Supplementary Materials for 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

1 Additional Background

1.1 Groups and Transformations

In mathematical, groups are abstract entities that capture the essence of symmetry through a set of transformations. A group consists of a set of elements with an operation, while satisfying certain conditions including closure, associativity, identity, and invertibility. Transformations that preserve certain properties of a space form a group, which is a cornerstone in the study of symmetry. Within the realm of group theory, $O(n)$ group comprises all $n \times n$ orthogonal matrices, crucial for exploring rotations and reflections in n-dimensional space that preserve distances and angles. $O(n)$ is defined as:

$$O(n) = \{Z \in \mathbb{R}^{n\times n} : Z^T Z = I\},$$

where Z^T is the transpose of Z and I is the identity matrix. The group plays a foundational role in the study of symmetry embedding in MARL algorithms.

1.2 Equivariance and Invariance

In MARL tasks, symmetry typically manifests as equivariance and invariance to rotational transformations. Consider $\mathcal{L}_g : X \to X$ as the transformation applied to X, associated with an abstract group element $g \in G$. A function $f : X \to Y$ is considered equivariant to g if there is a corresponding transformation in its output space, denoted by $\mathcal{K}_g : Y \to Y$, which satisfies the following condition:

$$f(\mathcal{L}_g(\mathbf{x})) = \mathcal{K}_g(f(\mathbf{x})).$$

Furthermore, if the function f satisfies the condition:

$$f(\mathcal{L}_g(\mathbf{x})) = f(\mathbf{x}),$$

then f is said to possess invariance with respect to g.

2 Detailed Proof for Propositions

We denote $\mathcal{L}_g \in O(n)$ is arbitrary orthogonal matrix. In any system $\mathcal{G} = \{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{H}}, A\}$, both $\boldsymbol{\mathcal{H}}$ and A are composed of scalar values and are invariant to geometric transformations, denoted as $\mathcal{L}_g\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{H}}$ and $\mathcal{L}_g A = A$. Conversely, $\mathcal{X}$ consists of equivariant elements that maintain equivariance under geometric transformations. So we have $\mathcal{L}_g\mathcal{G} = \{\mathcal{L}_g\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{H}}, A\}$.

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

2.1 Proof of Proposition 1

In this section, we provide the proof of the Proposition 1 in our paper.

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

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

Proof. For agent i in low-level system and agent j in high-level system, we show the node feature cluster is equivariant step by step, flowing the definition in Section 4.1 of the main text:

$$\begin{aligned}
Softmax\left(\mathcal{L}_g\mathcal{H}_i^{(L)}\right) &= Softmax\left(\mathcal{H}_i^{(L)}\right) = p_i,\\
\sum_{i=1}^{N} p_{ik}\mathcal{L}_g\mathcal{X}_i^{(L)} &= \mathcal{L}_g\left(\sum_{i=1}^{N} p_{ik}\mathcal{X}_i^{(L)}\right) = \mathcal{L}_g\mathcal{X}^{(0)}_{k,\text{high}},\\
\sum_{i=1}^{N} p_{ik}\mathcal{L}_g\mathcal{H}_i^{(L)} &= \sum_{i=1}^{N} p_{ik}\mathcal{H}_i^{(L)} = \mathcal{H}^{(0)}_{k,\text{high}},\\
P^T\mathcal{L}_g AP &= P^T AP = A_{\text{high}}.
\end{aligned}$$

Then, we can get the flowing equation:

$$\begin{aligned}
\mathcal{F}_{\text{NFC}}\left(\mathcal{L}_g\mathcal{G}^{(L)}\right) &= \mathcal{F}_{\text{NFC}}\left(\mathcal{L}_g\boldsymbol{\mathcal{X}}^{(L)}, \boldsymbol{\mathcal{H}}^{(L)}, A\right)\\
&= \{\mathcal{L}_g\boldsymbol{\mathcal{X}}^{(0)}_{\text{high}}, \boldsymbol{\mathcal{H}}^{(0)}_{\text{high}}, A_{\text{high}}\}\\
&= \mathcal{L}_g\mathcal{G}^{(0)}_{\text{high}}.
\end{aligned}$$

