ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240741

Exploiting Hierarchical Symmetry in Multi-Agent Reinforcement Learning

Yongkai Tian[a], Xin Yu[a], Yirong Qi[a], Li Wang[b], Pu Feng[a], Wenjun Wu[a,b], Rongye Shi[a,b] and Jie Luo[a,*]

[a]State Key Laboratory of Complex & Critical Software Environment, Beihang University, Beijing, China
[b]Institute of Artificial Intelligence, Beihang University, Beijing, China

Abstract. Achieving high sample efficiency is a critical research area in reinforcement learning. This becomes extremely difficult in multi-agent reinforcement learning (MARL), as the capacity of the joint state and action space grows exponentially with the number of agents. The reliance of MARL solely on exploration and trial-and-error, without incorporating prior knowledge, exacerbates the issue of low sample efficiency. Currently, introducing symmetry into MARL is an effective approach to address this issue. Yet the concept of hierarchical symmetry, which maintains symmetry across different levels of a multi-agent system (MAS), has not been explored in existing methods. This paper focuses on multi-agent cooperative tasks and proposes a method incorporating hierarchical symmetry, termed the Hierarchical Equivariant Policy Network (HEPN) which is $O(n)$-equivariant. Specifically, HEPN utilizes clustering to perform hierarchical information extraction in MAS, and employs graph neural networks to model agent interactions. We conducted extensive experiments across various multi-agent tasks. The results indicate that our method achieves faster convergence speeds and higher convergence rewards compared to baseline algorithms. Additionally, we have deployed our algorithm in a physical multi-robot system, confirming its effectiveness in real-world environments. Supplementary materials are available at https://yongkai-tian.github.io/HEPN/.

1 Introduction

In recent years, the development of multi-agent reinforcement learning (MARL) has achieved notable success in sequential decision-making tasks, such as multiplayer games [16, 29, 32], biology [33], traffic control [2, 39], and multi-robot systems [3, 4, 8, 18]. However, despite its wide applicability and success, MARL continues to confront a persistent challenge of low sample efficiency. As the number of agents increases, the capacity of the joint state and action spaces expands exponentially. Consequently, MARL requires a substantial amount of samples to gradually optimize policy in these large state-action spaces, leading to the issue of low sample efficiency [20, 27]. This problem primarily arises because MARL relies solely on trial-and-error, neglecting the use of prior knowledge.

Enhancing the sample efficiency of MARL fundamentally entails achieving more effective learning with limited interactions [34]. Integrating prior knowledge can significantly boost sample efficiency by reducing the size of the solution space, thereby accelerating the training process [27]. Prior knowledge in multi-agent systems (MAS) predominantly refers to symmetries, which are typically implemented through three approaches: data augmentation, loss function design, and network architecture design [12]. We refer to the first two methods as soft constraint method, while the third is termed hard constraint method. Data augmentation leverages symmetries to generate additional training data, thereby reducing the number of interactions with the environment during training [34, 35, 36]. Furthermore, researchers have introduced symmetry biases into loss functions to guide models in learning the symmetries inherent in MAS, thus accelerating the training process. Additionally, network structures can be designed to inherently possess symmetries, such as permutation invariance [11, 15] and rotational symmetry [28].

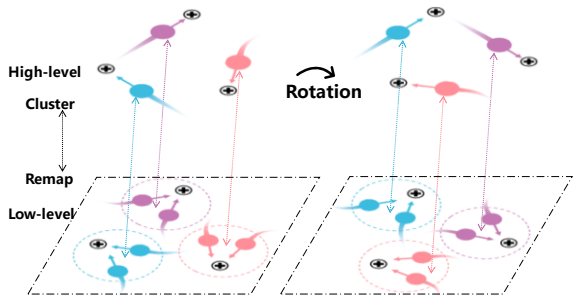


Figure 1. Example of rotation hierarchical equivariance in MAS. Rotating a system, whether at low or high level, the actions of agents also rotate.

However, soft constraint methods, which rely on guidance from generated data or loss functions to learn the intrinsic properties of the system, are heavily dependent on the diversity and quality of data. Furthermore, both soft and hard constraint methods overlook the inherent hierarchical structure within MAS. As shown in Figure 1, we illustrate the hierarchical structure presented in MAS. By modeling the MAS as a graph, agents with similarities can be regarded as a subsystem, forming a high-level system. In this high-level system, more efficient message passing and better learning of global information can be achieved [10], which are crucial for improving cooperation among agents [7]. The current limitation is that messages can only be passed between agents and cannot be inferred and aggregated in a hierarchical manner, which poses particular problems for multi-agent cooperative tasks. Therefore, how to utilize the inherent hierarchical structure and its symmetry within MAS remains a challenge, particularly crucial for multi-agent cooperative tasks.

In this paper, we focus on multi-agent cooperative tasks, where the absence of prior knowledge and hierarchical structure leads to inefficient learning. We propose a novel policy network structure based on Symmetric Dec-POMDP (S-Dec-POMDP) [34] called the Hierarchi-

* Jie Luo is the Corresponding Author. Email: luojie@buaa.edu.cn.

cal Equivariant Policy Network (HEPN) to achieve sample-efficient learning. Firstly, inspired by equivariant hierarchical graph neural networks [10], we design HEPN to explore and learn the hierarchical structure of MAS while ensuring strict symmetry properties. Furthermore, inspired by structural entropy [14], we propose partition loss to better learn the hierarchical structure within MAS. Finally, we validate the effectiveness of our algorithm on various multi-agent cooperative tasks including Rendezvous, Pursuit, and Resource Collection and also conduct verification in real world environments. The contributions of this paper can be summarized as follows:

- We propose the Hierarchical Equivariant Policy Network (HEPN), which utilizes the hierarchical symmetry in MAS to enhance the efficiency of MARL algorithms.
- We propose a partition loss aimed at better uncovering the hierarchical structure within MAS.
- We evaluate the performance of HEPN across several multi-agent cooperative tasks. Experimental results indicate that HEPN achieves faster convergence speeds and higher convergence rewards, thereby validating its effectiveness.
- We deploy HEPN in a physical multi-robot environment, confirming its effectiveness in real world.

2 Related Work

2.1 Hierarchy in MARL

In the realm of MARL, there has been some explorations in leveraging hierarchical structures of systems to enhance effectiveness of algorithms. In HAMA [22], the authors design a hierarchical graph attention network to model the hierarchical relationships between agents in either cooperative or competitive scenarios. SISA [37] introduces an unsupervised, adaptive hierarchical state clustering method. This method minimizes structural entropy to cluster agent states hierarchically, effectively filtering out irrelevant environmental information. In the domain of mean-field reinforcement learning, HMF [31] decomposes the entire population into multiple clusters and coordinates learning between two levels to optimize coordination in large-scale MAS. However, a common drawback of these methods is the lack of prior knowledge, requiring extensive trial-and-error, thereby leading to problems of low sample efficiency.

