Front. Comput. Sci., 2025, 19(11): 1911365
https://doi.org/10.1007/s11704-025-41250-2

RESEARCH ARTICLE

RDHNet: addressing rotational and permutational symmetries in continuous multi-agent systems

Dongzi WANG[1], Lilan HUANG[1], Muning WEN[2], Yuanxi PENG[1], Minglong LI (✉)[1], Teng LI (✉)[3]

1 College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China
2 School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
3 College of Advanced Interdisciplinary Studies, National University of Defense Technology, Changsha 410073, China

Abstract Symmetry is prevalent in multi-agent systems. The presence of symmetry, coupled with the misuse of absolute coordinate systems, often leads to a large amount of redundant representation space, significantly increasing the search space for learning policies and reducing learning efficiency. Effectively utilizing symmetry and extracting symmetry-invariant representations can significantly enhance multi-agent systems' learning efficiency and overall performance by compressing the model's hypothesis space and improving sample efficiency. The issue of rotational symmetry in multi-agent reinforcement learning has received little attention in previous research and is the primary focus of this paper. To address this issue, we propose a rotation-invariant network architecture for continuous action space tasks. This architecture utilizes relative coordinates between agents, eliminating dependence on absolute coordinate systems, and employs a hypernetwork to enhance the model's fitting capability, enabling it to model MDPs with more complex dynamics. It can be used for both predicting actions and evaluating action values/utilities. In benchmark tasks, experimental results validate the impact of rotational symmetry on multi-agent decision systems and demonstrate the effectiveness of our method. The code of RDHNet has been available at the website of github.com/wang88256187/RDHNet.

Keywords multi-agent, reinforcement learning, symmetry

1 Introduction

In recent years, multi-agent reinforcement learning (MARL) has made significant advancements [1] and has been widely applied in areas such as traffic planning [2,3], power management [4], gaming [5–7], and Robotics [8,9]. However, compared to single-agent reinforcement learning, MARL faces more challenging issues, including credit assignment [10,11], scalability [3,12], communication [13] and imperfect information games [14]. In this paper, we primarily focus on the symmetry problem in MARL.

Received November 19, 2024; accepted February 27, 2025

E-mail: liminglong10@nudt.edu.cn; liteng09@nudt.edu.cn

Symmetry is widely present in the natural world, and many studies in fields such as image processing, graph neural networks, and point clouds have focused on symmetry. In multi-agent reinforcement learning (MARL), research on symmetry is still in its infancy. Symmetry can be broadly categorized into permutation symmetry and rotational and mirror symmetry (in this paper, we refer to both rotational and mirror symmetry as rotational symmetry, as they can be derived from orthogonal transformations). Some research has been conducted on permutation symmetry in MARL [15–18], and HPN [17], with the most notable work being HPN. By designing a network structure with permutation-invariant inductive bias, permutation-invariant representations can be naturally obtained, leading to the corresponding action outputs. However, research on rotation invariance (RI) in MARL is still in its early stages.

Rotational symmetry is prevalent in the real world. For instance, if a person learns traffic rules at a driving school, they can easily generalize them to real-world traffic conditions without considering whether they are facing north or south. Similarly, a basketball team, after training, can play in a new arena without worrying about the actual orientation of the court. They only need to know their relative positions to teammates, opponents, and the basket. Humans naturally incorporate prior knowledge of rotational symmetry into specific tasks, resulting in higher sample efficiency when learning a new skill. However, this is not the case for machines. Without special consideration of rotation invariance, machines cannot generalize knowledge learned from state s to a rotated state s', shown in Fig. 1.

Some works [19–21] define the symmetry method and quantity for the agent's coordinate system, which improves the algorithm's performance on specific MARL tasks. However, it still cannot handle continuous rigid transformations in Euclidean space. Continuous rigid transformations involve rotating the agent's coordinates around a fixed point by arbitrary angles rather than assuming rotations are only multiples of 90 degrees or simple mirror reflections. Continuous transformations are more representative of real-world scenarios. Those two

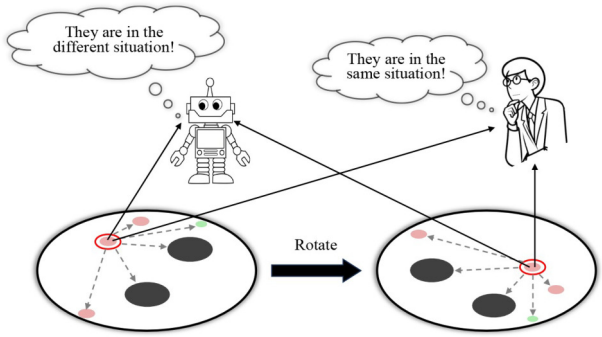

Fig. 1 Issue of redundant representations caused by rotational symmetry. In the figure, nodes of different colors and sizes represent various entities. At a particular moment, state s is rotated by a random angle, resulting in state s'. If the human is the agent marked by the red circle in the figure, they can easily perceive the equivalence between s and s' and take appropriate actions. However, suppose the agent marked in the figure is a robot. In that case, it may fail to recognize the relationship between s and s', making it difficult to generalize the knowledge gained from one state to another, demonstrating how rotation can create redundant state representations in MARL

transformations are shown in Fig. 2. Theoretically, if a reasonable model or algorithm could account for rotational symmetry, an agent could generalize knowledge learned from state s to the rotated state s', thereby significantly reducing the state representation space.