□

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

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

Proof. For agent i in low-level system and agent k in high-level system, the node feature remap is equivariant. We show the proof flowing the definition in Section 4.2 of the main text:

$$\begin{aligned}
\sum_{k=1}^{K} p_{ik}\mathcal{L}_g\mathcal{X}^{(L)}_{k,\text{high}} &= \mathcal{L}_g\left(\sum_{k=1}^{K} p_{ik}\mathcal{X}^{(L)}_{k,\text{high}}\right) = \mathcal{L}_g\mathcal{X}_{i,\text{map}},\\
\sum_{k=1}^{K} p_{ik}\mathcal{L}_g\mathcal{H}^{(L)}_{k,\text{high}} &= \sum_{k=1}^{K} p_{ik}\mathcal{H}^{(L)}_{k,\text{high}} = \mathcal{H}_{i,\text{map}}.
\end{aligned}$$

Then we can obtain the following derivation process:

$$\begin{aligned}\mathcal{F}_{\text{NFR}}\left(\mathcal{L}_g\mathcal{G}_{\text{high}}^{(L)}\right) &= \mathcal{F}_{\text{NFR}}\left(\mathcal{L}_g\boldsymbol{\mathcal{X}}_{\text{high}}^{(L)}, \boldsymbol{\mathcal{H}}_{\text{high}}^{(L)}, A_{\text{high}}\right)\\ &= \{\mathcal{L}_g\boldsymbol{\mathcal{X}}_{\text{map}}, \boldsymbol{\mathcal{H}}_{\text{map}}, A\}\\ &= \mathcal{L}_g\mathcal{G}_{map}.\end{aligned}$$

□

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

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

Proof. We first prove that Equation (16) in Section 4.3 of the main text is invariant for agent i under the transformation $\mathcal{L}_g$:

$$\begin{aligned}&\phi_o\left(\left\|\mathcal{L}_g\hat{\mathcal{X}}_i\right\|^2, \left\|\mathcal{L}_g\hat{\mathcal{X}}_{i,\text{map}}\right\|^2, \mathcal{L}_g\mathcal{H}_i, \mathcal{L}_g\mathcal{H}_{i,\text{map}}\right)\\ =&\phi_o\left(\left\|\hat{\mathcal{X}}_i\right\|^2, \left\|\hat{\mathcal{X}}_{i,\text{map}}\right\|^2, \mathcal{H}_i, \mathcal{H}_{i,\text{map}}\right)\\ =&\mathcal{H}_{i,\text{out}},\end{aligned}$$

following this, we can prove Equation (17) in Section 4.3 of the main text is equivariant:

$$\begin{aligned}&\left(\mathcal{L}_g\mathcal{X}_{i,\text{map}} - \mathcal{L}_g\mathcal{X}_i^{(0)}\right)\mathcal{L}_g\mathcal{H}_{i,out}\\ =&\left(\mathcal{L}_g\mathcal{X}_{i,\text{map}} - \mathcal{L}_g\mathcal{X}_i^{(0)}\right)\mathcal{H}_{i,out}\\ =&\mathcal{L}_g\left(\left(\mathcal{X}_{i,\text{map}} - \mathcal{X}_i^{(0)}\right)\mathcal{H}_{i,out}\right)\\ =&\mathcal{L}_g a_i.\end{aligned}$$

Synthesizing the above derivations, when the input of $\mathcal{F}_{\text{ACT}}(\cdot)$ undergoes an $\mathcal{L}_g$ transformation, the output correspondingly undergoes an $\mathcal{L}_g$ transformation. Hence, it can be concluded that $\mathcal{F}_{\text{ACT}}(\cdot)$ is equivariant. □

2.2 Elaboration on Corollary 1

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

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

Proof. Based on Proposition 1 and section 3.4 of the main text, we have the flowing conclusion:

$$\begin{aligned}\boldsymbol{\mathcal{H}}^{(L)}, \mathcal{L}_g\boldsymbol{\mathcal{X}}^{(L)} &= \mathcal{F}_{\text{EGN}}(\boldsymbol{\mathcal{H}}^{(0)}, \mathcal{L}_g\boldsymbol{\mathcal{X}}^{(0)}, A),\\ \mathcal{L}_g\mathcal{G}_{\text{high}}^{(0)} &= \mathcal{F}_{\text{NFC}}\left(\mathcal{L}_g\mathcal{G}^{(L)}\right).\end{aligned}$$

