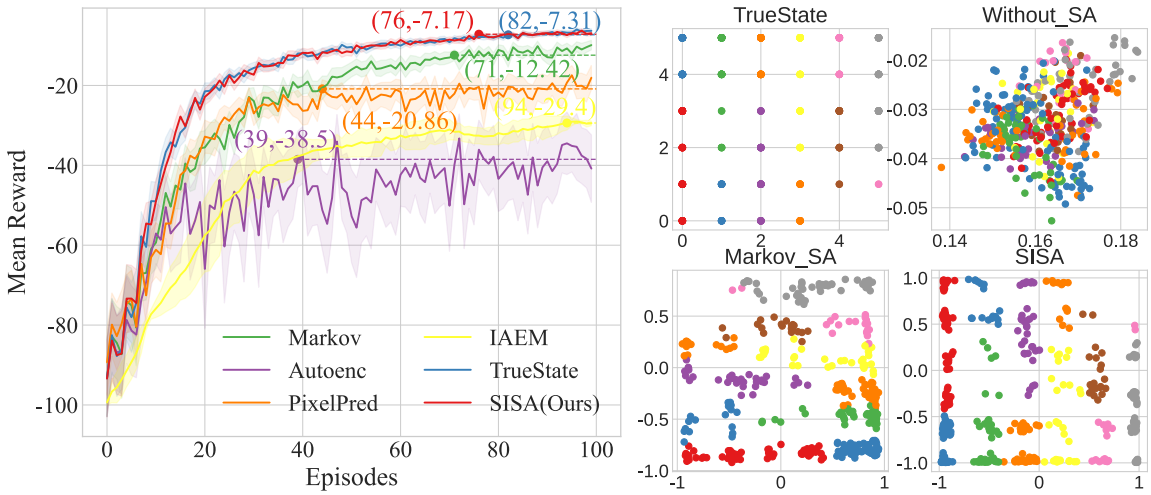


Figure 8: Average rewards over all episodes (left) and visualization of the 2-dimensional abstract state representations (right) for the navigation task within the 6×6 Gridworld environment.

6. Evaluation and Discussion

6.1 State Abstraction

We conduct comparative empirical experiments to demonstrate the benefits of our state abstraction mechanism SISA, focusing on offline abstractions in the visual Gridworld environment and online abstractions in continuous control tasks.

6.1.1 Offline Abstraction for Visual Gridworld

In the Gridworld environment, we illustrate the learning curves of SISA and other baselines for the offline navigation task in Figure 8. For reference, we also include the learning curve of the underlying DQN, labeled as TrueState, which is trained on ground-truth positions (x, y) without any observational noise or abstraction function. For each learning curve, we indicate the convergence point in parentheses within the figure. As shown in Figure 8 (left), SISA reaches convergence at 76 epochs with an average episodic reward of -7.17, achieving the smallest standard deviation and outperforming all baselines. Relative to the TrueState performance of $(82, -7.31)$, SISA maintains its advantages in both effectiveness and stability. Furthermore, Figure 8 (right) presents the 2-dimensional abstract representations of noisy observations for the 6×6 Gridworld, with different colors indicating ground-truth positions. Our abstraction mechanism more accurately reconstructs the relative positions of ground-truth states compared to the baselines. This success arises from an adaptive trade-off between filtering irrelevant details and preserving essential information, achieved through hierarchical aggregation within the optimal encoding tree.

6.1.2 Online Abstraction for Continuous Control

In the online abstraction experiments, we evaluate SISA and baseline methods across nine continuous tasks from the DMControl suite. Table 1 summarizes the average value and

Table 1: Summary of the mean episodic rewards for different tasks from DMControl: "average value ± standard deviation" and "average improvement (absolute value in percentage)". The best performance of our model is highlighted in bold, while the best performance among baselines is underlined.

Domain, Task	ball_in_cup-catch	cartpole-swingup	cheetah-run	finger-spin	reacher-easy
DBC	168.95 ± 84.76	317.74 ± 77.49	432.24 ± 181.43	805.90 ± 78.85	191.44 ± 69.07
SAC-AE	929.24 ± 39.14	839.23 ± 15.83	663.71 ± 9.16	898.08 ± 30.23	917.24 ± 38.33
RAD	937.97 ± 6.77	825.62 ± 9.80	802.53 ± 8.73	835.20 ± 93.26	908.24 ± 25.62
CURL	899.03 ± 30.61	824.46 ± 18.53	309.49 ± 8.15	949.57 ± 15.71	919.71 ± 28.03
Markov	919.10 ± 38.14	814.94 ± 17.61	642.79 ± 65.92	969.91 ± 8.41	806.34 ± 131.40
DrQv2	275.85 ± 44.87	573.10 ± 33.11	583.85 ± 11.52	629.23 ± 18.38	406.09 ± 66.09
Simsr	837.59 ± 18.43	803.25 ± 21.30	708.20 ± **2.81**	694.17 ± 42.84	883.52 ± 57.19
CMID	529.03 ± 37.44	817.49 ± 14.36	624.78 ± 15.72	843.74 ± 21.87	818.49 ± 33.07
$SISA_{pr}$	946.29 ± 8.63	858.21 ± 6.31	**806.67** ± 8.61	**970.45** ± 8.75	924.52 ± 19.04
SISA	**947.66 ± 7.03**	**861.37 ± 3.26**	803.32 ± 5.51	968.59 ± **6.54**	**941.71 ± 16.04**
Abs.(%) Avg. ↑	9.69(1.03)	22.14(2.64)	4.14(0.52)	0.54(0.06)	22.0(2.39)
Domain, Task	walker-walk	hopper-hop	hopper-stand	pendulum-swingup	average reward
DBC	331.97 ± 108.40	-	-	305.08 ± 86.78	284.55 ± 76.43
SAC-AE	895.33 ± 56.25	-	-	-	582.34 ± 25.07
RAD	907.08 ± 13.02	181.20 ± 1.80	891.87 ± 10.04	843.84 ± 8.99	792.61 ± 19.78
CURL	885.03 ± 9.88	-	-	-	541.58 ± 13.69
Markov	918.44 ± 12.58	184.66 ± 6.48	864.70 ± 34.28	162.58 ± 1.57	698.16 ± 35.15
DrQv2	588.19 ± 11.13	-	762.91 ± 11.31	821.35 ± 6.60	525.00 ± 24.10
Simsr	804.17 ± 13.77	-	771.89 ± 36.27	184.45 ± 7.23	640.09 ± 25.46
CMID	641.83 ± 27.15	-	173.44 ± 19.58	-	511.08 ± 20.77
$SISA_{pr}$	**921.64** ± 12.43	209.55 ± 6.46	893.54 ± **4.74**	839.19 ± 7.90	818.90 ± 9.21
SISA	919.78 ± **9.40**	**209.68 ± 6.23**	**900.45** ± 5.05	**851.94 ± 3.60**	**822.72 ± 6.96**
Abs.(%) Avg. ↑	3.20(0.35)	25.02(13.55)	8.58(0.96)	8.1(0.96)	30.11(3.80)

standard deviation of episodic rewards, excluding results with final rewards below 100.00. Our results show that SISA consistently outperforms the baselines across all DMControl tasks, achieving up to a 25.02 increase in mean episodic reward. This corresponds to a 13.55% improvement from 184.66 to 209.68 in the `hopper-hop` task. Moreover, SISA exhibits greater stability than other methods, with reduced standard deviations in six tasks. In the remaining tasks, SISA ranks among the lowest deviations, closely matching the top-performing baselines. Compared to our previous version, $SISA_{pr}$ (Zeng et al., 2023b), the current method achieves higher average rewards in six tasks and lower standard deviations in eight tasks. These results demonstrate that the undirected structural entropy optimization algorithm (Subsection 4.1) enhances both the effectiveness and stability of state abstraction.

To further analyze sample efficiency in the DMControl experiments, we define the target reward as 90% of SISA's final average value and report the timesteps required for both SISA and the best-performing baseline to reach this target. As shown in Figure 9, SISA reaches the mean episodic reward target in fewer steps than the baseline, demonstrating superior sample efficiency. Specifically, in the `hopper-stand` task, SISA improves sample efficiency by 64.86%, reducing the required timesteps from $222k$ to $78k$ to reach an episodic reward of 810.41. In summary, SISA have established a new state-of-the-art on DMControl, excelling in policy quality, stability, and sample efficiency in online learning with reward-based feedback. This success stems from our state abstraction mechanism, which optimally balances the compression of irrelevant information with the retention of essential features, enhancing both learning efficiency and policy performance.

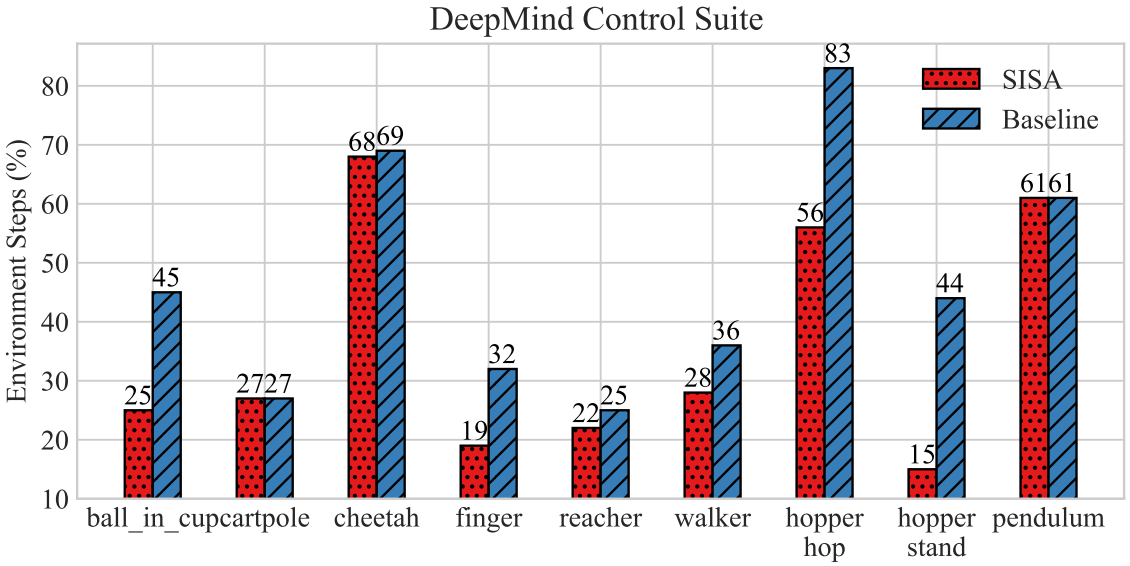


Figure 9: Sample-efficiency analysis of the SISA mechanism and baseline method in the DMControl continuous control tasks.

For each DMControl task, Figure 10 presents detailed learning curves for SISA and three leading baselines, showing the progression of mean episodic rewards and their convergence points throughout training. For example, in the `pendulum-swingup` task, SISA converges within $320,000$ timesteps and attains a mean reward of 851.94.