2.2 Symmetries in MARL

Figure 1 illustrates the concept of symmetry, showing that as the state rotates, the agents' directions of movement correspondingly rotate as well. Incorporating symmetry is an effective way to enhance sample efficiency in MARL. Recent research on incorporating symmetry includes three methods: data augmentation, loss function design, and network architecture design. The first two methods are referred to as soft constraint methods. The network architecture design is termed as hard constraint method. As for soft constraint methods, techniques like ESP incorporates symmetry into MARL algorithms by generating additional data and symmetry consistent loss through rotational symmetry [34]. PSE is an extension of ESP under imperfect symmetry conditions [35]. AdaptAUG selectively identifies beneficial data augmentations to improve sample efficiency and stability [36]. Regarding hard constraint methods, MF-PPO and HPN have designed network structures that ensure permutation invariance [11, 15]. Though numerous network structures with equivariant properties have been proposed [5, 6, 10, 23, 25, 26, 30], only Multi-Agent MDP Homomorphic Networks [28] have achieved the embedding of rotational symmetry in MARL. Despite these advancements, the utilization of hierarchical symmetry has not yet been explored in MARL. To our knowledge, this paper is the first to use hierarchical equivariant graph neural network to model MAS, allowing for the discovery of the system's hierarchical structure while incorporating strict symmetry.

3 Preliminaries

3.1 Notations

We model the multi-agent system as a graph $\mathcal{G}$, which includes nodes $\mathcal{V}$ representing N agents and edges $\mathcal{E}$ representing the connections between agents. The observations consist of equivariant features $\boldsymbol{\mathcal{X}}^{(0)}$ and scalar features $\boldsymbol{\mathcal{H}}^{(0)}$. Equivariant features refer to a characteristic that maintains equivariance under geometric transformations, denoted as $\boldsymbol{\mathcal{X}}^{(0)} = [\mathcal{X}_1^{(0)}, ..., \mathcal{X}_N^{(0)}]$ where $\boldsymbol{\mathcal{X}}^{(0)} \in \mathbb{R}^{N\times n\times M}$. This represents the combination of the equivariant features of N agents, with each agent characterized by M n-dimensional features, such as velocity $v_i \in \mathbb{R}^n$ and position $x_i \in \mathbb{R}^n$ leading to $\mathcal{X}_i^{(0)} = [v_i, x_i] \in \mathbb{R}^{n\times 2}$. Scalar features, on the other hand, refer to a characteristic that remains invariant under geometric transformations, denoted as $\boldsymbol{\mathcal{H}}^{(0)} = [\mathcal{H}_1^{(0)}, ..., \mathcal{H}_N^{(0)}]$. Here, $\boldsymbol{\mathcal{H}}^{(0)} \in \mathbb{R}^{N\times D}$ represents the combination of the scalar features of N agents, each containing D dimensions, such as distances and other scalar measurements. The edges $\mathcal{E}$ are related to the adjacency matrix A which is constructed based on geometric distances. In subsequent sections, we will abbreviate the complete information of a system, namely $\{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{H}}, A\}$, using the notation $\mathcal{G}$ when necessary like $\mathcal{G}^{(0)} = \{\boldsymbol{\mathcal{X}}^{(0)}, \boldsymbol{\mathcal{H}}^{(0)}, A\}$.

3.2 Dec-POMDP

A fully cooperative multi-agent task can be described as a decentralized partially observed Markov decision processes (Dec-POMDP) [17], which is defined as a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{O}, R, T, U, \gamma\rangle$. $\mathcal{N} = \{1, \ldots, N\}$ is the set of agents. $\mathcal{S}$ represents the set of environment state, and $s^t \in \mathcal{S}$ is the state at step t. $\mathcal{A} = \times_{i\in\mathcal{N}}\mathcal{A}_i$ denotes the joint action set where $\mathcal{A}_i$ is the set of actions available to agent i. Similarly, $\mathcal{O} = \times_{i\in\mathcal{N}}\mathcal{O}_i$ is the set of joint observations where $\mathcal{O}_i$ is the set of observations available to agent i. $T\left(s^{t+1} \mid s^t, \boldsymbol{a}^t\right)$ denotes the transition probability from s^t to s^{t+1} given the joint action $\boldsymbol{a}^t = \left(a_1^t, \ldots, a_N^t\right) \in \mathcal{A}$ by all n agents at step t. The subsequent observation o^{t+1} is given by $U(o^{t+1} \mid \boldsymbol{a}^t, s^{t+1})$. R is the immediate reward function. The discount factor is denoted by γ. Each agent employs the policy $\pi_{i,\theta}\left(a_i^t \mid o_i^t\right)$, parameterized by θ, to generate action a_i^t based on local observation o_i^t at step t. The goal is to optimize the expected discounted cumulative reward $J(\theta) = E_{a^t,s^t}\left[\sum_t \gamma^t R\left(s^t, a^t\right)\right]$.

3.3 Symmetric Dec-POMDP

By introducing symmetry constraints into Dec-POMDP, we obtain a subclass known as Symmetric Dec-POMDP (S-Dec-POMDP) [34]. Through S-Dec-POMDP, we can leverage the embeddings of inherent system symmetries to optimize policies in MARL. In this paper, we aim to propose a policy network $\mathcal{F}_{\mathrm{HEPN}}(\cdot)$ that ensures $O(n)$-equivariance. It can be formalized as follows:

$$\mathcal{F}_{\mathrm{HEPN}}\left(\boldsymbol{\mathcal{H}}^{(0)}, \mathcal{L}_g\boldsymbol{\mathcal{X}}^{(0)}, A\right) = \mathcal{L}_g\boldsymbol{a},$$

where $\mathcal{L}_g$ is the transformation for equivariant features and actions associated with group $g \in O(n)$. $\boldsymbol{\mathcal{H}}^{(0)}$ and $\boldsymbol{\mathcal{X}}^{(0)}$ are the initial scalar and equivariant feature of agents defined in section 3.1.

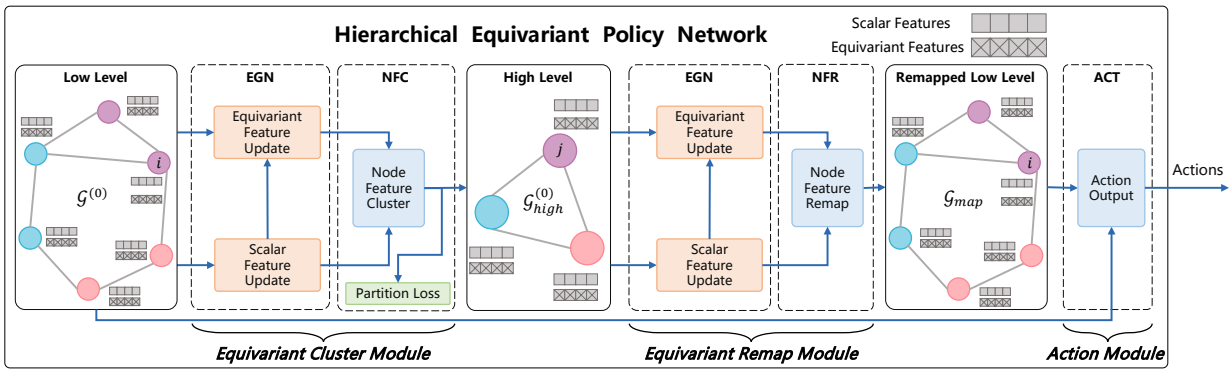


Figure 2. The overall framework of the proposed HEPN consisting of three main modules: 1) Equivariant Cluster Module, used to extract the hierarchical structure in multi-agent systems, clustering agents with similarities into a group to serve as agents in the high-level system; 2) Equivariant Remap Module, used to remap information from the high-level system back to the low-level system; 3) Action Module, used to generate the final action output.