We propose a Relative Direction Hypernetwork (RDHNet) architecture to extract relative directional and positional information and then use a symmetry computation module to aggregate this information. Effectively leveraging symmetry compresses redundant representation space, facilitating better knowledge sharing among agents and enhancing learning efficiency and robustness. Additionally, we utilize the high-order modeling capabilities of the hypernetwork module to improve the model's expressiveness, enabling it to handle tasks with more complex dynamics. RDHNet can be used to construct not only an action value/utility evaluation network but also a policy network that produces actions based on input states.

We conducted experiments on two continuous action tasks, *Cooperative Prey Predator* and *Cooperative Navigation*. The results demonstrate that, compared to other methods, RDHNet achieves superior performance. Moreover, during execution, RDHNet does not require absolute coordinate and directional information; it can make appropriate decisions based solely on relative observations from the agent's perspective, indicating that in situations where geographic positioning information is unavailable, our method remains effective, whereas other methods may fail to function. We summarize our contributions as follows:

- We formalized the symmetry problem in multi-agent systems, classifying it into permutation symmetry and rotational symmetry, providing a clearer problem definition for future research on symmetry.
- We propose a network architecture that can eliminate the redundant representation space caused by symmetry without significantly reducing the network's expressiveness.
- We compare different methods on benchmark tasks to analyze and confirm the real impact of symmetry on MARL problems. We also conduct separate ablation studies for rotation invariance (RI) and permutation invariance (PI).

2 Related works

2.1 Symmtry in GNNs and point clouds

Recent advances in symmetry studies within Graph Neural Networks (GNNs) and point cloud processing have significantly influenced the development of invariant models to permutations, rotations, and translations. For instance, the introduction of Graph Convolutional Networks (GCNs) [22] laid the foundation for permutation invariance in GNNs. In the context of point clouds, PointNet [23] and its successor PointNet++ [24] pioneered approaches that handle unordered point sets directly, enabling permutation invariance.

Building on these foundational works, recent studies have explored rotational and translational equivariance in graph neural networks (GNNs) and point cloud networks. Tensor Field Networks [25] and SE(3)-Transformers [26] exemplify this trend by incorporating roto-translation equivariant features, enabling more accurate and robust processing of 3D point clouds and molecular structures. Additionally, work [27] extended the expressive power of GNNs by proving the capability of certain architectures to universally approximate permutation-invariant functions, further advancing the theoretical understanding of symmetry in GNNs. DimeNet [28] captures angular dependencies between atoms in a molecular graph, which enhances the network's ability to model geometric structures without being affected by the specific orientation or position of the molecule in space.

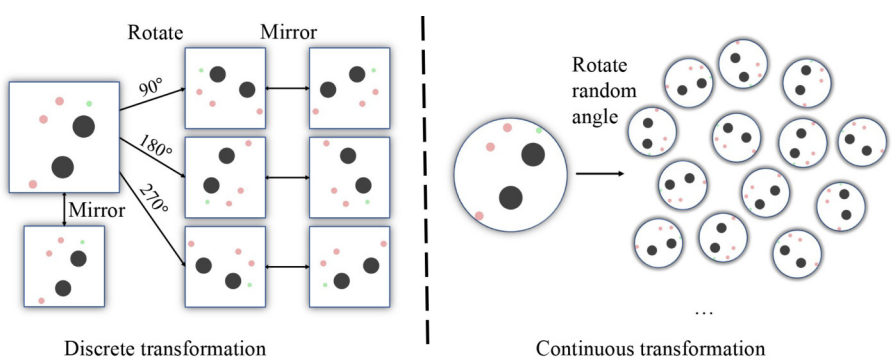

Fig. 2 Difference between discrete transformation and continuous transformation

2.2 Symmtry in MARL

Some works have begun to address the issue of redundant space caused by symmetry in multi-agent reinforcement learning (MARL). Methods such as ASN [15], UPDeT [16], and HPN [17] have focused on the redundancy problem due to permutation symmetry. They modify the actor network by incorporating prior knowledge to compress the redundant representation space caused by permutation order. The primary differences among them are: ASN [15] uses MLP as the basic network module, UPDeT [16] uses transformers, and HPN [17] uses hypernetworks. Essentially, they all integrate permutation symmetry into the network inductive bias to address the redundancy problem caused by different permutations of semantically identical states. However, these approaches do not address the issue of rotational symmetry.

Work [19] was the first to address the rotational symmetry issue in MARL, which involves symmetry operator pairs (L_g, K_g) to the model. Works [20,21] generate rotated samples during training and introduce a rotational symmetry loss function to exploit symmetry. However, a major limitation of these works is that they can only handle rotational symmetry at multiples of 90 degrees. They neither consider nor can be applied to continuous random rotational symmetry, which is precisely the focus of our work and is more aligned with real-world scenarios.

3 Problem formulation

A Multi-Agent Reinforcement Learning (MARL) problem is often considered a Markov decision process, which is a tuple of the form $(\mathcal{N},\mathcal{S},\mathcal{O},\mathcal{A},T,R,\rho,\gamma,\pi)$. Here, $\mathcal{N}$ is the set of all agents, $\mathcal{S}$ is the set of states, $\mathcal{O}=\mathcal{O}^{\infty}\times\mathcal{O}^{\in}\cdots\times\mathcal{O}^{|\mathcal{N}|}$ is the set of joint observations where $\mathcal{O}^k$ indicates the set of kth agent' observations, $\mathcal{A}=\mathcal{A}^{\infty}\times\mathcal{A}^{\|}\times\cdots\times\mathcal{A}^{|\mathcal{N}|}$, $T:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\to[0,1]$ is a transition function that specifies the probability of reaching state $s'\in\mathcal{S}$ after all agents taking their joint action $\boldsymbol{a}$ in state s, reward function $R:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ is to return a reward value to agents to evaluate the quality of the agent's action, ρ is the initial state distribution, γ is the discount factor. At one timestep, each agent needs to take action to the environment based on their observation, and then the state of the environment will change naturally. After this, agents will be rewarded $r=R(s,a)$ and new observations. A joint policy π is a function that maps joint observation $\boldsymbol{o}\in\mathcal{O}$ to joint action $\boldsymbol{a}\in\mathcal{A}$. π often can be divided into a set of all agents' policies $(\pi^1,\ldots\pi^k\ldots,\pi^{|\mathcal{N}|})$, where π^k is the policy function that maps kth agent' observation to its action. The core objective of MARL is to find the optimal policy $\pi^*=\arg\max_\pi\sum_{t=0}^{\infty}\mathbb{E}_{(o_t,a_t)\sim\rho_\pi}[\gamma^t r(s_t,a_t)]$, where the subscript t represents the variable value at the tth timestep. However, if we directly handle a MARL problem as formalized above without addressing symmetry, it typically results in a substantial amount of redundant representations, leading to low sample efficiency.