6.2 Skill-based Learning

In the bipedal robotic environment, we present the average rewards and standard deviations of SISL and other baselines after one million steps in Table 2. Notably, SISL consistently outperforms all baseline methods in each control task, achieving a maximum improvement of 18.75% in average reward, increasing from 11.2 to 13.3 in the `Hurdles` task. To further analyze training efficiency, we visualize the reward learning curves of SISL and the two best-performing baselines in the `Hurdles` and `Stairs` tasks. As shown in Figure 11, SISL consistently achieves a policy of comparable quality to all baselines while requiring fewer environment steps during training on both tasks. For instance, in the `Hurdles` task, to reach a reward of 12.96, the HSD-3 baseline requires 21.45 million environmental steps, while SISL achieves the same reward with only $985,000$ steps, highlighting the efficiency advantage of our approach.

Moreover, Figure 12 illustrates the skill selection process in SISL during a testing episode of the `Stairs` task, highlighting skill discovery dynamics and selection over time. During the episode, SISL discovers and selects skills with distinct temporal properties across different task phases. Specifically, walking upstairs requires regular control of the torso's X and Z positions, with occasional adjustments to the Y position and foot movements. Running forward follows a different pattern, primarily relying on the X position. To maintain balance while descending, the right foot is explicitly adjusted more frequently. SISL dynamically adapts skill discovery and selection to different environmental phases by minimizing structural entropy in directed abstract transitions, successfully completing the primary

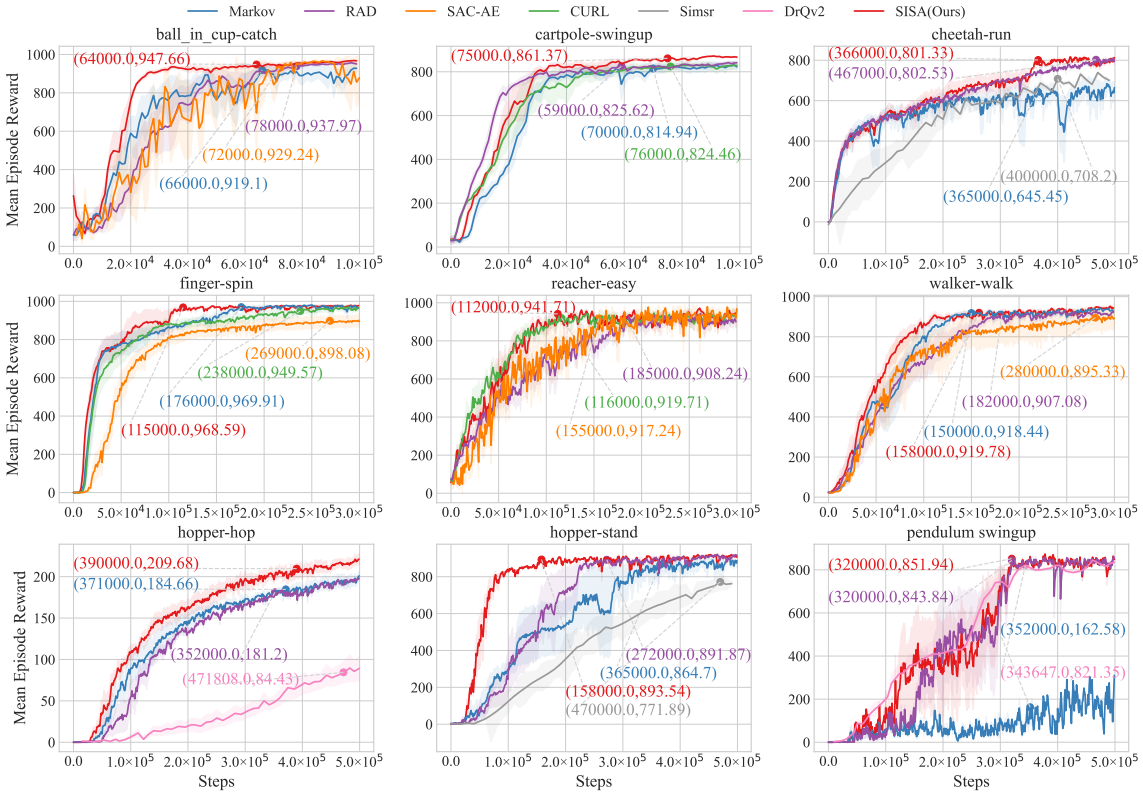


Figure 10: Learning curves of SISA and three leading baselines in various DM-Control tasks.

Table 2: Summary of the final performance across benchmark tasks using the bipedal robot: "average value ± standard deviation" and "average improvement (absolute value in percentage)". The best performance in each category is highlighted in bold, while the second-best performance is underlined.

Benchmark Task	Hurdles	Limbo	HurdlesLimbo	Stairs	Gaps	PoleBalance
SAC	-0.1 ± 6.2	$-0.1 \pm \underline{0.2}$	-0.1 ± 0.4	0.0 ± 4.8	-0.1 ± 0.5	231.5 ± 104.5
Switching Ensemble	-0.2 ± 3.0	-0.2 ± 4.3	-0.2 ± 3.6	1.1 ± 3.8	-0.2 ± 0.3	132.8 ± 230.1
HIRO-SAC	3.9 ± 1.6	1.1 ± 2.2	$3.5 \pm \underline{0.1}$	$0.0 \pm \underline{0.0}$	$\underline{0.0} \pm \underline{0.2}$	$96.4 \pm \underline{12.4}$
HIDIO	$-0.1 \pm \mathbf{0.1}$	$-0.1 \pm \mathbf{0.1}$	-0.2 ± 0.1	-0.2 ± 0.3	-0.2 ± 0.3	117.6 ± 33.8
HSD-3	$\underline{11.2} \pm 2.0$	$\underline{12.0} \pm 0.9$	$\underline{11.2} \pm 1.3$	$\underline{6.5} \pm 0.7$	-0.2 ± 8.9	$\underline{246.0} \pm 36.9$
SISL	$\mathbf{13.3} \pm \underline{0.9}$	$\mathbf{12.6} \pm 0.7$	$\mathbf{12.8 \pm 0.1}$	$\mathbf{7.0} \pm \underline{0.1}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{252.6 \pm 11.2}$
Abs.(%) Avg. ↑	2.1(18.75)	0.6(5.0)	1.6(14.29)	0.5(7.69)	0.0(0.0)	6.6(2.68)

task without task-specific prior knowledge. A video demonstration of SISL across multiple episodes and tasks is available on GitHub[2].

For the 7-DoF fetching benchmark, we compare SISL with baselines operating in either the original action space (PARROT and BC+FineTuning) or the skill space (SPiRL, Reskill, DSAA, and Louvain). Table 3 summarizes the average rewards and standard deviations after convergence across four control tasks, excluding final rewards below 10.00.

2. `https://selgroup.github.io/SIDM/`

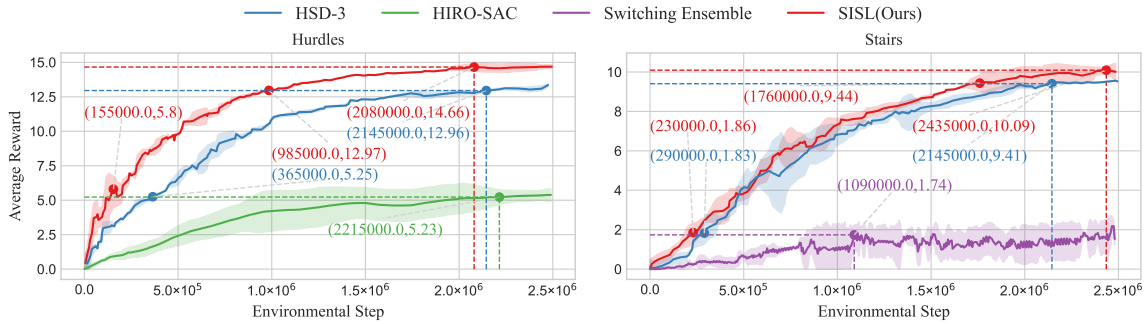


Figure 11: Efficiency comparison of our SISL and baseline methods in the bipedal robotic environment.

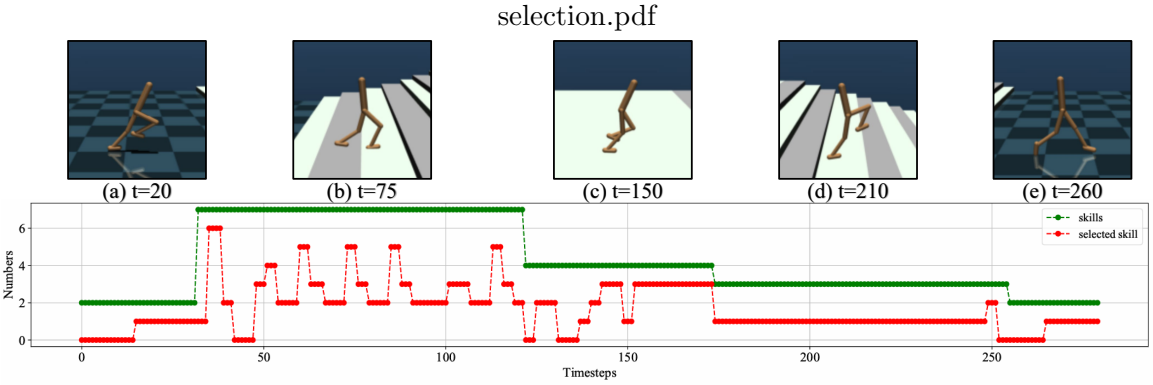


Figure 12: The skill discovery and selection during a testing episode of the `Stairs` task in our single-agent skill-based learning method.

SISL significantly outperforms all baselines in every benchmark task, achieving a maximum improvement of 32.70% from 42.42 to 56.29 in the `Slippery Push` task. In terms of learning stability, SISL exhibits minimal deviations in the `Pyramid Stack` and `Complex Hook` tasks and low deviations in `Table Cleanup` and `Complex Hook`, closely trailing the top-performing baseline, Reskill. Compared with the baseline DSAA, which also leverages abstract states for skill discovery, our SISL method demonstrates improved learning effectiveness, achieving an average improvement of 22.72 in average reward. This is because the skill hierarchy constructed by SISL captures more fine-grained temporal dynamics, leading to better agent exploration and policy optimization. Compared to Louvain, which is based on multi-level skill hierarchies, SISL achieves greater learning stability, reducing the standard deviation of final rewards by an average of 68.83%, from 3.16 to 0.99. This improvement stems from SISL's ability to adjust its skill hierarchy while mitigating observational noise through state abstraction, resulting in more stable learning trajectories and reduced performance fluctuations. Thus, while existing research shares technical similarities with this work in skill discovery via state abstraction or hierarchical skill construction, it further supports the validity of our approach. Moreover, guided by the structural information principle, we propose a unified hierarchical learning framework that spans from state abstraction to adaptive skill hierarchy construction, while effectively mitigating reliance on prior knowledge and reducing intrinsic observational noise in downstream tasks.