Since both $\mathcal{F}_{\text{EGN}}(\cdot)$ and $\mathcal{F}_{\text{NFC}}(\cdot)$ involved in the computation of $\mathcal{F}_{\text{ECM}}(\cdot)$ are equivariant, $\mathcal{F}_{\text{ECM}}(\cdot)$ is also equivariant. □

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

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

Proof. Based on Proposition 1 and section 3.4 of the main text, we have the flowing conclusion:

$$\begin{aligned}\boldsymbol{\mathcal{H}}_{\text{high}}^{(L)}, \mathcal{L}_g\boldsymbol{\mathcal{X}}_{\text{high}}^{(L)} &= \mathcal{F}_{\text{EGN}}(\boldsymbol{\mathcal{H}}_{\text{high}}^{(0)}, \mathcal{L}_g\boldsymbol{\mathcal{X}}_{\text{high}}^{(0)}, A_{\text{high}}),\\ \mathcal{L}_g\mathcal{G}_{map} &= \mathcal{F}_{\text{NFR}}\left(\mathcal{L}_g\mathcal{G}_{\text{high}}^{(L)}\right).\end{aligned}$$

Since both $\mathcal{F}_{\text{EGN}}(\cdot)$ and $\mathcal{F}_{\text{NFR}}(\cdot)$ involved in the computation of $\mathcal{F}_{\text{NFR}}(\cdot)$ are equivariant, $\mathcal{F}_{\text{NFR}}(\cdot)$ is also equivariant. □

2.3 Elaboration on Corollary 2

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

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

Proof. Based on Proposition 1 and Corollary 1 of the main text, we are able to arrive at the following conclusion:

$$\begin{aligned}\mathcal{L}_g\mathcal{G}_{\text{high}}^{(0)} &= \mathcal{F}_{\text{ECM}}\left(\mathcal{L}_g\mathcal{G}^{(0)}\right),\\ \mathcal{L}_g\mathcal{G}_{\text{map}} &= \mathcal{F}_{\text{ERM}}\left(\mathcal{L}_g\mathcal{G}_{\text{high}}^{(0)}\right),\\ \mathcal{L}_g\boldsymbol{a} &= \mathcal{F}_{\text{ACT}}\left(\mathcal{L}_g\mathcal{G}^{(0)}, \mathcal{L}_g\mathcal{G}_{\text{map}}\right).\end{aligned}$$

Since the three modules of $\mathcal{F}_{\text{HEPN}}(\cdot)$, namely $\mathcal{F}_{\text{ECM}}(\cdot)$, $\mathcal{F}_{\text{ERM}}(\cdot)$, and $\mathcal{F}_{\text{ACT}}(\cdot)$, are all equivariant, $\mathcal{F}_{\text{HEPN}}(\cdot)$ is also equivariant. □

3 Implementation Details

3.1 HEPN for MARL

We deploy our policy network HEPN within the Actor-Critic architecture, trained using the paradigm of centralized training and decentralized execution (CTDE). Specifically, we embed independent HEPN within both the Actor and the Critic. In the Actor, $\mathcal{N}(i)$ represents the set of agents within the visual range of agent i, which can only access local information. In the Critic, $\mathcal{N}(i)$ encompasses all agents except for agent i, thereby enabling the calculation of state value based on global state. Moreover, the state value should not vary with geometric transformations, hence the HEPN in the Critic must satisfy invariance. To achieve this, we define the value function and Q function based on Equation (16) in the Action Module as:

$$Value = \phi_V\left(\boldsymbol{\mathcal{H}}_{\text{out}}\right), \tag{1}$$

$$Q = \phi_Q\left(\boldsymbol{\mathcal{H}}_{\text{out}}, \|\boldsymbol{a}\|_2\right), \tag{2}$$

where $\|\boldsymbol{a}\|_2$ consists of scalars since $\|\cdot\|_2$ denotes l_2 norm. The $Value$ and Q are $O(n)$-invariant due to both $\|\boldsymbol{a}\|_2$ and $\mathcal{H}_{out}$ are $O(n)$-invariant.