3.4 Equivariant Graph Network

Given the input graph $\mathcal{G}^{(l)} = \{\boldsymbol{\mathcal{X}}^{(l)}, \boldsymbol{\mathcal{H}}^{(l)}, A\}$, the l-th Equivariant Graph Network (EGN) layer is used for message passing and aggregation within the graph to generate new features $\mathcal{G}^{(l+1)} = \{\boldsymbol{\mathcal{X}}^{(l+1)}, \boldsymbol{\mathcal{H}}^{(l+1)}, A\}$ while maintaining equivariance [23]. The whole procedure can be formalized as follows:

$$m_{ij} = \phi_e\left(\left\|\mathcal{X}_i^{(l)} - \mathcal{X}_j^{(l)}\right\|^2, \mathcal{H}_i^{(l)}, \mathcal{H}_j^{(l)}\right),$$
$$\mathcal{H}_i^{(l+1)} = \phi_h\left(\mathcal{H}_i^{(l)}, \sum\nolimits_{j\in\mathcal{N}(i)} m_{ij}\right),$$
$$\mathcal{X}_i^{(l+1)} = \phi_x(\mathcal{H}_i^{(l)})\mathcal{X}_i^{(l)} + \frac{1}{|\mathcal{N}(i)|}\sum\nolimits_{j\in\mathcal{N}(i)} \mathcal{X}_{ij}^{(l)}\phi_m(m_{ij}),$$

where $\|\cdot\|^2$ represents the column-wise l_2 distance, $\mathcal{N}(i)$ denotes the set of neighboring agents of agent i, and $|\mathcal{N}(i)|$ denotes the number of neighbors. $\phi_e(\cdot)$, $\phi_h(\cdot)$, $\phi_x(\cdot)$ and $\phi_m(\cdot)$ are all Multi-Layer Perceptrons (MLP). In subsequent sections, we denote EGN as:

$$\boldsymbol{\mathcal{H}}^{(L)}, \boldsymbol{\mathcal{X}}^{(L)} = \mathcal{F}_{\text{EGN}}(\boldsymbol{\mathcal{H}}^{(0)}, \boldsymbol{\mathcal{X}}^{(0)}, A),$$

where L represents that the network has L layers. For arbitrary orthogonal matrix $\mathcal{L}_g \in O(n)$, $\mathcal{F}_{\text{EGN}}$ satisfies:

$$\boldsymbol{\mathcal{H}}^{(L)}, \mathcal{L}_g\boldsymbol{\mathcal{X}}^{(L)} = \mathcal{F}_{\text{EGN}}(\boldsymbol{\mathcal{H}}^{(0)}, \mathcal{L}_g\boldsymbol{\mathcal{X}}^{(0)}, A).$$

4 Method

In this section, we will introduce the HEPN. The complete network architecture is shown in Figure 2. The overall procedure of the proposed HEPN is formulated as follows:

$$\mathcal{G}_{\text{high}}^{(0)} = \mathcal{F}_{\text{ECM}}\left(\mathcal{G}^{(0)}\right), \tag{1}$$
$$\mathcal{G}_{\text{map}} = \mathcal{F}_{\text{ERM}}\left(\mathcal{G}_{\text{high}}^{(0)}\right), \tag{2}$$
$$\boldsymbol{a} = \mathcal{F}_{\text{ACT}}\left(\mathcal{G}^{(0)}, \mathcal{G}_{\text{map}}\right). \tag{3}$$

Here, Equation (1) extracts the hierarchical structure of MAS by employing the Equivariant Cluster Module $\mathcal{F}_{\text{ECM}}(\cdot)$ to cluster similar agents in the low-level system $\mathcal{G}^{(0)}$, treating these clusters as agents in the high-level system $\mathcal{G}_{\text{high}}^{(0)}$. Equation (2) utilizes the Equivariant Remap Module $\mathcal{F}_{\text{ERM}}(\cdot)$ to recover the features from $\mathcal{G}_{\text{high}}^{(0)}$ back to the low-level system $\mathcal{G}_{\text{map}}$. Equation (3) employs the Action Module $\mathcal{F}_{\text{ACT}}(\cdot)$ to generate the final action output. The implementation details of HEPN can be found in supplementary materials.

4.1 Equivariant Cluster Module

The primary purpose of this module $\mathcal{F}_{\text{ECM}}(\cdot)$ is to cluster agents with similarities in the low-level system into a group, which serves as an agent in the high-level system. This module consists of EGN and Node Feature Cluster. Given the initial low-level system $\mathcal{G}^{(0)} = \{\boldsymbol{\mathcal{X}}^{(0)}, \boldsymbol{\mathcal{H}}^{(0)}, A\}$, the entire process can be described as follows:

$$\boldsymbol{\mathcal{H}}^{(L)}, \boldsymbol{\mathcal{X}}^{(L)} = \mathcal{F}_{\text{EGN}}\left(\boldsymbol{\mathcal{H}}^{(0)}, \boldsymbol{\mathcal{X}}^{(0)}, A\right), \tag{4}$$
$$\mathcal{G}_{\text{high}}^{(0)} = \mathcal{F}_{\text{NFC}}\left(\mathcal{G}^{(L)}\right). \tag{5}$$

Equation (4) is EGN described in section 3.4 which repeats L times. Due to the presence of the message passing mechanism in EGN, agents in the low-level system have acquired more global information compared to their initial state. Equation (5) operates on the updated system $\mathcal{G}^{(L)} = \{\boldsymbol{\mathcal{X}}^{(L)}, \boldsymbol{\mathcal{H}}^{(L)}, A\}$ and utilizes the Node Feature Cluster $\mathcal{F}_{\text{NFC}}(\cdot)$ to coarsen the low-level system into an abstract high-level system $\mathcal{G}_{\text{high}}^{(0)} = \{\boldsymbol{\mathcal{X}}_{\text{high}}^{(0)}, \boldsymbol{\mathcal{H}}_{\text{high}}^{(0)}, A_{\text{high}}\}$ with K agents, where $K < N$. Furthermore, we use Partition Loss to optimize the clustering results.