Table 3: Summary of the final performances across benchmark tasks with the 7-DOF robotic arms: "average value ± standard deviation" and "average improvement (absolute value in percentage)". The best performance in each category is highlighted in bold, while the second-best performance is underlined.

Benchmark Task	Fetch Table Cleanup	Fetch Slippery Push	Fetch Pyramid Stack	Fetch Complex Hock
SPiRL	11.33 ± 1.74	-	-	33.65 ± 3.34
BC + Fine-Tuning	-	14.81 ± 2.91	$15.43 \pm \underline{2.03}$	-
PARROT	-	32.10 ± 3.58	-	13.07 ± 2.89
Reskill	$\underline{35.48} \pm \mathbf{0.39}$	$42.42 \pm \mathbf{1.32}$	$\underline{16.91} \pm \underline{2.78}$	$\underline{58.93} \pm \underline{0.34}$
DSAA	21.47 ± 0.38	33.64 ± 1.72	10.58 ± 1.97	28.79 ± 3.02
Louvain	29.09 ± 2.48	$\underline{42.88} \pm 2.62$	13.28 ± 3.62	50.46 ± 3.92
SISL	$\mathbf{39.80} \pm \underline{0.71}$	$\mathbf{56.29} \pm \underline{1.43}$	$\mathbf{22.20 \pm 1.67}$	$\mathbf{67.06 \pm 0.13}$
Abs.(%) Avg. ↑	4.32(12.18)	13.41(32.27)	5.29(31.28)	8.13(13.80)

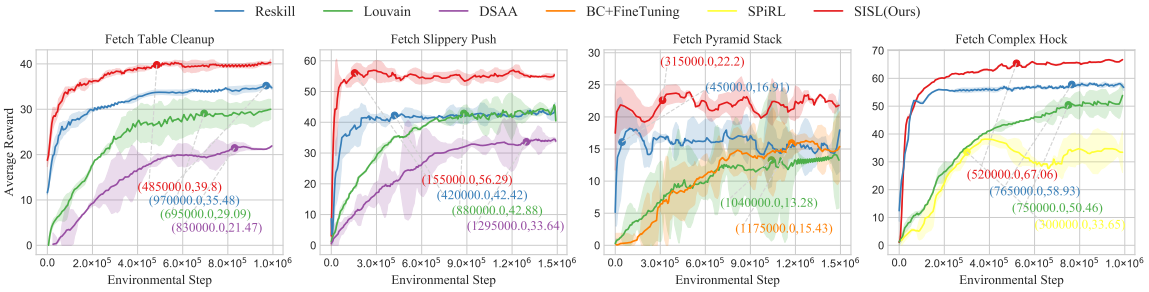


Figure 13: Learning curves of SISL and three leading baselines in various fetching tasks with the 7-DOF robotic arms.

Figure 13 presents detailed training learning curves for SISL and three leading baselines across each control task, showing the progression of mean episodic rewards and their convergence points. For example, in the `Table Cleanup` task, SISL converges within 485, 000 timesteps and attains a mean reward of 39.80.

6.3 Role-based Learning

In this subsection, we compare the SIRD method with state-of-the-art MARL algorithms across SMAC maps in the easy, hard, and super-hard categories. Table 4 summarizes the average win rates (above 10.00) and their standard deviations for each map category. SIRD outperforms all baseline algorithms, achieving up to a 5.19% improvement in average reward and a reduction of up to 88.26% in standard deviation, highlighting its performance advantages in both learning effectiveness and stability. By leveraging structural information principles, our action abstraction mechanism facilitates automatic role discovery, improving agent cooperation and reducing reliance on sensitive hyperparameters. These improvements are particularly evident in challenging exploration scenarios, such as the hard and super-hard maps.

Furthermore, we compare the SIRD method with baseline algorithms across all 14 SMAC maps to evaluate their overall performance. Figure 14 presents the average test win rate and the number of maps where each MARL algorithm leads in performance at different stages of policy learning. As shown in Figure 14 (left), SIRD surpasses all baselines and

Table 4: Summary of the test win rates under different map categories: "average value ± standard deviation" and "improvements/reductions (absolute value in percentage)". The best performance in each category is highlighted in bold, while the second-best performance is underlined.

Categories	Easy	Hard	Super Hard
COMA	16.67 ± 22.73	-	-
IQL	52.50 ± 40.69	73.44 ± 24.85	10.55 ± 18.49
VDN	85.01 ± 17.22	71.49 ± 18.78	71.10 ± 27.23
QMIX	$\underline{98.44} \pm \underline{2.10}$	87.11 ± 18.58	70.31 ± 38.65
QTRAN	64.69 ± 36.79	58.20 ± 45.37	16.80 ± 20.61
QPLEX	96.88 ± 5.04	$89.85 \pm \underline{11.35}$	84.77 ± 10.76
MAPPO	66.67 ± 35.35	61.72 ± 23.60	73.98 ± 16.45
RODE	93.47 ± 10.19	88.44 ± 20.96	$\underline{92.71} \pm \underline{9.20}$
ACE	-	$\underline{91.10} \pm 11.66$	88.15 ± 5.21
SIRD_{pr}	$98.61 \pm \mathbf{1.75}$	95.31 ± 6.63	95.71 ± 3.10
SIRD	$\mathbf{98.83} \pm 2.17$	$\mathbf{95.83 \pm 4.99}$	$\mathbf{97.50 \pm 1.08}$
Abs.(%) Avg. ↑	0.39(0.40)	4.73(5.19)	4.79(5.17)
Abs.(%) Dev. ↓	0.35(16.67)	6.36(56.04)	8.12(88.26)

achieves faster convergence. Remarkably, SIRD maintains the highest average test win rate throughout the last 60% of the learning process, achieving a final win rate of 96.7%, which exceeds the second-best (ACE at 92.73%) and the third-best (QPLEX at 90.99%) by 3.97% and 5.71%, respectively. This enhanced performance, particularly in policy quality and learning efficiency, stems from SIRD's effective exploration of action subsets identified via its action abstraction mechanism. Figure 14 (right) shows that SIRD achieves the best final policy in nearly half the maps (6 out of 14), significantly outperforming the baselines.

Figure 15 illustrates SIRD-driven multi-agent collaboration in the `1c3s5z` task, highlighting role variation and agent distribution throughout a testing episode. Compared to the role-based baseline RODE, SIRD dynamically adjusts its role set and action subspaces via action abstraction, without manual intervention, leading to improved performance. In summary, SIRD outperforms all MARL baselines in learning effectiveness and stability, establishing a new state-of-the-art on SMAC.

Figure 16 presents the training curves for SIRD and three leading baselines on each SMAC map, highlighting convergence points and showing standard deviations for each task. In the `MMM2` task, SIRD converges after $1,604,442$ timesteps and achieves an average win rate of 96.2%.

6.4 Generality Abilities

The proposed SIDM is a general framework and can be flexibly integrated with various single-agent and multi-agent RL algorithms, serving as a high-level method for skill and role discovery to enhance hierarchical learning performance.

For single-agent decision-making, we employ SAC (Haarnoja et al., 2018) and PPO (Schulman et al., 2017) as low-level RL algorithms in skill-based learning, forming the SL-

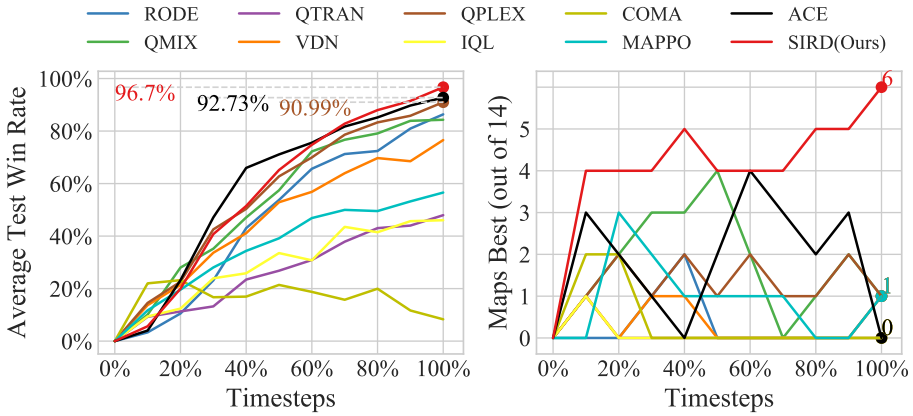


Figure 14: (left) The average test win rates across all 14 maps; (right) The number of maps (out of 14) where the algorithm achieves the highest average test win rate.

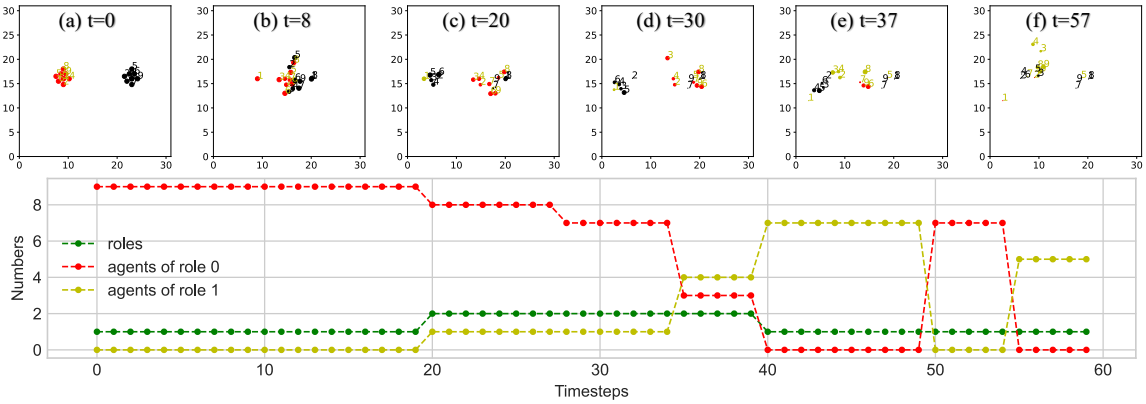


Figure 15: The role discovery and selection during a testing episode of the $1c3s5z$ map in our multi-agent role-based learning method.

SAC and SL-PPO variants. As shown in Figure 17, the SL-SAC and SL-PPO variants significantly outperform their original counterparts in terms of learning performance, particularly during training on the `Slippery Push` and `Pyramid Stack` tasks in the 7-DoF Fetching benchmark. For multi-agent coordination, we integrate the SIRD method with QMIX (Rashid et al., 2018) and QPLEX (Wang et al., 2021a) algorithms, resulting in the SI-QMIX and SI-QPLEX variants. As shown in Figure 18, SI-QMIX and SI-QPLEX outperform their original counterparts in terms of policy quality and sample efficiency, particularly on the `2c_vs_64zg` and `MMM2` maps in the SMAC benchmark. This leads to faster convergence and more effective multi-agent coordination during training.