Permutation symmetry occurs when the order of agent information does not affect the overall observation information, resulting in redundant representation space due to the fixed order setting. Let $o_i=\boldsymbol{e}=[e_1,e_2,\ldots,e_m]^T$, where e_k denotes the information of the k-th entity (an entity can be an enemy, a teammate, or other things that can influence agent making decision) observed by agent i. Specifically, let $\boldsymbol{e}=[e_1,e_2,\ldots,e_m]^T=\boldsymbol{x}=[x_1,x_2,\ldots,x_m]^T$. When the order of the entities' information in o_i changes, i.e., $\boldsymbol{e}=[e_1,e_2,\ldots,e_m]^T=[x_1,x_2,\ldots,x_m]^T\cdot g, g\in G$, where G is the set of all permutation matrices and g is a specific permutation matrix in G. It can be seen that regardless of the value of g, the physical information contained in o_i remains unaffected. In other words, the permutation order of the entity information is actually redundant in the decision-making process of agent i. However, if we do not address this issue, the permutation-induced symmetry will lead to a large amount of redundant representation space, i.e., $\boldsymbol{x}\cdot g_1,\boldsymbol{x}\cdot g_2,\ldots$ will all be recognized by the model as different observation information, thus resulting in redundant space. A function $f:X\to Y$, where $X=[x_1,x_2,\ldots,x_m]^T$ and $Y=[y_1,y_2,\ldots,y_m]^T$. If f satisfies $f([x_1,x_2,\ldots,x_m]^T*g)=f([x_1,x_2,\ldots,x_m]^T)=[y_1,y_2,\ldots,y_m]^T$, $\forall g\in G$, then we call f a permutation-invariant function. If we can construct a model that satisfies the properties of the f function, then permutation symmetry can be effectively addressed.

Rotational symmetry refers to the condition where rotating a state as a whole does not affect the relative physical information contained within that state. This implies that many rotational mappings of the same state lead to representational space redundancy. Let us further refine the observation o_i. We distinguish the features of each entity into spatial coordinates p_k and other information z_k, so the observation can be represented as $o_i=[[p_1,z_1]^T,[p_2,z_2]^T,\ldots,[p_m,z_m]^T]$. We then reconstruct o_i as $[[p_1,p_2,\ldots,p_m]^T,[z_1,z_2,\ldots,z_m]^T]$. Consider a binary function $h:(P,Z)\to Y$, where $P=[p_1,p_2,\ldots,p_m]^T$, $Z=[z_1,z_2,\ldots,z_m]^T$, and $Y=[y_1,y_2,\ldots,y_m]^T$. U is the set of all $m\times m$ rotation matrices, and $u\in U$ is a specific rotation matrix. If $h(P\cdot u,Z)=h(P,Z)=Y,\forall u\in U$, then h is called a rotation-invariant function. If a model satisfies the properties of the h function, we can say that the model possesses rotational invariance.

4 Rotational-invariant multi-agent reinforcement learning

Rotation-invariant multi-agent reinforcement learning (MARL) does not require an absolute coordinate reference system; it only considers the relative positional relationships between entities. This approach aligns more closely with human intuition and facilitates better knowledge generalization. This chapter provides a detailed overview of the RDHNet framework's structure, which is shown in Fig. 3.

4.1 Directional information proccess

To address the issue of rotational symmetry, it is first necessary to distinguish which elements are direction-related and which are direction-independent in a decision-making scenario involving multiple entities. Information such as the volume and mass of entities is typically direction-invariant. In contrast, the position, motion direction, and force direction of entities are closely related to the reference direction (changing

with the reference frame).

Figure 4 shows the observation of agent i at a specific moment in the *Cooperative Prey Predator* task. In this observation, it can be seen that regardless of the choice of reference direction, the angles and lengths of the lines connecting each entity to agent i do not change. We need to leverage this invariance in multi-agent scenarios to compress the representation dimensions of observation information. This involves deconstructing and reorganizing the original information structure, replacing the reference frame based on absolute direction with a reference frame defined by the line connecting a specific entity and the agent itself.