Node Feature Cluster. The $\mathcal{F}_{\text{NFC}}(\cdot)$ is used for clustering agents and generating high-level features to construct the high-level system, which can be represented by the following four equations:

$$p_i = Softmax\left(\mathcal{H}_i^{(L)}\right), \tag{6}$$
$$\mathcal{X}_{k,\text{high}}^{(0)} = \sum_{i=1}^{N} p_{ik}\mathcal{X}_i^{(L)}, \tag{7}$$
$$\mathcal{H}_{k,\text{high}}^{(0)} = \sum_{i=1}^{N} p_{ik}\mathcal{H}_i^{(L)}, \tag{8}$$
$$A_{\text{high}} = P^T AP, \tag{9}$$

where p_i denotes the probability of agent i belonging to each cluster, and $P = [p_{ik}]_{N\times K}$ represents the probability matrix. $\mathcal{H}_i^{(L)}$ and $\mathcal{X}_i^{(L)}$ are scalar and equivariant features of agent i, respectively. First, we calculate the probability using the updated features in the low-level system, which is achieved through a softmax function shown in Equation (6). Next, each group is considered as a node in the high-level system, whose features are the weighted sum of the features in the low-level system as shown in Equations (7) and (8). The weights are provided by the P. Finally, as illustrated in Equation (9), we use P to compute the high-level adjacency matrix A_{high}.

Partition Loss. To enhance the clustering effectiveness within the Equivatiant Cluster Module, inspired by structural entropy [14], we propose a partition loss function. Following Equation (9), we define the edge weighs w_{ij} as:

$$w_{ij} = \begin{cases} cos\_sim\left(\mathcal{H}_i^{(L)}, \mathcal{H}_j^{(L)}\right) + 1 & \text{if } e_{ij} \in \mathcal{E}, \\ 0 & others \end{cases}, \tag{10}$$

where $cos\_sim(\cdot)$ representing the cosine similarity. Suppose that $\{\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_C\}$ is a partition of nodes $\mathcal{V}$. We define the partition loss of $\mathcal{G}$ by the partition as:

$$loss = -\sum_{c=1}^{C} \frac{vol(\mathcal{V}_c)}{m_c} - \sum_{c=1}^{C} \frac{g_c}{vol(\mathcal{G})} \log_2 \frac{vol(\mathcal{V}_c) + g_c}{vol(\mathcal{G})}, \tag{11}$$

where $vol(\mathcal{G}) = \sum_{i,j \in \mathcal{V}, i \neq j} w_{ij}$ is the volume of $\mathcal{G}$, $vol(\mathcal{V}_c) = \sum_{i,j \in \mathcal{V}_l, i \neq j} w_{ij}$ is the volume of $\mathcal{V}_c$, m_c is the number of edges whose endpoints are both within $\mathcal{V}_c$, and g_c is the sum of edge weights with exactly one endpoint in $\mathcal{V}_c$. Minimizing this loss ensures optimal graph partitioning. The first term encourages the maximization of the total edge weight within each cluster. This optimization promotes greater cohesion and homogeneity within individual clusters, enhancing the similarity among vertices within the same cluster. The second term focuses on the inter-cluster edges and aims to minimize the ambiguity at cluster boundaries. Reducing this term results in fewer and weaker connections between distinct clusters, ultimately emphasizing the demarcation between them. The overall loss function of HEPN is the sum of the MARL algorithm's loss and partition loss.

4.2 Equivariant Remap Module

In this section, we delve into the Equivariant Remap Module, which aims at remapping the information from high-level system back to the low-level for subsequent action generation. It comprises two parts: EGN and Node Feature Remap. Given the initial state of the high-level system $\mathcal{G}_{\text{high}}^{(0)} = \{\boldsymbol{\mathcal{X}}_{\text{high}}^{(0)}, \boldsymbol{\mathcal{H}}_{\text{high}}^{(0)}, A_{\text{high}}\}$, the whole process is formulated as:

$$\boldsymbol{\mathcal{H}}_{\text{high}}^{(L)}, \boldsymbol{\mathcal{X}}_{\text{high}}^{(L)} = \mathcal{F}_{\text{EGN}}\left(\boldsymbol{\mathcal{H}}_{\text{high}}^{(0)}, \boldsymbol{\mathcal{X}}_{\text{high}}^{(0)}, A_{\text{high}}\right), \tag{12}$$

$$\mathcal{G}_{map} = \mathcal{F}_{\text{NFR}}\left(\mathcal{G}_{\text{high}}^{(L)}\right), \tag{13}$$

where $\mathcal{F}_{\text{EGN}}(\cdot)$ is EGN described in section 3.4. Similar to the Equivariant Cluster Module described in section 4.1, the state of the high-level system first undergoes an update through the EGN. Since the agents in the high-level system represent a category of agents in the low-level system, their updates through a message passing mechanism can acquire more global information compared to performing the same operations in the low-level system, achieving more efficient feature updates. Then, Node Feature Remap $\mathcal{F}_{\text{NFR}}(\cdot)$ will project the features from the high-level system $\mathcal{G}_{\text{high}}^{(L)} = \{\boldsymbol{\mathcal{X}}_{\text{high}}^{(L)}, \boldsymbol{\mathcal{H}}_{\text{high}}^{(L)}, A_{\text{high}}\}$ back to the low-level system $\mathcal{G}_{map} = \{\boldsymbol{\mathcal{X}}_{\text{map}}, \boldsymbol{\mathcal{H}}_{\text{map}}, A\}$.

Node Feature Remap. The Node Feature Remap aims to remap the high-level features to the low-level. The process is formulated as:

$$\mathcal{X}_{i,\text{map}} = \sum\nolimits_{k=1}^{K} p_{ik} \mathcal{X}_{k,\text{high}}^{(L)}, \tag{14}$$

$$\mathcal{H}_{i,\text{map}} = \sum\nolimits_{k=1}^{K} p_{ik} \mathcal{H}_{k,\text{high}}^{(L)}. \tag{15}$$

The high-level agents remap the features back to the agents in the low-level system through a weighted summation using weights provided by the columns of the probability matrix P, resulting in the mapped system $\mathcal{G}_{\text{map}} = \{\boldsymbol{\mathcal{X}}_{\text{map}}, \boldsymbol{\mathcal{H}}_{\text{map}}, A\}$.

4.3 Action Module

In this section, we introduce the last module of HEPN, the Action Module $\mathcal{F}_{\text{act}}(\cdot)$, which is used to generate actions for the agents. It combines features from both $\mathcal{G}^{(0)}$ and $\mathcal{G}_{\text{map}}$ to determine the actions for each agent, as given by:

$$\mathcal{H}_{i,\text{out}} = \phi_o\left(\left\|\hat{\mathcal{X}}_i\right\|^2, \left\|\hat{\mathcal{X}}_{i,\text{map}}\right\|^2, \mathcal{H}_i, \mathcal{H}_{i,\text{map}}\right), \tag{16}$$

$$a_i = \left(\mathcal{X}_{i,\text{map}} - \mathcal{X}_i^{(0)}\right) \mathcal{H}_{i,out}, \tag{17}$$

where $\hat{\mathcal{X}}_i = \mathcal{X}_i^{(0)} - \bar{\mathcal{X}}_i$, and $\bar{\mathcal{X}}_i$ is the average of the features of agent i's neighbors. The calculation of $\hat{\mathcal{X}}_{i,\text{map}}$ can be analogously derived from $\hat{\mathcal{X}}_i$. $\phi_o(\cdot)$ is implemented as MLP. Equation (16) defines $\mathcal{H}_{i,\text{out}}$, the scaling factor for computing actions, which integrates the equivariant features and scalar features from both $\mathcal{G}^{(0)}$ and $\mathcal{G}_{\text{map}}$. Equation (17) details how the action a_i is computed. The action of agent i is derived from the difference in equivariant features between $\mathcal{G}^{(0)}$ and $\mathcal{G}_{\text{map}}$, and is scaled by $\mathcal{H}_{i,\text{out}}$. The difference in equivariant features determines the direction of the action while the scaling factor determines the magnitude of the action. This combination ensures that the generation of actions maintains equivariance while incorporating sufficient information.