These experimental results highlight the effectiveness of applying structural information principles to adaptively identify inherent hierarchical decision-making structures, forming the foundation for general skill-based and role-based learning methods.

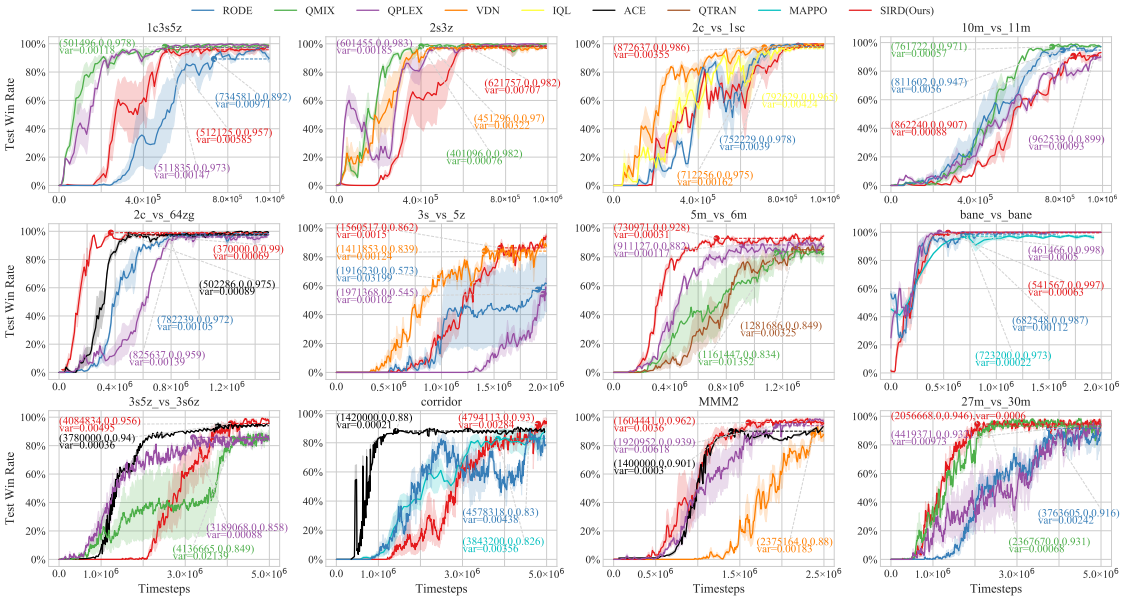
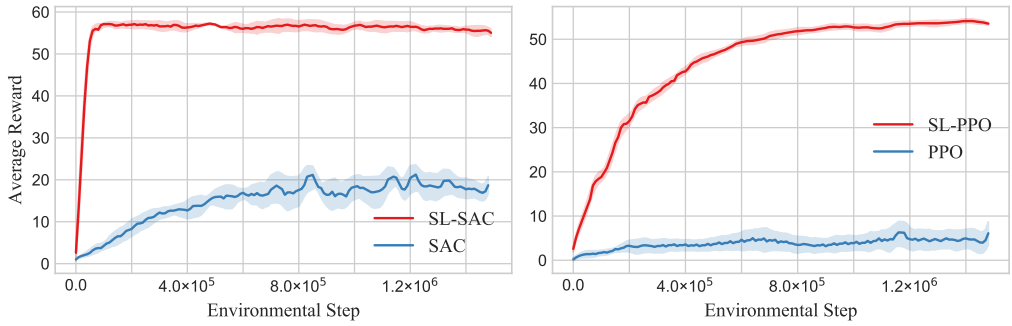


Figure 16: Learning curves of SIRD and three leading baselines in various SMAC maps.

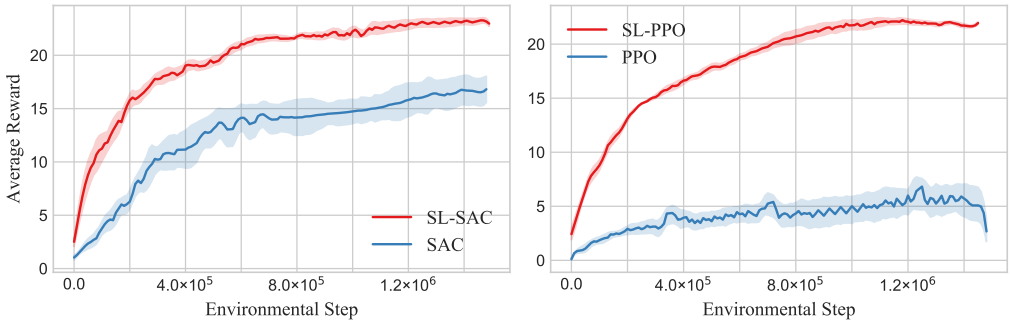
6.5 Ablation Studies

We evaluate the contribution of state abstraction (Subsection 4.1) and directed entropy optimization (Subsection 4.2) to the performance advantages of SIDM in single-agent skill-based learning. Specifically, we introduce two SISL variants—SISL-AS and SISL-DS—by disabling these components, respectively. In SISL-AS, the skill set is directly extracted from demonstration data and remains fixed throughout the learning process. In SISL-DS, the parameter h is set to K, and the set $\mathcal{K}_h$ is restricted to a single skill defined at the largest time scale. This configuration restricts the skill set to a single skill, which is insufficient for learning complex decision-making tasks. As shown in Figure 19, the significantly lower training performance of SISL-DS underscores the importance of directed structural entropy in capturing key transition patterns and constructing a skill hierarchy. The performance gap between SISL and SISL-AS highlights the benefits of adaptive skill discovery and observational noise mitigation through the state abstraction mechanism, as discussed in Subsection 6.2.

In multi-agent scenarios, we conduct ablation studies to validate the impact of the graph construction and edge filtration modules in role-based learning (Subsection 4.5) on the `2c_vs_64zg` and `1c3s5z` SMAC maps. We develop two simplified SIRD variants, SIRD-ST and SIRD-SP, by removing either the graph construction or edge filtration module to isolate their individual contributions. Specifically, SIRD-ST identifies roles by applying K-means clustering to the joint action space. In contrast, SIRD-SP directly optimizes the encoding tree of the complete action graph for role discovery. As shown in Figure 20, SIRD significantly outperforms SIRD-ST in both mean test win rate and policy stability, emphasizing the crucial role of graph construction in hierarchical learning. The comparison between SIRD and SIRD-SP suggests that edge filtration accelerates learning by reducing convergence timesteps without compromising performance.



(a) Fetch Slippery Push



(b) Fetch Pyramid Stack

Figure 17: Training curves of SISL integrated with single-agent RL methods, specifically SAC and PPO, in fetching tasks with the 7-DOF robotic arms.

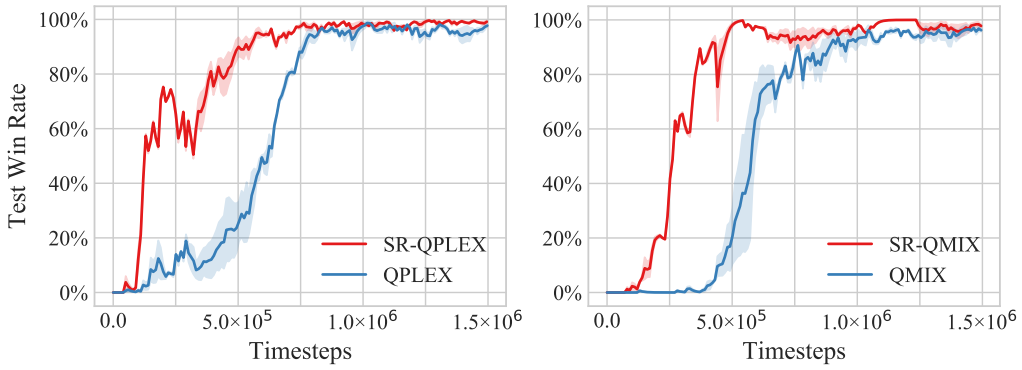
Table 5: Sensitivity analysis on alternative similarity metrics in the SISA method using the DMControl suite.

Task	cheetah-run	finger-spin	reacher-easy	walker-walk
SISA_{bm}	802.94 ± 7.28	968.96 ± 8.03	943.06 ± 15.82	921.61 ± 11.66
SISA_{sfs}	802.77 ± 8.04	968.14 ± 8.19	938.29 ± 13.80	920.71 ± 10.53
SISA_{hrd}	803.19 ± 6.91	970.03 ± 10.58	939.52 ± 18.29	923.08 ± 14.47
SISA_{crs}	802.59 ± 7.40	970.49 ± 8.95	941.33 ± 14.09	923.55 ± 13.70
SISA	803.32 ± 5.51	968.59 ± 6.54	941.71 ± 16.04	919.78 ± 9.40

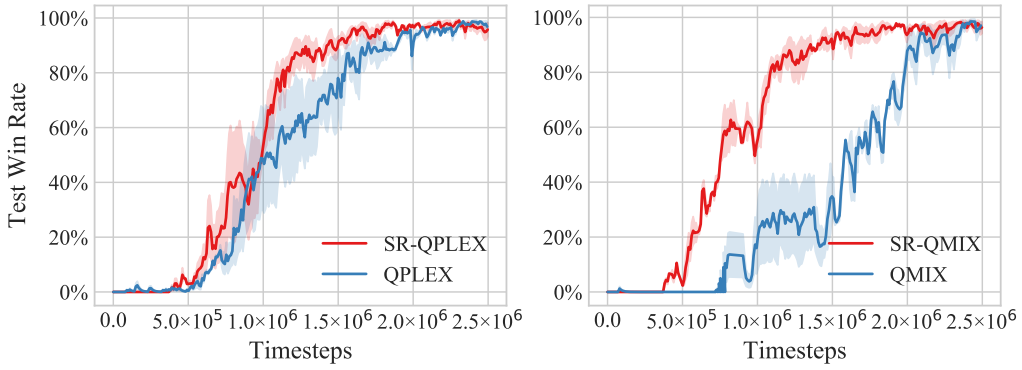
6.6 Sensitivity Analysis

In this subsection, we perform a sensitivity analysis on two essential parameters in the SIDM framework: the similarity metric in state abstraction and the encoding tree height in action abstraction.

In the single-agent scenario, we investigate the bisimulation metric (Castro, 2020), successor feature similarity (Hoang et al., 2021), Hilbert representation difference (Park et al., 2024), and contrastive representation similarity (Eysenbach et al., 2022) as alternative approaches for quantifying state similarity in state abstraction, resulting in the SISA variants SISAbm, SISAsfs, SISAhrd, and SISAcrs, respectively. For the SISA_{sfs} variant, we extract



(a) 2c_vs_64zg



(b) MMM2

Figure 18: Training curves of SIRD integrated with multi-agent algorithms, specifically QMIX and QPLEX, in SMAC maps.