3.2 Algorithm Description

Our algorithm based on S-Dec-POMDP is described by Algorithm 1. Our ultimate objective is to develop the policy based on HEPN, which inherently possesses strict symmetry and derives actions by considering the hierarchical structure within the multi-agent system. At each step t of an episode, the initial system $\mathcal{G}^t$ is constructed based on the agents' initial observations, followed by obtaining the joint actions step by step as introduced in Section 4 of the main text. During the training phase, the hierarchical capability of HEPN is optimized through Partition loss, and the parameters of the actor and critic networks are updated. Notably, HEPN is a generic module that can generally be applied to different MARL algorithms.

4 Environmental Setting

In this section, we have supplemented the detailed settings of the environment, and will introduce the dynamics, action space, observation space, and reward function. Finally, we provide the hyperparameters for the experiments along with the software and hardware environments used.

Algorithm 1 The optimization of HEPN.

```
1: Input: Number of: agents $N$, episodes $K$, steps for episode $T$.
2: Initialize: HEPN based actor network $\{\theta_i\}_{i=1}^N$, HEPN based
   critic network $\phi$, replay buffer $\mathcal{D} \leftarrow \emptyset$, batch size $M$.
3: for $k = 0, 1, \dots, K-1$ do
4:     Initial state $s_t \leftarrow \{o_i^0\}_{i=1}^N$.
5:     for $t = 0, 1, \dots, T-1$ do
6:         Get $\mathcal{G}^t$ from $\boldsymbol{o}^t$.
7:         Get $\mathcal{G}^t_{\text{high}}$ via $\mathcal{F}_{\text{ECM}}(\cdot)$.
8:         Get $\mathcal{G}^t_{\text{map}}$ via $\mathcal{F}_{\text{ERM}}(\cdot)$.
9:         Obtain the joint action $\boldsymbol{a}^t$ form $\mathcal{F}_{\text{ACT}}(\cdot)$.
10:        Execute joint action $\boldsymbol{a}^t$ and collect the transition
           $\tau^t = \{\boldsymbol{s}^t, \boldsymbol{o}^t, \boldsymbol{a}^t, \boldsymbol{r}^t, \boldsymbol{s}^{t+1}, \boldsymbol{o}^{t+1}\}$.
11:        Push transition $\tau^t$ into replay buffer $\mathcal{D}$.
12:    end for
13:    if $k \geq M$ then
14:        Sample a minibatch $\mathcal{B} = \{\boldsymbol{s}_j^t, \boldsymbol{o}_j^t, \boldsymbol{a}_j^t, \boldsymbol{r}_j^t, \boldsymbol{s}_j^{t+1}, \boldsymbol{o}_j^{t+1}\}_{j=0}^M$
           from $\mathcal{D}$.
15:        Calculate the partition loss using Equation (11) in Section
           4.1 of the main text.
16:        Update parameter $\{\theta_i\}_{i=1}^N$ of the actor network.
17:        Update parameter $\phi$ of the critic network.
18:    end if
19: end for
```

4.1 Dynamics

In our environments, the dynamics of the agents are controlled by velocity in the x and y directions. This control mechanism can be mathematically modeled to facilitate analysis and simulation. The dynamics of each agent i are defined by its position (x_i, y_i) and velocity (v_{x_i}, v_{y_i}).

The state of agent i at any time t can be described by the equations:

$$\dot{x}_i(t) = v_{x_i}(t),$$
$$\dot{y}_i(t) = v_{y_i}(t),$$

where $\dot{x}_i(t)$ and $\dot{y}_i(t)$ represent the rate of change of the agent's position in the x and y directions, respectively. The velocities $v_{x_i}(t)$ and $v_{y_i}(t)$ are inputs to the system which can be controlled directly to guide the agent's movement.

To update the position of agent i, we integrate these rates over time. This results in the updated positions:

$$x_i(t + \Delta t) = x_i(t) + v_{x_i}(t) \cdot \Delta t,$$
$$y_i(t + \Delta t) = y_i(t) + v_{y_i}(t) \cdot \Delta t,$$

where Δt is the time step.