4.4 Theoretical Analysis

In this section, we analyze the equivariance properties of our method, HEPN, as described in the following theorem and corollary.

Proposition 1. *For arbitrary orthogonal matrix $\mathcal{L}_g \in O(n)$, the modules of our network satisfy:*

1. *The Node Feature Cluster $\mathcal{F}_{\text{NFC}}(\cdot)$ is equivariant:*

$$\mathcal{L}_g \mathcal{G}_{\text{high}}^{(0)} = \mathcal{F}_{\text{NFC}}\left(\mathcal{L}_g \mathcal{G}^{(L)}\right).$$

2. *The Node Feature Remap $\mathcal{F}_{\text{NFR}}(\cdot)$ is equivariant:*

$$\mathcal{L}_g \mathcal{G}_{map} = \mathcal{F}_{\text{NFR}}\left(\mathcal{L}_g \mathcal{G}_{\text{high}}^{(L)}\right).$$

3. *The Action Module $\mathcal{F}_{\text{ACT}}(\cdot)$ is equivariant:*

$$\mathcal{L}_g \boldsymbol{a} = \mathcal{F}_{\text{ACT}}\left(\mathcal{L}_g \mathcal{G}^{(0)}, \mathcal{L}_g \mathcal{G}_{\text{map}}\right).$$

The detailed proof is provided in the supplementary materials. Based on Proposition 1 and section 3.4, we can conclude that $\mathcal{F}_{\text{ECM}}(\cdot)$ and $\mathcal{F}_{\text{ERM}}(\cdot)$ is equivariant:

Corollary 1. *For arbitrary orthogonal matrix $\mathcal{L}_g \in O(n)$, the modules of our network satisfy:*

1. *The Equivariant Cluster Module $\mathcal{F}_{\text{ECM}}(\cdot)$ is equivariant:*

$$\mathcal{L}_g \mathcal{G}_{\text{high}}^{(0)} = \mathcal{F}_{\text{ECM}}\left(\mathcal{L}_g \mathcal{G}^{(0)}\right).$$

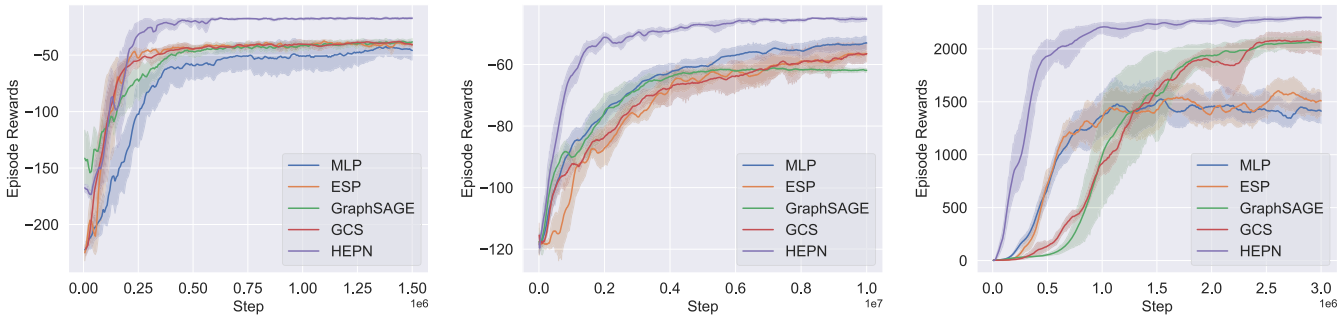
2. *The Equivariant Remap Module $\mathcal{F}_{\text{ERM}}(\cdot)$ is equivariant:*

$$\mathcal{L}_g \mathcal{G}_{\text{map}} = \mathcal{F}_{\text{ERM}}\left(\mathcal{L}_g \mathcal{G}_{\text{high}}^{(0)}\right).$$

Based on Proposition 1 and Corollary 1, we can conclude that our entire network $\mathcal{F}_{\text{HEPN}}(\cdot)$ is $O(n)$-equivariant:

Corollary 2. *For arbitrary and orthogonal matrix $\mathcal{L}_g \in O(n)$, our whole network HEPN $\mathcal{F}_{\text{HEPN}}(\cdot)$ satisfies:*

$$\mathcal{L}_g \boldsymbol{a} = \mathcal{F}_{\text{HEPN}}\left(\boldsymbol{\mathcal{H}}^{(0)}, \mathcal{L}_g \boldsymbol{\mathcal{X}}^{(0)}, A\right).$$



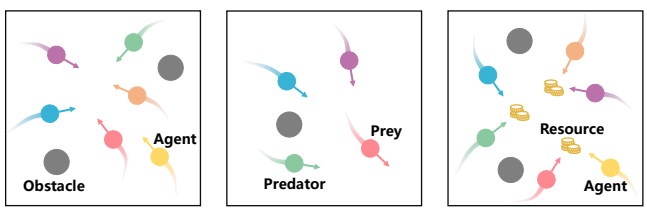
(a) Rendezvous (b) Pursuit (c) Resource Collection

Figure 3. The learning curves of HEPN, MLP, GraphSAGE, ESP and GCS across three tasks are presented. Each experiment is conducted five times with different random seeds to ensure the reliability of the results.

5 Experiments

5.1 Environmental Setting

Baselines. Our HEPN focuses on incorporating symmetry into MARL algorithms. We use Exploiting Symmetry Prior (ESP) [34] as a baseline, as it is the most recent and effective method in this field. Additionally, given that MLP is commonly used as policy networks in MARL, it also serve as one of our baselines, with MAPPO [32] selected for our paper. MARL with Graph Neural Networks, like GraphSAGE [9, 24] and graph-based coordination strategy (GCS) [21], due to their relevance in modeling interactions among agents, are also selected as baseline algorithms.



(a) Rendezvous (b) Pursuit (c) Resource Collection

Figure 4. The simulated tasks considered in the experiments.