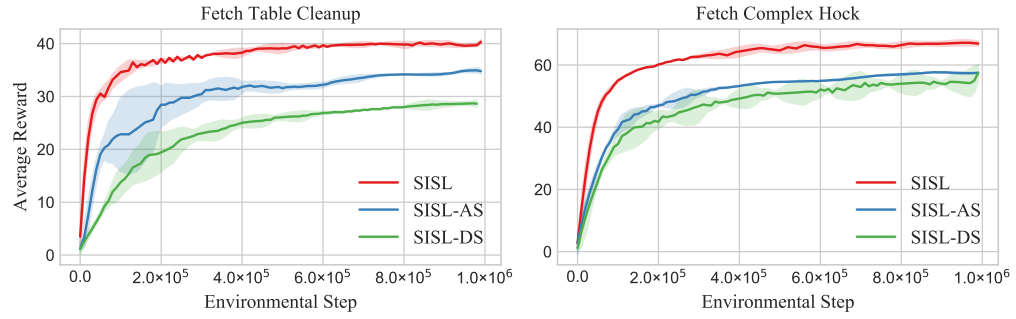


Figure 19: Ablation studies on state abstraction and directed optimization in single-agent, skill-based learning.

temporally aligned state-action pairs to learn their successor features, using the inner product of the resulting feature representations to quantify similarity. For the SISA_{hrd} variant, we calculate the ratio between the Hilbert representation difference and the maximum observed difference across the dataset as the state similarity. For the SISA_{crs} variant, we define

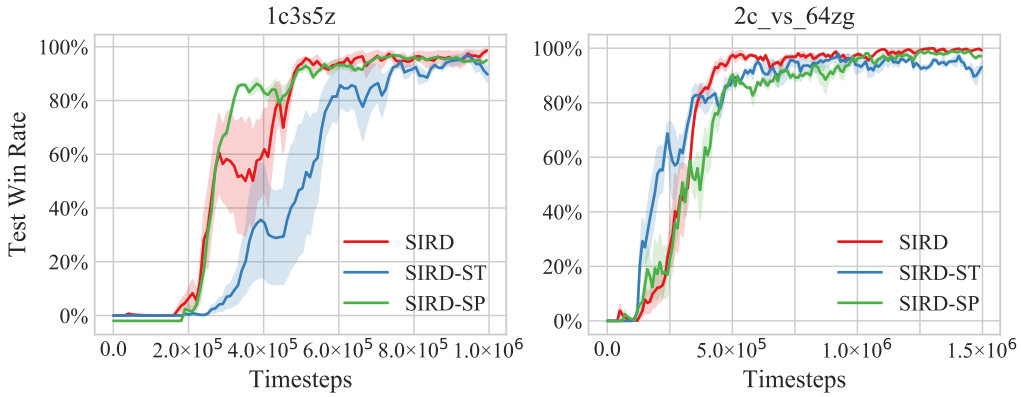


Figure 20: Ablation studies on graph construction and edge filtration in multi-agent, role-based learning.

Table 6: Sensitivity analysis on encoding tree height in SIRD method using the SMAC benchmark.

Category, Map	Easy Maps		Hard Maps		Super Hard Maps	
	1c3s5z	2s3z	2c_vs_64zg	3s_vs_5z	MMM2	27m_vs_30m
SIRD-2	95.72 ± 0.59	98.24 ± 0.71	99.04 ± 0.07	86.18 ± 0.15	96.20 ± 0.36	94.61 ± 0.06
SIRD-3	95.07 ± 0.76	97.60 ± 0.77	96.78 ± 0.25	84.53 ± 0.41	96.83 ± 0.19	95.47 ± 0.13
SIRD-4	94.83 ± 0.64	97.49 ± 0.80	95.26 ± 0.29	84.70 ± 0.44	96.69 ± 0.18	95.84 ± 0.18

positive and negative pairs for contrastive learning based on state community partitions to obtain state representations, and again use the inner product of these representations as the similarity measure. We evaluate the SISA mechanism and its variants on four DMControl tasks and summarize their average task rewards after convergence in Table 5. Across different similarity metrics, our state abstraction mechanism consistently achieves stable decision-making performance, demonstrating the generalization ability of our method and justifying the use of a simple similarity measure based on the Pearson correlation coefficient.

In the multi-agent collaboration setting, we adjust the encoding tree height parameter K in the action abstraction to 3 and 4. For each encoding tree, abstract actions are represented as the parent nodes of leaf nodes. We summarize SIRD's performance at different tree heights across four SMAC benchmark maps of varying difficulty in Table 6. The SIRD achieves the best performance at $K = 2$ in both relatively easy and hard tasks. This is because, in these scenarios, an overly complex role division is unnecessary, and a two-layer action abstraction is sufficient to fulfill task requirements. Increasing the tree height expands the candidate role set, making role-based learning more challenging. For super-hard tasks, deeper encoding tree structures better facilitate complex agent collaboration in these extremely difficult scenarios.

7. Conclusion

This paper proposes SIDM, a novel hierarchical learning framework designed to reduce reliance on prior knowledge and manual definitions in both skill-based and role-based learning. It also addresses undirected limitations of current structural information principles.

Specifically, SIDM introduces an adaptive abstraction mechanism that extracts abstract state and action representations from historical interaction trajectories. Directed structural entropy is formally defined and optimized to capture transition dynamics between abstract states, enabling the discovery of hierarchical skills. Building on these foundations, we develop skill-based learning methods for single-agent decision-making and role-based strategies for multi-agent collaboration, resulting in substantial performance improvements.

Comprehensive evaluations on challenging benchmarks demonstrate that SIDM significantly and consistently outperforms state-of-the-art baselines in terms of learning effectiveness, stability, and efficiency. These comparative results highlight the importance of adaptively balancing the compression of irrelevant information with the retention of essential features, as well as dynamically capturing the temporal hierarchy of skills and roles to enhance hierarchical decision-making. Furthermore, sensitivity analysis and ablation studies emphasize the generalizability of the SIDM framework and the individual contributions of the abstraction mechanism and direct entropy optimization.

Our objective is to provide an innovative and unified perspective on leveraging structural information in state-action trajectories to uncover hierarchical learning structures, ultimately improving decision-making performance. In future work, we plan to explore deeper encoding trees for hierarchical state-action abstraction and evaluate SIDM in more complex environments.

Acknowledgments

The corresponding authors are Hao Peng and Angsheng Li. This work has been supported by NSFC through grants 62322202, 62441612 and 62432006, Local Science and Technology Development Fund of Hebei Province Guided by the Central Government of China through grant 246Z0102G, the "Pioneer" and "Leading Goose" R&D Program of Zhejiang" through grant 2025C02044, National Key Laboratory under grant 241-HF-D07-01, Hebei Natural Science Foundation through grant F2024210008, and Yunnan Provincial Major Science and Technology Special Plan Project 202502AD080012.

Appendix A. Framework Details

A.1 Notations

Table 7: Glossary of Notations.

Notation	Description
$\mathcal{M}_s; \mathcal{M}_m; \mathcal{M}_\phi$	Single-agent, Multi-agent, and Abstract Markov decision processes
$n; n_i; \mathcal{N}$	Batch Size; Single agent; Agent set
$\mathcal{S}; \mathcal{A}; \mathcal{Z}$	State space; Action space; Abstract space
$S; A; Z$	State variable; Action variable; Abstract Variable
$s; a; z$	Single state; Single action; Single abstract element
$r; \mathcal{R}; \mathcal{R}_\phi; \mathcal{R}^{in}$	Reward; Reward function; Abstract reward function; Intrinsic reward function
$\mathcal{P}; \mathcal{P}_\phi$	Transition function; Abstract transition function
γ	Discount factor
$\pi; \pi_\rho; \pi_\kappa; \pi^h$	Agent policy; Role policy; Skill policy; High-level policy
$\rho; \Psi; t_i$	Single role; Role space; Subtask
$\kappa; \mathcal{K}$	Single skill/option; Skill space
τ	Individual action-observation history
$\mathcal{I}; \mathcal{T}$	Initiation set; Termination condition
$G; G_{dir}$	Homogeneous weighted undirected and directed graphs
$G_s; G_a; G^*$	State graph; Action graph; Sparse graph
$v; d_v; V$	Single vertex; Vertex degree; Vertex set
$e; E; E_{dir}$	Single edge; Set of undirected edges; Set of directed edges
$w; W; W_{dir}$	Edge weight; Weight functions for undirected and directed edges
$\lambda; \alpha; \nu; T$	Root node; Tree node; Leaf node; Encoding tree
$V_\alpha; \mathcal{V}$	Vertex subset; Volume term
H	Structural entropy
$L; K$	Number of children node; Maximal height of encoding tree
$\eta; U_i$	Optimization operator; Node set locating specific layer
$f; \mathcal{C}$	Embedding function; State or action correlation
$h_s; h_\alpha; h_T; h$	State representation; Node representation; Tree height; Skill parameter
$\pi_s; \pi_e$	Stationary distribution; Eigenvector
$\mu_{mg}; \mu_{cb}; \mu$	Added nodes by *merge* and *combine* operations; Average value
k; k	Filtration parameter; Index variable
$\mathcal{B}; \mathcal{L}$	Replay buffer; Training loss

Appendix B. Detailed Derivations

B.1 Derivations of Directed Structural Entropy Variations

Because of the properties of the encoding tree T_{dir}, for each non-leaf node α, it holds for its child nodes' corresponding vertex sets that:

$$\bigcap_{i=1}^{L_\alpha} V_{\alpha_i} = \emptyset, \quad \bigcup_{i=1}^{L_\alpha} V_{\alpha_i} = V_\alpha. \tag{31}$$

Equations 18 and 19 can be rewritten as follows:

$$\begin{aligned}\mathcal{V}_\alpha &= \sum_{v_i \in V} \sum_{v_j \in V_\alpha} \left[\pi_s(v_i) \cdot W'_{dir}(v_i, v_j)\right] \\ &= \sum_{k=1}^{L_\alpha} \left[\sum_{v_i \in V} \sum_{v_j \in V_{\alpha_k}} \left[\pi_s(v_i) \cdot W'_{dir}(v_i, v_j)\right] \right] \\ &= \sum_{k=1}^{L_\alpha} \mathcal{V}_{\alpha_k},\end{aligned} \tag{32}$$

$$\begin{aligned}g_\alpha &= \sum_{v_i \notin V_\alpha} \sum_{v_j \in V_\alpha} \left[\pi_s(v_i) \cdot W'_{dir}(v_i, v_j)\right] \\ &= \sum_{k=1}^{L_\alpha} \left[\sum_{v_i \notin V_\alpha} \sum_{v_j \in V_{\alpha_k}} \left[\pi_s(v_i) \cdot W'_{dir}(v_i, v_j)\right] \right] \\ &= \sum_{k=1}^{L_\alpha} \left[\sum_{v_i \notin V_{\alpha_k}} \sum_{v_j \in V_{\alpha_k}} \left[\pi_s(v_i) \cdot W'_{dir}(v_i, v_j)\right] - \sum_{l=1, l\neq k}^{L_\alpha} \left[\sum_{v_i \in V_{\alpha_l}} \sum_{v_j \in V_{\alpha_k}} \left[\pi_s(v_i) \cdot W'_{dir}(v_i, v_j)\right] \right] \right] \\ &= \sum_{k=1}^{L_\alpha} \left[g_{\alpha_k} - \sum_{l=1, l\neq k}^{L_\alpha} g_{\alpha_l, \alpha_k} \right] \\ &= \sum_{i=1}^{L_\alpha} g_{\alpha_i} - \sum_{i \neq j}^{L_\alpha} g_{\alpha_i, \alpha_j}.\end{aligned} \tag{33}$$