As an example, consider the observation shown in Fig. 4. First, a reference entity j is selected. Then, with agent i as the origin and the line between agent i and entity j as the zero-degree direction, a polar coordinate system is established, and the polar coordinates of each entity in this system are calculated. After the Relative Direction Layer (RDL), the information of kth entity E_k should be processed into a tuple $(z_{ijk}, d_{ijk}, \theta_{ijk})$ by the following equation:

$$\begin{aligned} z_{ijk} &= z_k, \\ d_{ijk} &= (x_k - x_i, y_k - y_i), \\ \theta_{ijk} &= \arctan\frac{y_k - y_i}{x_k - x_i} - \arctan\frac{y_j - y_i}{x_j - x_i}, \end{aligned} \tag{1}$$

where z_{ijk} represents some attributes of entity k that are independent of directional changes, while d_{ijk} and θ_{ijk} correspond to the length and angle, respectively, in the polar coordinate system of entity k. In this system, the zero axis is defined by the line from entity i to entity j, with the pole located at the position of entity i. In practical processing, since $\arctan(x)$ is an odd function with a period of π, it results in two possible values within a 2π range. This requires discarding one of the values based on the specific context, complicating efficient algorithm execution. The same issue arises when $\arcsin(x)$ and $\arccos(x)$ are used independently. To address this issue, we propose replacing θ_{ijk} in Equation (1) with $m_{ijk} = \left[\sin(\theta_{ijk}), \cos(\theta_{ijk})\right]$ for representing the angle information. More detail about computing $\left[\sin(\theta_{ijk}), \cos(\theta_{ijk})\right]$ is in **Appendix C**.

If subsequent evaluation of action is needed, the actions must also be processed in the "Relative Direction Layer". Here, the action (usually a force vector) of agent i originally based on the absolute coordinate system is converted into a tuple (l_{ij}, ϕ_{ij}) based on the aforementioned polar coordinate system.

At this point, the information of each entity has been separated and processed individually. After passing through the Relative Direction Layer, the positional information of entities in the environment is fully decoupled from the absolute coordinate system, making their location data solely dependent on their relative positions to other entities.

4.2 Information embedding and aggregation

Once the direction-dependent and direction-independent information of each entity is separated, the next step is to encode and aggregate this entity information. As shown in Fig. 3, when using entity j as a reference, an information tuple $(z_{ijk}, d_{ijk}, m_{ijk})$ for entity k can be obtained. Each element of the tuple is encoded in different ways: the direction-independent information z_{ijk} is encoded using a multilayer perceptron (MLP), denoted as Emb_z; the distance information d_{ijk} is encoded using radial basis functions (RBF), denoted as Emb_d; and the angle information is encoded using sine and cosine functions. For action value evaluation, the action vector a_i is similarly processed, with the relative angle ϕ_{ij} encoded using sine and cosine functions and the magnitude of a_i encoded using RBF. Here, every entity's information is encoded into $e_{ijk} = [Emb_z(z_{ijk}), Emb_d(d_{ijk}), Emb_\theta(m_{ijk})]$. The encoded information e_{ijk} of entity k inputs into a specific

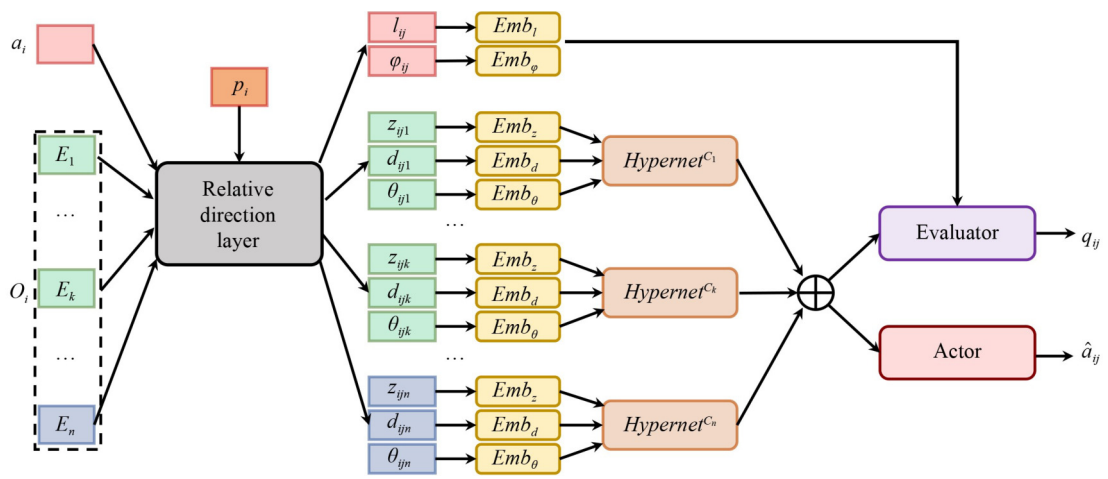

Fig. 3 The overall framework of RDHNet. Initially, the information E_k of the kth entity in the agent's observation is decomposed into coordinates p_k and direction-independent information z_k. Then Relative Direction Layer (RDL) maps the positions of entities i and j, converting the Cartesian coordinates P_k into the corresponding polar coordinates (d_{ijk}, θ_{ijk}). If we need RDHNet to output the action value/utility, the agent's action vector a_i should also be transferred to l_{ij} and the relative angle ϕ_{ij} by the RDL. Each type of information is then encoded by its respective encoder and input into the hypernetworks for each entity type. The symmetry module aggregates this information to obtain direction-invariant representations. Finally, these representations are input into the actor or critic module to output the corresponding actions a_{ij} or action values q_i