Tasks. We evaluate the superiority of our method through experiments conducted on three multi-agent continuous cooperative tasks: 1) **Rendezvous**, where agents autonomously gather without a predefined target point; 2) **Pursuit**, where multiple predators attempt to chase a faster-moving prey; 3) **Resource Collection**, which requires agents to mine all the resources from multiple resource pools while avoiding collisions with other agents and obstacles [1]. Figure 4 illustrates these tasks. The key hyperparameters of the algorithm are consistent with those in [38]. More details about the experimental setting can be found in the supplementary materials.

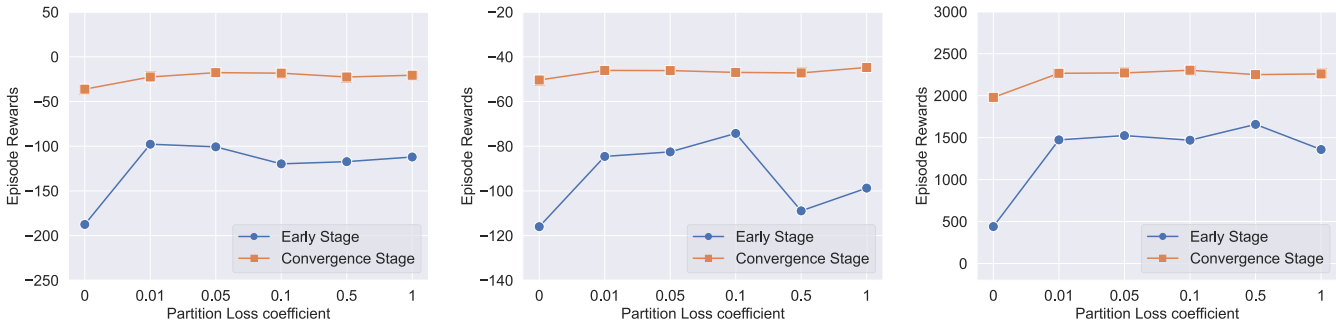
5.2 Main Results

In this section, we present and analyze the main results of the experiments conducted under the environmental settings described in Section 5.1. The number of agents in every task is set at 10. The experimental results for each algorithm represent the average of five different random seeds, as shown in Figure 3. The results indicate that when our HEPN is applied to the MARL algorithm, it achieves faster convergence speeds and higher rewards.

Rendezvous. In Rendezvous task, as shown in Figure 3(a), the HEPN exhibits a marked superiority in achieving higher convergence rewards. It also demonstrates a modest improvement in terms of convergence speed compared to other methods. Notably, ESP emerged as the most effective algorithm within the baselines. This highlights the benefits of leveraging the intrinsic symmetry present within the system. Nonetheless, the gap in reward outcomes between ESP and HEPN suggests that the effectiveness of methods employing soft constraints generally falls short of those utilizing strategic network architecture designs. Moreover, both GCS and GraphSAGE outperform the standard MLP, affirming the advantage of incorporating graph structures into MARL algorithms.

Pursuit. In Pursuit task, the prey adopts a Voronoi strategy as detailed in [40]. Our target is to train agents to chase the prey. As we can see in Figure 3(b), HEPN distinctly surpasses all other baseline algorithms in terms of both convergence speed and reward in this task. ESP underperforms relative to the standard MLP, potentially due to its inability to adeptly handle the environmental uncertainties introduced by the prey's movements. This suggests that ESP struggles with more complex tasks. Both GraphSAGE and GCS show faster convergence speeds than ESP and are comparable to the standard MLP, indicating that the use of graph structures can mitigate the uncertainty in system. However, the inferior convergence rewards suggest that despite graph structure capturing the interactions between agents, they fail to sufficiently generalize the entire environmental state in the absence of inherent symmetry and hierarchical structure which may diminish performance.

Resource Collection. In Resource Collection, we set three distinct resource pools, which allows for the possibility that multiple agents might converge on the same pool. The insights from Figure 3(c) demonstrate that the HEPN significantly outperforms the baseline methods. This advantage is attributed to HEPN's ability to efficiently direct agents toward resource pools while simultaneously avoiding collisions. The ESP method shows only a slight improvement over the standard MLP, and its reward at convergence is considerably lower than other algorithms. This observation suggests that data augmentation methods might not be ideally suited for complex tasks. On the other hand, both GraphSAGE and GCS display slower convergence speeds. While their final reward levels do surpass those of both the standard MLP and ESP, they still fall short of reaching the high performance levels exhibited by HEPN. These results further confirms the advantages of integrating intrinsic symmetry and hierarchical structures within algorithms to optimize performance.



(a) Rendezvous (b) Pursuit (c) Resource Collection

Figure 5. Sensitivity of the partition loss coefficient across three tasks. Each point represents the average reward value at the early stage or the convergence stage of training.

5.3 *Ablation Study*

In this section, we demonstrate the results of ablation studies on the hierarchical structure and partition loss of HEPN across three tasks: Rendezvous, Pursuit, and Resource Collection. These experiments are designed to assess the effectiveness of different components within the method.

Table 1. Ablation study results for the hierarchical structure across three tasks. Each value represents the average convergence reward of the model.

Model	Task		
	Rendezvous	Pursuit	Collection
HEPN	**-15.2**	**-43.1**	**2319.2**
EPN	-16.3	-51.6	2168.4
HPN	-29.7	-56.1	2137.9
GPN	-33.0	-58.5	2103.0

Effectiveness of Hierarchy and Equivariant. We compare four different variants: 1) HEPN, which includes all components described in this paper; 2) EPN, which removes the hierarchical structure; 3) HPN, which omits the equivariance; 4) GPN, where both the hierarchical structure and equivariance are removed, essentially being a basic GNN, We use the results of GraphSAGE here. The performances of these variants, as shown in Table 1, indicate that the removal of hierarchical structures results in a slight decrease in performance, which intensifies with increasing task complexity. The removal of equivariance leads to a significant performance decline, and the basic GNN model performs the worst. Thus, it can be concluded that the incorporating of hierarchical structures greatly assists in complex tasks within multi-agent systems, while the consideration of equivariance significantly enhances algorithmic performance.