Additionally, this dynamical model is particularly advantageous for applications involving Simulation to Reality (Sim2Real) transfer. The high fidelity with which these models simulate real-world physics ensures that behaviors learned in the virtual environment can be effectively transferred to physical systems without significant loss of performance. This capability is critical in robotic applications where real-world testing is expensive or impractical.

4.2 Action Space

The action space of each agent is defined by the velocities in the x and y directions. Specifically, each agent can independently control its velocity along these two axes within prescribed limits. The set of all possible actions (v_{x_i}, v_{y_i}) for agent i is thus a continuous space bounded by the maximum velocity constraints v_{max_x} and v_{max_y}. Mathematically, the action space for agent i can be described as:

$$a_i = \{(v_{x_i}, v_{y_i}) \mid |v_{x_i}| \leq v_{max_x}, |v_{y_i}| \leq v_{max_y}\}$$

4.3 Observation Space

In this section, we will introduce the observation space. For each agent i, its observation can be represented as $\{\mathcal{H}_i, \mathcal{X}_i\}$.

Rendezvous. The observation $\{\mathcal{H}_i, \mathcal{X}_i\}$ for agent i is mathematically defined as follows:

$$\mathcal{H}_i = \{d_{ij}, \cos(\theta_{ij}), \sin(\theta_{ij}) \mid j \neq i\},$$
$$\mathcal{X}_i = \{x_i, y_i, v_{x_i}, v_{y_i}\},$$

where d_{ij} is the distance between agent i and another agent j. θ_{ij} is the angle between the direction of agent i and the line connecting it to agent j. x_i and y_i represent the position coordinates, and v_{x_i} and v_{y_i} denote the velocity components of agent i.

Pursuit. In the Pursuit task, the observation space for each agent i extends the concepts utilized in the Rendezvous task by incorporating additional elements to account for interactions with the prey. Mathematically, the observation space for agent i in the Pursuit task can be represented as:

$$\mathcal{H}_i = \{d_{ij}, \cos(\theta_{ij}), \sin(\theta_{ij}), d_{ie}, \cos(\theta_{ie}), \sin(\theta_{ie}) \mid j \neq i\}$$
$$\mathcal{X}_i = \{x_i, y_i, v_{x_i}, v_{y_i}\}$$

where $d_{ie}, \cos(\theta_{ie})$, and $\sin(\theta_{ie})$ measure the distance and the cosine and sine of the relative angle to the prey, respectively.

Resource Collection. In the Resource Collection task, the observation space for each agent i is specifically designed to facilitate efficient resource collecting by incorporating detailed information about the proximity and orientation of resource pools. The observation space can be mathematically represented as:

$$\mathcal{H}_i = \{d_{ij}, \cos(\theta_{ij}), \sin(\theta_{ij}), d_{ir}, \cos(\theta_{ir}), \sin(\theta_{ir}) \mid j \neq i\},$$
$$\mathcal{X}_i = \{x_i, y_i, v_{x_i}, v_{y_i}\},$$

where d_{ir}, $\cos(\theta_{ir})$, and $\sin(\theta_{ir})$, specifically provide the distance, the cosine and sine of the angle to the nearest resource pool.

4.4 Reward Function

In this section, we will introduce the reward functions for different tasks.

Rendezvous. In the Rendezvous task, the reward function is designed to encourage agents to minimize the distance between each other, thereby facilitating their aggregation at a single point. This objective is quantitatively expressed by setting the reward function as the negative sum of the distances between all pairs of agents. Furthermore, to prevent excessively large actions, which can lead to unstable or inefficient behaviors, a penalty is applied to large action outputs. The collective reward function for the environment can be defined as:

$$R = -\sum_{i=1}^{N} \sum_{j=i+1}^{N} d_{ij} - \lambda \sum_{i=1}^{N} \left(\|a_i\|_2\right),$$

where d_{ij} denotes the distance between agent i and j, $\|a_i\|_2$ represent the l_2 norm of the action of agent i, N is the total number of agents, and λ is a tuning parameter that controls the penalty for the magnitude of the action outputs.