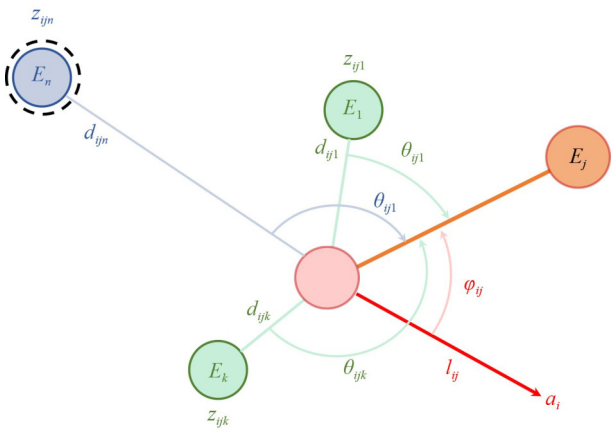

Fig. 4 An observation from agent in *Cooperative Prey Predator*. This figure is an example of an observation, where circles of different colors represent different types of entities (corresponding to Fig. 3). Regardless of the state's rotation, the variables shown in the figure remain unchanged in the polar coordinate system where the zero axis is defined by the agent and entity j. Specifically, this means that the length of the line segments connecting each entity to the agent and their angles with respect to the zero axis are invariant

hypernetwork to obtain an overall abstract representation of the entity. To enhance the network's expressive capability, different hypernetworks are used for different entity categories. For example, if the category of k is C_k, the corresponding hypernetwork $Hypernet^{C_k}$ is used, whose structure is generally consistent with that in [17]. Finally, the abstract representations of each entity are aggregated using a symmetric aggregation function, which ensures that the output is invariant to the permutation of input variables (i.e., a permutation-invariant function f as defined in Section 3). This representation can be computed as:

$$\tilde{o}_{ij} = \sum_{k \in \mathcal{N}(i), k \neq j} \mathcal{M}_H^{C_k}(e_{ijk}), \tag{2}$$

where $\mathcal{M}_H^{C_k}$ refers to the hypernet for the class of entity k, and $\mathcal{N}(i)$ is an entity set that contains all the entities in the entity i's view. By this point, we have effectively utilized both permutation symmetry and rotational symmetry, achieving compression of redundant representations $\tilde{o}_{ij}$ for entity i's observation based on the position of entity j (i.e., the symmetric physical information is always processed to yield the same representation). Here, if it is necessary to evaluate the action utility value, a_{ij} will be similarly calculated.

4.3 Algorithm inplementation

Theoretically, RDHNet can be combined with any mainstream continuous-action MARL algorithm. However, in practice, we chose COMIX [29], which currently demonstrates state-of-the-art performance in continuous multi-agent tasks, as the foundation for our implementation. We adopt the COMIX scheme in FACMAC [30], which employs the cross-entropy method (CEM) [31] by QMIX-style [29]. Specifically, both observations and actions are processed to obtain their corresponding rotation-invariant representations $\tilde{o}_{ij}$ and $\tilde{a}_{ij}$. Action utility value is computed by $q_{ij} = M_E(\tilde{o}_{ij}, \tilde{a}_{ij})$, where the q_{ij} denotes that the utility is computed with respect to reference entity j, and the M_E is the evaluation network in Fig. 3. We can select reference j based on different strategies and obtain q_i from q_{ij} (which will be discussed in detail later). Once q_i is obtained, the mixer network can compute Q_{tot} as following:

$$Q_{\text{tot}} = w_1(s)q_1 + w_2(s)q_2 + \cdots + w_n(s)q_n + b(s), \tag{3}$$

where $w_i(s)$ are weights generated by a hypernetwork (conditioned on the global state s) that ensures all $w_i(s) \geqslant 0$ to maintain monotonicity. $b(s)$ is a state-dependent bias, also generated by the hypernetwork. In practice, the weights $w_i(s)$ and the bias $b(s)$ are produced by a separate hypernetwork that takes the state s as input. This structure can factorize the joint action-value function Q_{tot} while respecting the monotonicity condition, making decentralized execution possible.

During training, the loss for the final parameter update follows the Bellman equation as described in QMIX:

$$L(\eta) = \sum_{i=1}^{b} \left(r + \gamma \max \overline{Q}_{tot} - Q_{tot}\right)^2, \tag{4}$$

where the overline denotes the target network, the target network is typically updated with a delay, and its gradients are frozen during gradient ascent to enhance training stability [32].

How to choose the referred entity?

Regarding the selection of the reference entity j, we employ a simple yet feasible approach —alternately selecting each entity within the observation range as the reference target j. The final output (either a_{ij} or q_{ij}) is averaged over all reference entities to reduce variance and enhance training stability. For example, the final action value $q_i = \frac{1}{n}\sum_{k=0}^{n} q_{ik}$, as this approach helps to reduce variance and improve training stability. Additionally, if performance is not the only consideration, such as when computational overhead is also a factor, we have employed alternative selection methods. A detailed analysis of the Computational Complexity will be presented in the next subsection, and the experimental results will be shown in **Section** 5.

The detailed processing steps of RDHNet are outlined in Algorithm 1. At the same time, Algorithm 2 provides a detailed description of the process by which RDHNet, combined with CEM [31], outputs actions and their utility.

4.4 Improvement for large-scale tasks

For tasks involving a large number and variety of entities, the initial version of RDHNet does indeed introduce significant computational costs. As analyzed in **Appendix A**, the computational complexity is $O(n^3)$, where n represents the number of agents. This results in substantial computational overhead for large-scale tasks. To address this issue in tasks with a large number of agents, we propose an optimization strategy. Specifically, we set an upper limit k on the number of reference agents when constructing the polar coordinate system. When the number of entities $n \leqslant k$, a polar coordinate system is constructed and computed for each entity. When $n > k$, only k entities are selected from the n entities to construct the polar coordinate system, reducing the

computational complexity to $O(n^2)$, which is also analyzed in **Appendix A**. Various strategies can be used to select the entities, such as choosing the k nearest entities, selecting entities of different types, or uniformly sampling entities based on distance. We believe the specific selection strategy should depend on the task scenario. In our task, we adopted the strategy of selecting the k nearest agents. The impact of different selection quantities on the model's performance is evaluated in the **Section** 5.