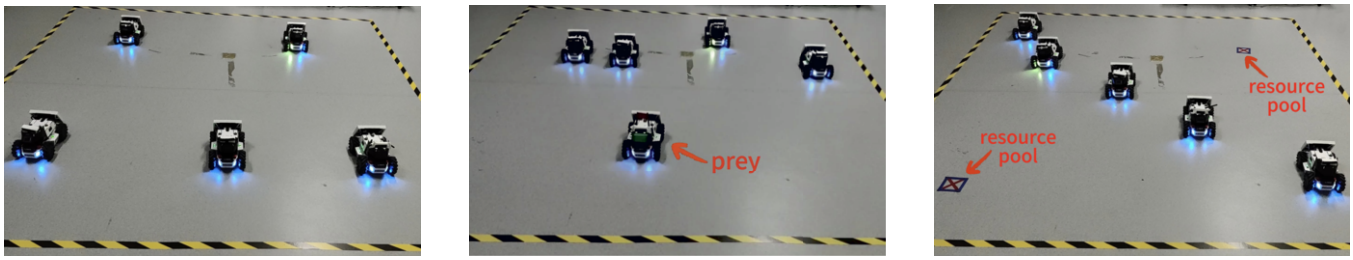
Effectiveness of Partition Loss. We selecte a range of loss function coefficients [0, 0.01, 0.05, 0.1, 0.5, 1] and conducte experiments in three tasks. The experimental results are illustrated in Figure 5, where we have selected the reward values from both the early and the convergence stage of training. Specifically, we have chosen 150k and 390k, 300k and 2100k, and 300k and 2100k steps as representative of the early and convergence stage of the three tasks, respectively. From Figure 5, it can be observed that in all three tasks, the use of partition loss effectively enhances the reward values of the algorithm, both in early stages and convergence stages of training. Notably, the significant improvement in reward values during the early stages suggests that partition loss can effectively accelerate the convergence speed. However, it is important to note that the coefficient of partition loss is not necessarily better when larger. For example, when the coefficient exceeds 0.05, a decline in the early reward is observed in the Rendezvous task, with similar phenomena occurring in the Pursuit and Resource Collection as well. This indicates the need for selecting appropriate partition loss coefficients for different tasks. Nevertheless, it is noteworthy that the coefficient of partition loss does not significantly affect the final convergence rewards. In summary, the experimental results demonstrate that partition loss, by optimizing the results of clustering, effectively enhances both the convergence speed and the final convergence rewards. We also find that the convergence speed of the algorithm shows some sensitivity to the partition loss coefficient, but its impact on the convergence rewards is minimal.

Table 2. This table displays the impact of varying numbers of agents on different tasks, as expressed by the mean convergence rewards of the models.

Task	Num	Model				
		HEPN	MLP	ESP	SAGE	GCS
Rendezvous	5	**-21.8**	-29.9	-29.3	-26.1	-35.8
	15	**-24.1**	-33.9	-34.7	-33.1	-34.7
	20	**-24.6**	-38.7	-35.0	-33.9	-40.0
	30	**-25.4**	-34.9	-35.8	-35.3	-39.2
Pursuit	5	**-67.1**	-73.3	-79.6	-73.1	-75.3
	15	**-36.6**	-51.6	-48.3	-47.4	-59.1
	20	**-30.3**	-51.1	-47.0	-48.5	-69.2
	30	**-25.3**	-55.9	-53.4	-47.7	-67.5
Collection	5	**2295.4**	2183.9	1989.5	2105.4	2283.0
	15	**2209.9**	1273.9	1552.8	1330.7	1106.7
	20	**2295.8**	603.3	1151.3	795.8	337.9
	30	**2297.7**	1054.4	767.3	966.5	146.9

5.4 *The Impact of Different Numbers of Agents*

In this section, we explore the performance of different models with varying numbers of agents, 5, 10, 15, 20 and 30, as shown in Table 2. The numbers represent the average convergence rewards for each model. The results also demonstrate the effectiveness of HEPN: 1) In Rendezvous, HEPN consistently performs the best across all agent numbers and shows higher stability, whereas other models exhibited performance declines with increasing numbers; 2) In Pursuit, HEPN's performance became more pronounced as the number of agents increased, highlighting its effectiveness in large-scale complex tasks. 3) In Resource Collection, all algorithms except HEPN showed significant performance decline, further validating the efficacy of our method in handling complex, large-scale tasks.



(a) Rendezvous (b) Pursuit (c) Resource Collection

Figure 6. Demonstration of the physics environments across three tasks.

Table 3. Behavioral analysis results of different algorithms on three tasks: The inter-agent distances at different stages for Rendezvous; The nearest, farthest and mean agent-to-prey distances for Pursuit; The agent-to-pool distances at different stages for Resource Collection.

	Rendezvous			Pursuit			Resource Collection		
Model	50-step	100-step	200-step	Nearest	Farthest	Mean	50-step	100-step	200-step
HEPN	**1571.3**	**77.7**	**1.1**	**12.5**	**47.8**	**27.5**	**11.4**	**3.0**	**1.6**
MLP	2957.5	2119.0	722.6	16.2	51.8	32.3	21.5	10.2	5.6
ESP	2835.3	2276.7	592.6	16.7	51.5	32.1	12.3	10.3	7.4
GraphSAGE	2987.0	1937.1	414.5	16.8	50.6	30.8	25.8	12.8	1.8
GCS	2634.5	1653.5	227.2	15.5	52.2	31.2	23.9	12.36	5.6

5.5 Behavior Analysis

In this section, we analyze the behavior of models trained and converged using different methods arcoss three tasks. For each task evaluation, we set the episode length to 300, by which point all models can complete the task. We assess the performance using distance as a metric, with smaller distances indicating better performance:

Rendezvous. We use the sum of distances between all pairs of agents as a metric, termed inter-agent distance. We record the mean inter-agent distances at the 50th, 100th, and 200th time steps across all three tasks. As shown in Table 3, our HEPN model performs better at all three stages, having completed the Rendezvous task by the 200th time step, while the other algorithms are far from completion.

Pursuit. We use the distance between agents and the prey, denoted as agent-to-prey distance, as the evaluation metric. Given that the prey always tries to escape, we assess performance using the nearest, farthest, and mean distances to the prey throughout the whole episode. The data in Table 3 shows that agents trained with HEPN are more effective at encircling the prey.

Resource Collection. We use the mean distance between the agents and the resource pools as a metric, referred to as agent-to-pool distance. We assess algorithm performance using the agent-to-pool distances at the 50th, 100th, and 200th time steps. The results indicate that HEPN-controlled agents can reach the resource pool faster.

6 Demonstration on Robots

In this section, we evaluate the performance of algorithms through a Sim2Real approach. We have deployed our trained models in a real-world setting to assess their effectiveness across all three tasks. Specifically, Limo robots are employed as the task agents, which are controlled via the Robot Operating System (ROS) [13] and supplemented with a NOKOV motion capture system to get real-time environment states. Figure 6 provides a schematic of the actual application scenario. The configurations for the tasks are as follows: 5 agents for Rendezvous, 4 agents chasing 1 prey for Pursuit, and Resource Collection involves 5 agents and 2 resource pools. To accurately assess the performance of algorithms in a real-world environment, we compare HEPN along with the best-performing baselines for each task. The reward function is composed of the cumulative distance of all agents, along with auxiliary training regularization terms which do not intuitively reflect the algorithm's performance in real-world experiments. Therefore, we have chosen the cumulative distance as the evaluation metric. The results shown in Table 4 demonstrate that our method can complete tasks faster, thereby proving the effectiveness of HEPN in real-world scenarios. Additional details and video are provided in Supplementary Materials.

Table 4. Performance results of the methods in real-world tasks.