As shown in Figure 21, one *merge* (η_{mg}) operation on non-leaf sibling nodes α and β is executed as follows:

$$\begin{cases} \mu_{mg}^- = \alpha^- = \beta^-, & \\ \alpha_i{}^- = \mu_{mg} & 1 \le i \le L_\alpha, \\ \beta_i{}^- = \mu_{mg} & 1 \le i \le L_\beta, \end{cases} \tag{34}$$

where μ_{mg} is the added tree node via the *merge* operation. Before the *merge* operation, the entropy sum of nodes α, β, and their child nodes is:

$$H^{T_{dir}}(G'_{dir}; \alpha) + H^{T_{dir}}(G'_{dir}; \beta) = -\frac{g_\alpha}{\mathrm{vol}(G'_{dir})} \cdot \log_2 \frac{\mathcal{V}_\alpha}{\mathcal{V}_{\alpha^-}} - \frac{g_\beta}{\mathrm{vol}(G'_{dir})} \cdot \log_2 \frac{\mathcal{V}_\beta}{\mathcal{V}_{\beta^-}}, \tag{35}$$

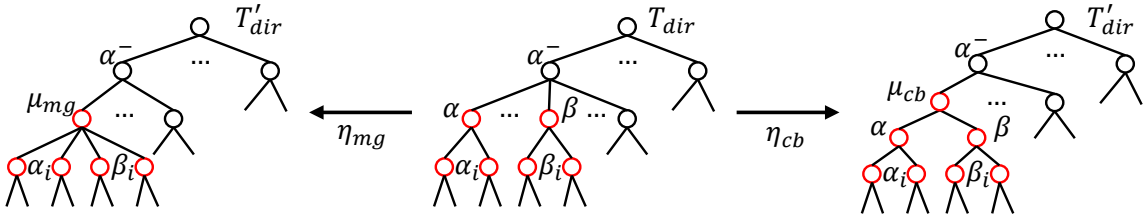


Figure 21: The *merge* and *combine* operations on sliding tree nodes.

$$\sum_i^{L_\alpha} H^{T_{dir}}(G'_{dir};\alpha_i)+\sum_i^{L_\beta} H^{T_{dir}}(G'_{dir};\beta_i)=\sum_i^{L_\alpha}\left[-\frac{g_{\alpha_i}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\alpha_i}}{\mathcal{V}_\alpha}\right]+\sum_i^{L_\beta}\left[-\frac{g_{\beta_i}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\beta_i}}{\mathcal{V}_\beta}\right]. \tag{36}$$

After the *merge* operation, their entropy sum is given by:

$$H^{T'_{dir}}(G'_{dir};\mu_{mg})=-\frac{g_{\mu_{mg}}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_{\alpha^-}}, \tag{37}$$

$$\sum_i^{L_\alpha} H^{T'_{dir}}(G'_{dir};\alpha_i)+\sum_i^{L_\beta} H^{T'_{dir}}(G'_{dir};\beta_i)=\sum_i^{L_\alpha}\left[-\frac{g_{\alpha_i}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\alpha_i}}{\mathcal{V}_{\mu_{mg}}}\right]+\sum_i^{L_\beta}\left[-\frac{g_{\beta_i}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\beta_i}}{\mathcal{V}_{\mu_{mg}}}\right]. \tag{38}$$

Therefore, the entropy variation $\Delta_{mg}(T_{dir},\alpha,\beta)$ is calculated as follows:

$$\sum_i^{L_\alpha} H^{T_{dir}}(G'_{dir};\alpha_i)-\sum_i^{L_\alpha} H^{T'_{dir}}(G'_{dir};\alpha_i)=\sum_i^{L_\alpha}\left[-\frac{g_{\alpha_i}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\alpha}\right]=-\frac{\sum_i^{L_\alpha} g_{\alpha_i}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\alpha}, \tag{39}$$

$$\sum_i^{L_\beta} H^{T_{dir}}(G'_{dir};\beta_i)-\sum_i^{L_\beta} H^{T'_{dir}}(G'_{dir};\beta_i)=\sum_i^{L_\beta}\left[-\frac{g_{\beta_i}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\beta}\right]=-\frac{\sum_i^{L_\beta} g_{\beta_i}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\beta}, \tag{40}$$

$$\begin{aligned} H^{T'_{dir}}(G'_{dir};\mu_{mg}) &= -\frac{g_{\mu_{mg}}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_{\alpha^-}} \\ &= \frac{g_{\alpha,\beta}+g_{\beta,\alpha}-g_\alpha-g_\beta}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_{\alpha^-}}, \end{aligned} \tag{41}$$

$$\begin{aligned} &H^{T_{dir}}(G'_{dir};\alpha)+H^{T_{dir}}(G'_{dir};\beta)-H^{T'_{dir}}(G'_{dir};\mu_{mg})= \\ &\frac{g_\alpha}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\alpha}+\frac{g_\beta}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\beta}+\frac{g_{\alpha,\beta}+g_{\beta,\alpha}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\mu_{mg}}}, \end{aligned} \tag{42}$$

$$\begin{aligned}\Delta_{mg}(T_{dir},\alpha,\beta) &= \frac{g_\alpha - \sum_i^{L_\alpha} g_{\alpha_i}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\alpha} + \frac{g_\beta - \sum_i^{L_\beta} g_{\beta_i}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\beta} + \frac{g_{\alpha,\beta}+g_{\beta,\alpha}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\mu_{mg}}}\\ &= \frac{g_{\alpha,\beta}+g_{\beta,\alpha}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\mu_{mg}}} - \frac{\sum_{i\neq j}^{L_\alpha} g_{\alpha_i,\alpha_j}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\alpha} - \frac{\sum_{i\neq j}^{L_\beta} g_{\beta_i,\beta_j}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{mg}}}{\mathcal{V}_\beta}.\end{aligned} \tag{43}$$

On the other hand, as shown in Figure 21, one *combine* (η_{cb}) operation on sibling nodes α and β is executed as follows:

$$\mu_{cb}{}^- = \alpha^-, \quad \alpha^- = \mu_{cb}, \quad \beta^- = \mu_{cb}, \tag{44}$$

where μ_{cb} is the added tree node via the *combine* operation. After the *combine* operation, the entropy sum of these nodes is given by:

$$H^{T'_{dir}}(G'_{dir};\mu_{cb}) = -\frac{g_{\mu_{cb}}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\mu_{cb}}}{\mathcal{V}_{\alpha^-}}, \tag{45}$$

$$H^{T'_{dir}}(G'_{dir};\alpha) + H^{T'_{dir}}(G'_{dir};\beta) = -\frac{g_\alpha}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_\alpha}{\mathcal{V}_{\mu_{cb}}} - \frac{g_\beta}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_\beta}{\mathcal{V}_{\mu_{cb}}}, \tag{46}$$

$$\sum_i^{L_\alpha} H^{T'_{dir}}(G'_{dir};\alpha_i) + \sum_i^{L_\beta} H^{T'_{dir}}(G'_{dir};\beta_i) = \sum_i^{L_\alpha} H^{T_{dir}}(G'_{dir};\alpha_i) + \sum_i^{L_\beta} H^{T_{dir}}(G'_{dir};\beta_i). \tag{47}$$

Therefore, the entropy variation $\Delta_{cb}(T_{dir},\alpha,\beta)$ is calculated as follows:

$$\begin{aligned}\Delta_{cb}(T_{dir},\alpha,\beta) &= \left[H^{T_{dir}}(G'_{dir};\alpha) + H^{T_{dir}}(G'_{dir};\beta)\right] - \left[H^{T'_{dir}}(G'_{dir};\mu_{cb}) + H^{T'_{dir}}(G'_{dir};\alpha) + H^{T'_{dir}}(G'_{dir};\beta)\right]\\ &= \frac{g_\alpha}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\mu_{cb}}} + \frac{g_\beta}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\mu_{cb}}} - \frac{g_{\mu_{cb}}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\mu_{cb}}}\\ &= \frac{g_\alpha + g_\beta - g_{\mu_{cb}}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\mu_{cb}}}\\ &= \frac{g_{\alpha,\beta}+g_{\beta,\alpha}}{\mathrm{vol}(G'_{dir})}\cdot\log_2\frac{\mathcal{V}_{\alpha^-}}{\mathcal{V}_{\mu_{cb}}}.\end{aligned} \tag{48}$$

Appendix C. Theoretical Proofs

C.1 Proof of Proposition 1

Proof We provide two distinct proofs for this theorem, one using the **Perron-Frobenius theorem** and another using the **Markov Chain fundamental theorem**.

C.1.1 Proof Based on Perron-Frobenius Theorem

For this non-negative and irreducible matrix A'_{dir} where each row sums to 1, the Perron-Fronbenius theorem ensures:

- There exists an eigenvalue 1 and a corresponding eigenvector π_e such that:

$$A'_{dir}\pi_e = \pi_e. \tag{49}$$

- The eigenvalue 1 is simple (i.e., it has algebraic multiplicity 1).
- The corresponding eigenvector π_e can be chosen to have strictly positive components.
- The absolute value of any other eigenvalue is less than 1.

Since the eigenvalue 1 is simple and the corresponding eigenvector π_e has strictly positive components, we can derive the unique stationary distribution π_s by:

$$\pi_s = \frac{\pi_e}{\sum_{v\in V}\pi_e(v)}. \tag{50}$$

To verify that π_s is indeed a stationary distribution, we need to show that it remains unchanged by the application of A'_{dir}:

$$A'_{dir}\pi_s = A'_{dir}\frac{\pi_e}{\sum_{v\in V}\pi_e(v)} = \frac{A'_{dir}\pi_e}{\sum_{v\in V}\pi_e(v)} = \frac{\pi_e}{\sum_{v\in V}\pi_e(v)} = \pi_s. \tag{51}$$

Because G'_{dir} is strongly connected and A'_{dir} is a stochastic matrix, any initial distribution $\pi^{(0)}$ will converge to the stationary distribution π_s under repeated application of A'_{dir}:

$$\lim_{k\to\infty}(\pi^{(0)}(A'_{dir})^k) = \pi_s. \tag{52}$$

This convergence is guaranteed by the fact that all other eigenvalues of A'_{dir} have magnitudes less than 1, leading to their contributions diminishing to zero as k increases.