5 Experiments

In this section, we utilize experiments to validate the effectiveness of our method. Across two types of tasks, RDHNet demonstrates significant superiority when compared to state-of-the-art MARL algorithms for continuous action tasks.

5.1 Task domains and baseline

We tested the algorithm on two types of tasks, which are both developed based on the MPE (multiagent-particle-envs) [33]. MPE features highly realistic physical mechanisms, such as inertia and air damping, making it highly suitable for validating the performance of our algorithm. The first task is *Prey Predator (PP)*, in which the prey moves at a slower speed following a fixed strategy, and the goal is to control the predators to get as close to the prey as possible. The second task is a *Cooperative Navigation (CN)*, where the objective is to control the agents to cover every target point. We adopted a simplified strategy in both tasks to focus on the research problem studied in this paper, where agents use global observation information in all algorithms. In both tasks, the agents' actions are force vectors, and there are obstacles present. The agents need to avoid obstacles as much as possible while achieving their task goals.

Through our investigation, we found that there are relatively few multi-agent reinforcement learning (MARL) methods specifically targeting continuous action spaces. Therefore, we selected FACMAC [30], MADDPG [33], independent DDPG (IDDPG) [34], as well as COVDN [30] and COMIX [30], as baselines to evaluate the performance of our approach. COVDN and COMIX employ the cross-entropy method (CEM) [31] to successfully extend value decomposition-based methods, such as VDN [35] and QMIX [29], to continuous action problems.

To ensure the fairness of the comparative experiments, we made every effort to keep the common parameters consistent. Since all the methods used in this paper are off-policy, maintaining parameter consistency is relatively easy. The parameters listed in Table 1 are common to all methods, and thus they remain the same across all experiments. Additionally, to mitigate the influence of randomness, all experiments in this study were conducted with different random seeds. This strict control of parameter consistency is crucial to ensure the validity of our experimental results.

Algorithm 1 RDHNet based on COMIX

input : ξ: RDHNet
ω:Mixer network
T: total timestep
n: number of agents
K_i: timesteps of the parameters update interval
$\mathcal{B}$: replay buffer
K_b: sampling threshold of the replay buffer

Initialize all network parameters;
Initialize the replay buffer $\mathcal{B}$;
for $t = 1, 2, ..., T$ **do**
  Obtain each agent's observation $\mathbf{o} = \{o_i\}_{i=1}^n$ *and global state* s;
  for *agenti* $= 1, 2, ..., n$ **do**
    foreach *agent* $\in \mathcal{N}(i)$ **do**
      compute q_i *and* a_i *by* ***Algorithm*** *2 with inputing* (o_i^t, ξ) ;
  Execute joint action $\mathbf{a} = \{a_i\}_{i=1}^n$, *and obtain reward* r;
  Store $(s, \mathbf{o}, \mathbf{a}, r)$ *to* $\mathcal{B}$;
  if t *mod* $T_i = 0$ *and* $|\mathcal{B}| > K_b$ **then**
    Sample a batch of trajectories from $\mathcal{B}$;
    Compute $Q_{tot} = f_\omega([q_1, q_2, ..., q_n])$ *for each data by Equation 3*;
    Compute the batch data loss $\sum L(\xi, \omega)$ *by Equation 4*;
    Update parameters ξ, ω *by gradient descent*;

output : RDHNet with the optimal parameters ξ^*

Algorithm 2 CEM with RDHNet

input : ξ: RDHNet
o_i: observation of agent i
K_c: iteration number of CEM
K_s: number of samples in every CEM iteration

Initialize the Gaussian distribution $U(\mu, \sigma^2)$ *as a standard normal distribution*;
for $c = 1, ..., K_c$ **do**
  Sample K_s *actions* $\{a_i^k\}_{k=1}^{K_s}$ *from* $U(\mu, \sigma^2)$;
  foreach *agent* $j \in \mathcal{N}(i)$ **do**
    Compute: $q_{ij}^k = f_\phi(o_i, a_i^k, p_j)$ for each element in $\{a_i^k\}_{k=1}^{K_s}$ by Equation (1∼ 2);
    Compute the optimal action $a_i^* =_k q_{ij}^k$;
    Compute the utility of the optimal action $q_{ij} = \max_k(q_{ij}^k)$;
    Let $\mu = a_i^*$;
Compute the utility of agent i: $q_i = \frac{1}{|\mathcal{N}(i)|} \sum_j q_{ij}$ *Let* $a_i = a_i^*$.;

output : q_i, a_i

Table 1 Hyperparameters used for RDHNet based on COMIX

Hyperparameter	Value	Hyperparameter	Value
buffer size	5000 episodes	act noise	0.1
batch size	32	evaluate interval	2000
learning rate	0.01	grad norm clip	0.5
exploration mode	gaussian noise	optimizer	adam
discount factor	0.99	target update interval	200
hypernet hidden size	64	hypernet layers	2
hypernet class	3	target update weight	0.001

5.2 Performance compared with baseline

Figure 5 illustrates the convergence performance of each algorithm during training. From the Fig. 5, we can observe that in the *Cooperative Navigation* tasks, RDHNet consistently converges quickly and stably to the best results. Meanwhile, COMIX, COVDN, IDDPG, and MADDPG also achieve relatively good convergence, but FACMAC underperforms significantly, exhibiting poor convergence speed, final performance, and stability. In the *predator prey* tasks, RDHNet also achieves the fastest convergence and best performance across both scenarios. Furthermore, we observe that as the number of entities in the *predator prey* tasks increases, the advantages of RDHNet become more pronounced. Specifically, RDHNet demonstrates overwhelming superiority in the scenario with six predators compared to the other algorithms. We speculate that this is because the representation space is larger in scenarios with more entities, allowing RDHNet, which leverages rotational symmetry to compress the representation space, to have a more significant advantage. Table 2 lists the performance of all models after trained, showing that RDHNet achieves the best performance in four out of five tasks. The results confirm the objective presence of rotational symmetry in MARL tasks and demonstrate the effectiveness of our proposed method in addressing this issue.