Task	Model	
	HEPN	Best Baseline
Rendezvous	**659.4**	759.9
Pursuit	**1337.7**	1531.7
Collection	**823.9**	1098.5

7 Conclusion

This paper aims to address the issue of low sample efficiency in MARL. We introduce a novel method called Hierarchical Equivariant Policy Network (HEPN), which leverages the intrinsic hierarchical symmetry within MAS to significantly reduce the need for policy exploration in MARL, thereby enhancing learning efficiency. Additionally, we propose a partition loss to more effectively cluster agents, enhancing the overall performance of the algorithm. Experimental results demonstrate that our method significantly outperforms existing techniques in terms of convergence speed and rewards. The applicability and superiority of our approach are further validated in real-world physical scenarios. This work illustrates a potential pathway for enhancing algorithm efficiency by utilizing prior knowledge. In the future, we will explore the application of our method in other reinforcement learning domains [19].

Acknowledgements

This work is supported in part by the National Science and Technology Major Project (Grant No. 2018AAA0102300)

References

[1] X. Chen, X. Liu, Y. Ba, S. Zhang, B. Ding, and K. Li. Selective learning for sample-efficient training in multi-agent sparse reward tasks. In *ECAI 2023*, pages 413–420. IOS Press, 2023.

[2] T. Chu, J. Wang, L. Codecà, and Z. Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1086–1095, 2019.

[3] P. Feng, X. Yu, J. Liang, W. Wu, and Y. Tian. Mact: Multi-agent collision avoidance with continuous transition reinforcement learning via mixup. In *International Conference on Swarm Intelligence*, pages 74–85. Springer, 2023.

[4] P. Feng, J. Liang, S. Wang, X. Yu, R. Shi, and W. Wu. Hierarchical consensus-based multi-agent reinforcement learning for multi-robot cooperation tasks, 2024. URL https://arxiv.org/abs/2407.08164.

[5] M. Finzi, S. Stanton, P. Izmailov, and A. G. Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176. PMLR, 2020.

[6] F. Fuchs, D. Worrall, V. Fischer, and M. Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in neural information processing systems*, 33:1970–1981, 2020.

[7] S. Gronauer and K. Diepold. Multi-agent deep reinforcement learning: a survey. *Artif. Intell. Rev.*, 55(2):895–943, feb 2022. ISSN 0269-2821. doi: 10.1007/s10462-021-09996-w. URL https://doi.org/10.1007/s10462-021-09996-w.

[8] S. Gu, J. G. Kuba, Y. Chen, Y. Du, L. Yang, A. Knoll, and Y. Yang. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence*, 319:103905, 2023.

[9] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[10] J. Han, W. Huang, T. Xu, and Y. Rong. Equivariant graph hierarchy-based neural networks. *Advances in Neural Information Processing Systems*, 35:9176–9187, 2022.

[11] H. Jianye, X. Hao, H. Mao, W. Wang, Y. Yang, D. Li, Y. Zheng, and Z. Wang. Boosting multiagent reinforcement learning via permutation invariant and permutation equivariant networks. In *The Eleventh International Conference on Learning Representations*, 2022.

[12] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[13] A. Koubâa et al. *Robot Operating System (ROS).*, volume 1. Springer, 2017.

[14] A. Li and Y. Pan. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory*, 62(6):3290–3339, 2016.

[15] Y. Li, L. Wang, J. Yang, E. Wang, Z. Wang, T. Zhao, and H. Zha. Permutation invariant policy optimization for mean-field multi-agent reinforcement learning: A principled approach. *arXiv preprint arXiv:2105.08268*, 2021.

[16] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[17] F. A. Oliehoek, C. Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

[18] J. Orr and A. Dutta. Multi-agent deep reinforcement learning for multi-robot applications: a survey. *Sensors*, 23(7):3625, 2023.

[19] T. Peng, W. Wu, H. Yuan, Z. Bao, Z. Pengrui, X. Yu, X. Lin, Y. Liang, and Y. Pu. Graphrare: Reinforcement learning enhanced graph neural network with relative entropy. *arXiv preprint arXiv:2312.09708*, 2023.

[20] Y. Qiu, Y. Jin, L. Yu, J. Wang, Y. Wang, and X. Zhang. Improving sample efficiency of multi-agent reinforcement learning with non-expert policy for flocking control. *IEEE Internet of Things Journal*, 2023.

[21] J. Ruan, Y. Du, X. Xiong, D. Xing, X. Li, L. Meng, H. Zhang, J. Wang, and B. Xu. Gcs: Graph-based coordination strategy for multi-agent reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1128–1136, 2022.

[22] H. Ryu, H. Shin, and J. Park. Multi-agent actor-critic with hierarchical graph attention network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7236–7243, 2020.

[23] V. G. Satorras, E. Hoogeboom, and M. Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[24] Q. Sun, X. Wei, and X. Yang. Graphsage with deep reinforcement learning for financial portfolio optimization. *Expert Systems with Applications*, 238:122027, 2024.

[25] K. S. Tai, P. Bailis, and G. Valiant. Equivariant transformer networks. In *International Conference on Machine Learning*, pages 6086–6095. PMLR, 2019.

[26] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[27] E. Van der Pol, D. Worrall, H. van Hoof, F. Oliehoek, and M. Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020.

[28] E. van der Pol, H. van Hoof, F. A. Oliehoek, and M. Welling. Multi-agent mdp homomorphic networks. In *International Conference on Learning Representations*, 2021.

[29] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[30] Y. Wang and J. Chodera. Spatial attention kinetic networks with e (n)-equivariance. In *The Eleventh International Conference on Learning Representations*, 2022.

[31] C. Yu. Hierarchical mean-field deep reinforcement learning for large-scale multiagent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11744–11752, 2023.

[32] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[33] X. Yu, W. Wu, P. Feng, and Y. Tian. Swarm inverse reinforcement learning for biological systems. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 274–279. IEEE, 2021.

[34] X. Yu, R. Shi, P. Feng, Y. Tian, J. Luo, and W. Wu. Esp: Exploiting symmetry prior for multi-agent reinforcement learning. In *ECAI 2023*, pages 2946–2953. IOS Press, 2023.

[35] X. Yu, R. Shi, P. Feng, Y. Tian, S. Li, S. Liao, and W. Wu. Leveraging partial symmetry for multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17583–17590, 2024.

[36] X. Yu, Y. Tian, L. Wang, P. Feng, W. Wu, and R. Shi. Adaptaug: Adaptive data augmentation framework for multi-agent reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10814–10820, 2024. doi: 10.1109/ICRA57147.2024.10611035.

[37] X. Zeng, H. Peng, A. Li, C. Liu, L. He, and P. Yu. Hierarchical state abstraction based on structural information principles. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)*, 2023.

[38] Y. Zhong, J. G. Kuba, X. Feng, S. Hu, J. Ji, and Y. Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25:1–67, 2024.

[39] W. Zhou, D. Chen, J. Yan, Z. Li, H. Yin, and W. Ge. Multi-agent reinforcement learning for cooperative lane changing of connected and autonomous vehicles in mixed traffic. *Autonomous Intelligent Systems*, 2(1):5, 2022.

[40] Z. Zhou, W. Zhang, J. Ding, H. Huang, D. M. Stipanović, and C. J. Tomlin. Cooperative pursuit with voronoi partitions. *Automatica*, 72:64–72, 2016.