Therefore, the stationary distribution π_s exists and is unique, and it corresponds to the eigenvector associated with the eigenvalue 1 of the adjacency matrix A'_{dir}.

C.1.2 Proof Based on Markov Chain Theorem

We interpret the weighted adjacency matrix A'_{dir} as the transition probability matrix P of a Markov chain. Since each row of A'_{dir} sums to 1, it satisfies the conditions of a right stochastic matrix.

Because the graph G'_{dir} is strongly connected, the Markov chain is irreducible, meaning that there exists a path between any two states in the chain. Additionally, since all transition

probabilities are positive, the chain is aperiodic, ensuring that no strict cycle behavior prevents convergence to a unique stationary distribution.

By the fundamental theorem of Markov chains, an irreducible and aperiodic Markov chain with a finite state space has a unique stationary distribution π_s, which satisfies:

$$\pi_s P = \pi_s. \tag{53}$$

Expanding this equation using the transition matrix $P = A'_{dir}$, we get:

$$\pi_s A'_{dir} = \pi_s. \tag{54}$$

Thus, π_s is a left eigenvector of A'_{dir} associated with the eigenvalue 1.

To ensure that π_s represents a valid probability distribution, we normalize it so that:

$$\sum_{v \in V} \pi_s(v) = 1. \tag{55}$$

Moreover, the Perron-Frobenius theorem guarantees that for an irreducible, non-negative stochastic matrix A'_{dir}, the eigenvalue 1 is simple (having algebraic multiplicity 1), and its corresponding eigenvector π_s has strictly positive components.

Since the Markov chain is irreducible and aperiodic, it converges to the stationary distribution π_s regardless of the initial distribution $\pi^{(0)}$:

$$\lim_{k \to \infty} \pi^{(0)} (A'_{dir})^k = \pi_s. \tag{56}$$

This confirms that π_s is the unique stationary distribution and corresponds to the unique eigenvector of A'_{dir} associated with eigenvalue 1.

■

C.2 Proof of Theorem 2

Proof We analyze the structure of the discovered skill set $\mathcal{K}_h$ when $h = K$. In this case, the skill set is defined as:

$$\mathcal{K}_h = \{\langle \mathcal{I}_{\kappa_i}, \pi_{\kappa_i}, \mathcal{T}_{\kappa_i} \rangle \mid \alpha_i \in U_h^z\} = \{\langle \mathcal{I}_{\kappa_1}, \pi_{\kappa_1}, \mathcal{T}_{\kappa_1} \rangle \mid \alpha_1 = \lambda\}, \tag{57}$$

where λ is the root node in the optimal encoding tree T^*_{dir}, corresponding to the entire set Z_s of abstract states.

For the discovered skill κ_1, we explicitly define:

- The initiation set $\mathcal{I}_{\kappa_1}$ consists of all abstract states except the one with the highest stationary probability:

$$\mathcal{I}_{\kappa_1} = \{z_i^s \mid z_i^s \neq \arg\max_{z_j^s \in Z_s} \pi_s(z_j^s)\}. \tag{58}$$

- The skill policy maximizes an intrinsic reward $\mathcal{R}^{in}$ defined over state transitions:

$$\pi^*_{\kappa_1} = \arg\max_{\pi_{\kappa_1}} \mathbb{E}_{\mathcal{P}} \left[\sum \gamma^t \mathcal{R}^{in}(z_i^s, z_j^s) \right]. \tag{59}$$

- The termination condition is satisfied when the state with the highest probability is reached:

$$\mathcal{T}_{\kappa_1}(z_i^s) = \begin{cases} 1 & \text{if } \arg\max_{z_j^s \in Z_s} \pi_s(z_j^s) = z_i^s, \\ 0 & \text{otherwise.} \end{cases} \tag{60}$$

We now define the eigenoption κ_e associated with the largest eigenvalue of the adjacency matrix A'_{dir}. Eigenoptions are derived from the principal eigenvector π_e of A'_{dir}, which satisfies:

$$A'_{dir}\pi_e = \pi_e. \tag{61}$$

To learn the eigenoption's option policy, we define the intrinsic reward function $\mathcal{R}^{in}$ based on the eigenvector:

$$\mathcal{R}_e^{in}(z_i^s, z_j^s) = \pi_e(z_i^s) - \pi_e(z_j^s). \tag{62}$$

This ensures that the option policy moves toward the most "important" state in the eigenvector ranking. The eigenoption is entirely defined by:

- Initiation set:

$$\mathcal{I}_{\kappa_e} = \{z_i^s \mid z_i^s \neq \arg\max_{z_j^s \in Z_s} \pi_e(z_j^s)\}. \tag{63}$$

- Option policy:

$$\pi^*_{\kappa_e} = \arg\max_{\pi_{\kappa_e}} \mathbb{E}_{\mathcal{P}} \left[\sum \gamma^t \mathcal{R}_e^{in}(z_i^s, z_j^s) \right]. \tag{64}$$

- Termination condition:

$$\mathcal{T}_{\kappa_e}(z_i^s) = \begin{cases} 1 & \text{if } \arg\max_{z_j^s \in Z_s} \pi_e(z_j^s) = z_i^s, \\ 0 & \text{otherwise.} \end{cases} \tag{65}$$

According to Appendix C.1, for the irreducible and stochastic matrix A'_{dir}, we know:

$$\pi_s(z_i^s) = \pi_e(z_i^s), \tag{66}$$

$$\mathcal{R}^{in}(z_i^s, z_j^s) = \mathcal{R}_e^{in}(z_i^s, z_j^s). \tag{67}$$

Thus, since both κ_1 and κ_e share the same initiation set, option policy, and termination condition, we conclude:

$$\kappa_1 = \kappa_e. \tag{68}$$

This proves that the only skill discovered at $h = K$ is the eigenoption associated with the largest eigenvalue. ■

Appendix D. Environment Details

D.1 Visual Gridworld

The visual gridworld environment consists of a 6×6 grid, where an agent can navigate using four discrete actions: up, down, left, and right. Each position on the grid corresponds to a unique state, which is represented to the agent in visual form. Specifically, the agent's (x, y) position is encoded into a one-hot image representation, where each grid cell is depicted as a 3×3 pixel patch within an 18×18 image. The center pixel of the corresponding patch is activated, and the image is subsequently smoothed using a truncated Gaussian kernel to produce a more realistic visual input. To introduce variability and challenge the agent's perception, per-pixel noise sampled from another truncated Gaussian distribution is added to the image.

During training, a single grid position is randomly designated as the goal state for each random seed. The agent receives a reward of -1 at every timestep until it reaches the goal state, after which a new episode begins with the agent placed in a randomly selected location that is not the goal. This setup encourages the agent to learn efficient navigation strategies that minimize cumulative negative reward.

D.2 DMControl Suite

The DeepMind Control Suite (Tunyasuvunakool et al., 2020) is a comprehensive benchmarking platform for reinforcement learning algorithms, featuring a diverse array of continuous control tasks that simulate various physical systems. Each task is designed with distinct state and action spaces, as well as task-specific reward functions. Together, these components define the environment's dynamics and the agent's learning objectives, enabling rigorous evaluation and comparison of reinforcement learning algorithms on complex control problems.

The state spaces include critical physical parameters such as positions, velocities, and angular velocities, providing agents with essential information for decision-making. The action spaces define how agents can influence the environment, typically through torques applied to joints or forces exerted on bodies. Reward functions are tailored to incentivize specific behaviors, such as balancing a pole, reaching a target, walking stably, hopping forward, catching a ball, or swinging up and stabilizing a pendulum.

D.3 Robotic Control Environment

This benchmark includes a suite of bipedal locomotion (Gehring et al., 2021) and 7-degree-of-freedom (7-DoF) robotic manipulation (Silver et al., 2018) tasks, designed to evaluate the adaptability and robustness of reinforcement learning algorithms under dynamic and variable conditions.

In the bipedal locomotion scenarios, robots are initialized at predefined positions and configurations. To simulate real-world uncertainties, joint positions are perturbed with noise sampled uniformly from the interval $[-0.1, 0.1]$, while joint velocities are perturbed with Gaussian noise scaled by 0.1. These perturbations are consistent with those used in standard MuJoCo benchmark tasks. Each environment provides three distinct observation modalities: (1) proprioceptive states, which include internal robot states such as joint angles

and velocities; (2) task-specific observations, representing additional sensory inputs relevant to the task; and (3) goal state measurements, which indicate the desired outcome and are primarily used to compute relative goals in low-level policy inputs. Episodes typically last for 1000 interaction steps, unless terminated earlier due to entry into invalid states, as defined by the specific robot's configuration.

The manipulation tasks, based on a 7-DoF robotic arm, involve diverse object interaction challenges. Each task introduces unique physical or dynamical variations to evaluate the generalization capabilities of learned policies:

• Slippery Push: The robot must push a block to a target location on a low-friction surface, increasing the difficulty of precise control.

• Table Cleanup: A rigid tray is introduced into the environment, requiring the robot to place a block into it while navigating around new obstacles.

• Pyramid Stack: The robot must stack a small red block onto a larger blue block, demanding accurate positioning and stability.

• Complex Hook: The robot uses a hook to manipulate objects that are otherwise unreachable, with added difficulty from random object shapes and irregularities on the table surface.

Each task is episodic, with predefined step limits (e.g., 50 or 100 steps), and sparse rewards provided only upon successful task completion.

D.4 SMAC Benchmark

The StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) is a benchmark suite designed to evaluate cooperative multi-agent reinforcement learning (MARL) algorithms. Built upon the StarCraft II game engine, SMAC presents a series of micromanagement scenarios where each agent controls an individual unit and operates under partial observability and decentralized execution constraints.

In SMAC, agents receive local observations that include information about nearby allies and enemies, such as relative positions, health status, and available actions. The discrete action space includes movement in cardinal directions, attacking specific targets, and halting. The environment's dynamics are governed by the underlying StarCraft II physics and game rules, introducing complexities such as terrain advantages, unit types, and attack ranges. Each scenario is designed to assess various aspects of cooperative behavior, such as focus fire, unit positioning, and retreat strategies. The reward structure typically assigns positive rewards for eliminating enemy units and negative rewards for allied losses, thereby encouraging agents to develop effective combat tactics.

References

David Abel. A theory of abstraction in reinforcement learning. *ArXiv Preprint ArXiv:2203.00397*, 2022.

David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. State abstractions for lifelong reinforcement learning. In *ICML*, pages 10–19. PMLR, 2018.

David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L Littman, and Lawson LS Wong. State abstraction as compression in apprenticeship learning. In *AAAI*, volume 33, pages 3134–3142, 2019.