5.3 Ablation study

To assess the impact of different symmetries in MARL, we conducted ablation experiments focusing on rotational and permutation symmetries. Given that rotational invariance is built upon permutation invariance, our experimental setup compared three scenarios: baseline, baseline with IE and PE, and baseline with only PE. The baseline used was COMIX, which is considered the best MARL algorithm for continuous action tasks. For implementing permutation invariance (PI), we replaced the MLP in COMIX with the HPN network. The method incorporating both permutation invariance (PI) and rotation invariance (IE) was our proposed RDHNet network.

The results, shown in Fig. 6, reveal that introducing PE into

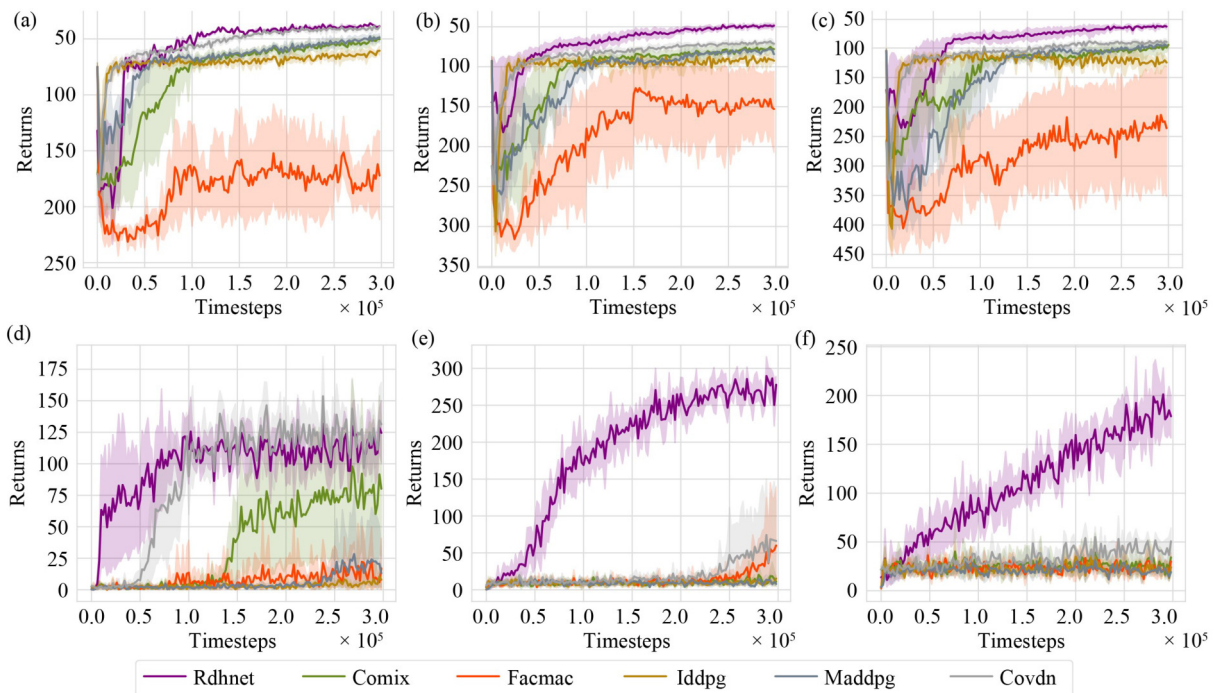

Fig. 5 Performance comparison between RDHNet and other SOTA algorithms. (a) Performance on *Cooperative Navigation* with 3 agents; (b) performance on *Cooperative Navigation* with 5 agents; (c) performance on *Cooperative Navigation* with 7 agents; (d) performance on *predator prey* with 3 agents; (e) performance on *predator prey* with 6 agents; (f) performance on *predator prey* with 9 agents

Table 2 The mean and standard of returns

Task	RDHNet (Ours)	COMIX	COVDN	FACMAC	MADDPG	IDDPG
cn(3 agent)	**−36.50±0.80**	−49.48±3.19	−39.65±1.93	−172.66±36.52	−48.11±3.79	−62.20±3.70
cn(5 agent)	**−48.17±3.73**	−76.54±7.76	−69.69±3.84	−141.53±33.84	−80.95±3.75	−92.11±8365
cn(7 agent)	**−61.91±4.05**	−97.41±9.47	−88.91±2.36	−211.83±115.19	−97.43±4.28	−117.51±15.39
pp(3 predator)	115.41±17.23	74.64±56.78	**125.44±28.60**	22.26±23.88	20.63±39.21	7.01±8.5
pp(6 predator)	**280.80±16.35**	15.39±9.26	65.82±55.98	38.86±45.96	8.43±4.96	13.03±4.01
pp(9 predator)	**178.80±25.48**	34.20±16.13	30.00±8.57	24.20±4.87	19.20±3.97	50.60±13.95