Pursuit. In the Pursuit task, the reward function is designed to minimize the distance between the prey and the closest pursuing agent. This design encourages the pursuers to effectively coordinate their movements to capture the prey as quickly as possible. The reward function for this environment can be expressed as follows:

$$R = -\min_{i \in \text{pursuers}} d_{ie},$$

where d_{ie} represents the distance between the prey and each pursuing agent i. The function $\min(\cdot)$ ensures that the reward is maximized when the distance between the prey and the nearest pursuer is minimized.

Resource Collection. In the Resource Collection task, the reward function is designed to optimize resource gathering while ensuring safe navigation. The primary objective is to minimize the distance between agents and resource pools, facilitating efficient resource collection. Simultaneously, the reward function penalizes collisions with obstacles and other agents to promote safe moving. The reward function can be expressed as:

$$R = -\sum_{i=1}^{N} \min_{r \in \text{pools}} d_{ir} - \alpha C - \beta \sum_{r \in \text{pools}} |n_r - \bar{n}|,$$

where d_{ir} is the distance between agent i and resource pool r, C represents the penalty incurred from collisions with obstacles or other agents, n_r is the number of agents at resource pool r, $\bar{n}$ is the average number of agents per resource pool, aiming for a uniform distribution. α and β are weighting factors that adjust the importance of avoiding collisions and balancing agent distribution, respectively.

4.5 Hyperparameter settings

In this section, we present the hyperparameter settings used for each algorithm in our experiments. In this paper, we employ parameter sharing to train the model for homogenous agents. Our HEPN can also applied to heterogeneous task. To ensure optimal performance of our models, a detailed adjustment of a range of hyperparameters is conducted. This section comprehensively lists the hyperparameter settings for each model, facilitating the replication of our experimental results and further research. In this paper, we deploy HEPN within the MAPPO [1]. The hyperparameters used are shown in Table 1. The hyperparameter settings are consistent with those used in HARL [2].

Table 1. Hyperparameter settings

Hyperparameter	Value
episode length	200
use valuenorm	True
use linear lr decay	True
use proper time limits	True
use feature normalization	True
hidden size	128
actor learning rate	5e-4
critic learning rate	5e-4
optimizer epsilon	1e-8
weight decay	0
ppo epoch	5
critic epoch	5
use clipped value loss	True
clip param	0.2
actor num mini batch	1
critic num mini batch	1
entropy coefficient	0.01
value loss coefficient	1
gamma	0.99
use huber loss	True
huber delta	10.0
use gae	True
gae lambda	0.95
clip param	0.2

4.6 Computing Environment

Our experiments were conducted on a Linux server based on the 12th Gen Intel(R) Core(TM) i9-12900KF processor. This server is equipped with 16 cores, 24 threads, and 24GB of RAM. Additionally, we utilized two GPUs for accelerating training: the NVIDIA GeForce RTX 3080 and the NVIDIA GeForce RTX 3090 Ti. The operating system version is 20.04.1. All of our experiments were carried out using Python version 3.8.18, with PyTorch version 2.0.0+cu117.

4.7 Physical Deployment

We conduct real-world experiments using a set of Limo robots, a robotic development platform that integrates four types of locomotion modalities and is based on ROS (Robot Operating System), suitable for research and educational purposes. One of the advantages of this platform is its convenient control via Mecanum wheels. During the experiments, we designate a host computer as the ROS master node and use the output actions from a trained neural network model as ROS commands to directly control the Limo robots. The experiments take place in a $2.8m \times 2.8m$ indoor area, where we employ a motion capture system to acquire real-time locations of the robots. This data is then used to compute the observation inputs required by the policy network. We compare our method, HEPN, against the best-performing baseline algorithms. For the Rendezvous task, we select ESP; for the Pursuit task, MLP is chosen; and for the Resource Collection task, we use GCS.

Acknowledgements

By using the `ack` environment to insert your (optional) acknowledgements, you can ensure that the text is suppressed whenever you use the `doubleblind` option. In the final version, acknowledgements may be included on the extra page intended for references.

References

[1] 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.

[2] 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.