Cameron Allen, Neev Parikh, Omer Gottesman, and George Konidaris. Learning markov state abstractions for deep reinforcement learning. *NeurIPS*, 34:8229–8241, 2021.

Amnon Attali, Pedro Cisneros-Velarde, Marco Morales, and Nancy M Amato. Discrete state-action abstraction via the successor representation. *ArXiv Preprint ArXiv:2206.03467*, 2022.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *NeurIPS*, 30, 2017.

Alex Beeson and Giovanni Montana. Improving td3-bc: Relaxed policy constraint for offline learning and stable online fine-tuning. In *Offline RL Workshop: Offline RL as a"Launchpad"*, 2022.

Richard Bellman. A markovian decision process. *JMM*, pages 679–684, 1957.

Noélie Bonjean, Wafa Mefteh, Marie-Pierre Gleizes, Christine Maurel, and Frédéric Migeon. *Handbook on agent-oriented design processes*. Springer, 2014.

Eamonn Butler. *The condensed wealth of nations*. Centre for Independent Studies, 2012.

Yuwei Cao, Hao Peng, Zhengtao Yu, and Philip S. Yu. Hierarchical and incremental structural entropy minimization for unsupervised social event detection. In *AAAI*, volume 38, pages 8255–8264, 2024.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *AAAI*, volume 34, pages 10069–10076, 2020.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Matheus Ribeiro Furtado de Mendonça, Artur Ziviani, and André da Motta Salles Barreto. Abstract state transition graphs for model-based reinforcement learning. In *BRACIS*, pages 115–120. IEEE, 2018.

Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah Hanna, and Stefano Albrecht. Conditional mutual information for disentangled representations in reinforcement learning. *NeurIPS*, 36, 2024.

Joshua B Evans and Özgür Şimşek. Creating multi-level skill hierarchies in reinforcement learning. *NeurIPS*, 36:48472–48484, 2023.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *NeurIPS*, 35:35603–35620, 2022.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI*, pages 2974–2982, 2018.

Jonas Gehring, Gabriel Synnaeve, Andreas Krause, and Nicolas Usunier. Hierarchical skills for efficient exploration. *NeurIPS*, 34:11553–11564, 2021.

Deborah M Gordon. The organization of work in social insect colonies. *Nature*, 380(6570): 121–124, 1996.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, pages 1861–1870. PMLR, 2018.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *NeurIPS*, 35:26091–26104, 2022.

Christopher Hoang, Sungryull Sohn, Jongwook Choi, Wilka Carvalho, and Honglak Lee. Successor feature landmarks for long-horizon goal-conditioned reinforcement learning. *NeurIPS*, 34:26963–26975, 2021.

Marcus Hutter. Extreme state aggregation beyond markov decision processes. *TCS*, 650: 73–91, 2016.

Raphaël Jeanson, Penelope F Kukuk, and Jennifer H Fewell. Emergence of division of labour in halictine bees: contributions of social interactions and behavioural variance. *Animal Behaviour*, 70(5):1183–1193, 2005.

Yuu Jinnai, Jee Won Park, David Abel, and George Konidaris. Discovering options for exploration by minimizing cover time. In *ICML*, pages 3130–3139. PMLR, 2019.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model based reinforcement learning for atari. In *ICLR*, 2019.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, pages 5639–5650. PMLR, 2020a.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *NeurIPS*, 33:19884–19895, 2020b.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *NeurIPS*, 33:741–752, 2020a.

Seungjae Lee, Jigang Kim, Inkyu Jang, and H Jin Kim. Dhrl: A graph-based approach for long-horizon and sparse hierarchical reinforcement learning. *NeurIPS*, 35:13668–13678, 2022.

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In *ICLR*, 2019.

Youngwoon Lee, Jingyun Yang, and Joseph J Lim. Learning to coordinate manipulation skills via skill behavior diversification. In *ICLR*, 2020b.

Kemas Muslim Lhaksmana, Yohei Murakami, and Toru Ishida. Role-based modeling for designing agent behavior in self-organizing multi-agent systems. *IJSEKE*, 28(01):79–96, 2018.

Angsheng Li and Yicheng Pan. Structural information and dynamical complexity of networks. *IEEE TIT*, 62(6):3290–3339, 2016.

Angsheng Li, Xianchen Yin, and Yicheng Pan. Three-dimensional gene map of cancer cell types: Structural entropy minimisation principle for defining tumour subtypes. *Scientific Reports*, 6:1–26, 2016.

Angsheng Li, Xianchen Yin, Bingxiang Xu, Danyang Wang, Jimin Han, Yi Wei, Yun Deng, Ying Xiong, and Zhihua Zhang. Decoding topologically associating domains with ultra-low resolution hi-c data by graph structural entropy. *Nature Communications*, 9:1–12, 2018.

Chuming Li, Jie Liu, Yinmin Zhang, Yuhong Wei, Yazhe Niu, Yaodong Yang, Yu Liu, and Wanli Ouyang. Ace: Cooperative multi-agent q-learning with bidirectional action-dependency. In *AAAI*, volume 37, pages 8536–8544, 2023.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*, pages 157–163. Elsevier, 1994.

Yiwei Liu, Jiamou Liu, Zijian Zhang, Liehuang Zhu, and Angsheng Li. Rem: From structural entropy to community structure deception. *NeurIPS*, 32, 2019.

Marlos C Machado, Andre Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation. *JMLR*, 24(80): 1–69, 2023.

Sridhar Mahadevan and Mauro Maggioni. Value function approximation with diffusion wavelets and laplacian eigenfunctions. *NeurIPS*, 18, 2005.

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *JMLR*, 8(10), 2007.

Kenneth Marino, Abhinav Gupta, Rob Fergus, and Arthur Szlam. Hierarchical rl using an ensemble of proprioceptive periodic policies. In *ICLR*, 2018.

Paul Mattes, Rainer Schlosser, and Ralf Herbrich. Hieros: Hierarchical imagination on structured state space sequence world models. In *ICML*, pages 35079–35103. PMLR, 2024.

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. In *ICLR*, 2018.

Josh Merel, Arun Ahuja, Vu Pham, Saran Tunyasuvunakool, Siqi Liu, Dhruva Tirumala, Nicolas Heess, and Greg Wayne. Hierarchical visuomotor control of humanoids. In *ICLR*, 2019.

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: Reusable neural controllers for vision-guided whole-body tasks. *TOG*, 39(4):39–1, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *NeurIPS*, 31, 2018.

Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *ArXiv Preprint ArXiv:1909.10618*, 2019.

Yicheng Pan, Feng Zheng, and Bingchen Fan. An information-theoretic perspective of hierarchical clustering. *ArXiv Preprint ArXiv:2108.06036*, 2021.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *ICML*, pages 39737–39761, 2024.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *CoRL*, pages 188–204. PMLR, 2021.

Doina Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

Washington Ramos, Michel Silva, Edson Araujo, Victor Moura, Keller Oliveira, Leandro Soriano Marcolino, and Erickson R Nascimento. Text-driven video acceleration: A weakly-supervised reinforcement learning method. *TPAMI*, 45(2):2492–2504, 2022.

Krishan Rana, Ming Xu, Brendan Tidd, Michael Milford, and Niko Sünderhauf. Residual skill policies: Learning an adaptable skill-based action space for reinforcement learning for robotics. In *Conference on Robot Learning*, pages 2095–2104. PMLR, 2023.

Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, pages 4292–4301, 2018.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *AAMAS*, pages 2186–2188, 2019.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv Preprint ArXiv:1707.06347*, 2017.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *ArXiv Preprint ArXiv:1812.06298*, 2018.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *ICLR*, 2020.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *ICML*, pages 5887–5896, 2019.

Changyin Sun, Wenzhang Liu, and Lu Dong. Reinforcement learning with task decomposition for cooperative multiagent systems. *TNNLS*, 32(5):2054–2065, 2020.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *AAMAS*, pages 2085–2087, 2018.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *AI*, 112(1-2):181–211, 1999.

Richard S Sutton, Marlos C Machado, G Zacharias Holland, David Szepesvari, Finbarr Timbers, Brian Tanner, and Adam White. Reward-respecting subtasks for model-based reinforcement learning. *AI*, 324:104001, 2023.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS One*, 12:e0172395, 2017.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*, volume 31, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012.

Giang Truong, Huu Le, Erchuan Zhang, David Suter, and Syed Zulqarnain Gilani. Unsupervised learning for maximum consensus robust fitting: A reinforcement learning approach. *TPAMI*, 2022.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. Dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, and etc. Alphastar: Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. In *ICLR*, pages 1–27, 2021a.

Tonghan Wang, Heng Dong, Victor R. Lesser, and Chongjie Zhang. Roma: Multi-agent reinforcement learning with emergent roles. In *ICML*, pages 9876–9886, 2020.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. In *ICLR*, pages 1–24, 2021b.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. Bayesian policy search for multi-agent role discovery. In *AAAI*, pages 624–629, 2010.

Michael J. Wooldridge, Nicholas R. Jennings, and David Kinny. The gaia methodology for agent-oriented analysis and design. *AAMAS*, 3(3):285–312, 2000.

Junran Wu, Xueyuan Chen, Ke Xu, and Shangzhe Li. Structural entropy guided graph hierarchical pooling. In *ICML*, pages 24017–24030. PMLR, 2022.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *ICLR*, 2021a.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *AAAI*, volume 35, pages 10674–10681, 2021b.

Hongyu Zang, Xin Li, and Mingzhong Wang. Simsr: Simple distance-based state representations for deep reinforcement learning. In *AAAI*, volume 36, pages 8997–9005, 2022.

Xianghua Zeng, Hao Peng, and Angsheng Li. Effective and stable role-based multi-agent collaboration by structural information principles. In *AAAI*, volume 37, pages 11772–11780, 2023a.

Xianghua Zeng, Hao Peng, Angsheng Li, Chunyang Liu, Lifang He, and Philip S Yu. Hierarchical state abstraction based on structural information principles. In *IJCAI*, pages 4549–4557, 2023b.

Xianghua Zeng, Hao Peng, and Angsheng Li. Adversarial socialbots modeling based on structural information principles. In *AAAI*, volume 38, pages 392–400, 2024.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *ICLR*, 2020.

Jesse Zhang, Haonan Yu, and Wei Xu. Hierarchical reinforcement learning by discovering intrinsic options. In *ICLR*, 2021.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *JMLR*, 25(1-67):1, 2024.

Ming Zhou, Ziyu Wan, Hanjing Wang, Muning Wen, Runzhe Wu, Ying Wen, Yaodong Yang, Yong Yu, Jun Wang, and Weinan Zhang. Malib: A parallel framework for population-based multi-agent reinforcement learning. *JMLR*, 24(150):1–12, 2023.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning should not retain offline data. In *ICLR*, 2025.

Zheng-Mao Zhu, Shengyi Jiang, Yu-Ren Liu, Yang Yu, and Kun Zhang. Invariant action effect model for reinforcement learning. In *AAAI*, volume 36, pages 9260–9268, 2022.