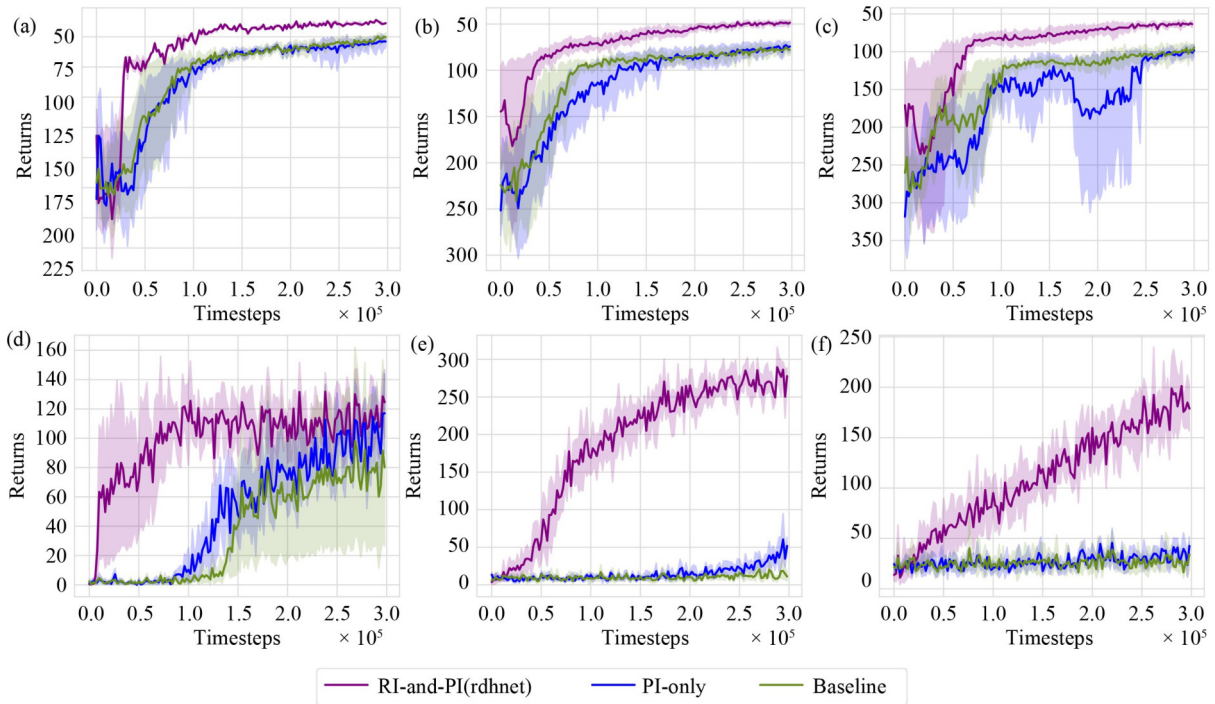

Fig. 6 Ablation studies. The baseline is COMIX, PI-only is COMIX with HPN, and PI-and-RI is RDHNet. (a) Performance on *Cooperative Navigation* with 3 agents; (b) performance on *Cooperative Navigation* with 5 agents; (c) performance on *Cooperative Navigation* with 7 agents; (d) performance on *predator prey* with 3 agents; (e) performance on *predator prey* with 6 agents; (f) performance on *predator prey* with 9 agents

the baseline improved performance across various Cooperative Prey Predator task scenarios compared to the original baseline. However, in the cooperative navigation task, performance did not significantly improve and even worsened as the number of entities increased. We speculate that the HPN network's use of PE reduced the policy's representational capacity. Nonetheless, the method incorporating both IE and PE achieved the best performance across all task scenarios, with its advantage becoming more pronounced as the number of entities increased. These experiments confirm that symmetry indeed exists in MARL problems and has a tangible impact on MARL performance.

5.4 Influence with different number of referred agents

In this section, we focus on studying the impact of selecting different numbers of reference entities on the performance and computational cost of RDHNet. We conduct an extensive analysis using the *CN 3a* cn 3a task, where we vary the number of reference entities k_r used to construct the polar coordinate system, in order to investigate how k_r affects the performance of RDHNet. The experimental results are shown in Fig. 7.

From the experimental results, it appears that fewer reference entities yield better performance, particularly during the early stages of training. A possible explanation for this phenomenon is that when more entities are selected, the optimization process during backpropagation must satisfy additional constraints, which makes the model's learning process more challenging. We plan to conduct further in-depth exploration and research on this phenomenon in the future. However, we have validated the feasibility of limiting RDHNet's computational complexity to $O(n^2)$.

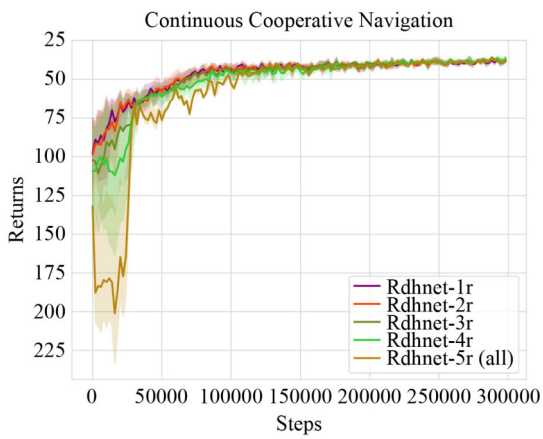

Fig. 7 The performance of RDHNet with different referred numbers of entities

5.5 Latent embedding visualization

In this section, we use the data distribution in the latent space to demonstrate RDHNet's capability to compress the representation space effectively. Specifically, we first interact with the environment using a random policy to collect a dataset $\mathcal{D}$. Next, we apply rotation to the data, rotating each sample by arbitrary angles (in practice, each data point generates ten rotated versions). After constructing the dataset $\mathcal{D}$, each (o,a) pair in the dataset is input into different models. We then extract the features from the layer preceding the output layer as the data representation in the latent space described by the model.

These representations are clustered using t-SNE, with the results shown in Fig. 8. The figure shows that the representations of the baseline (COMIX) are the most

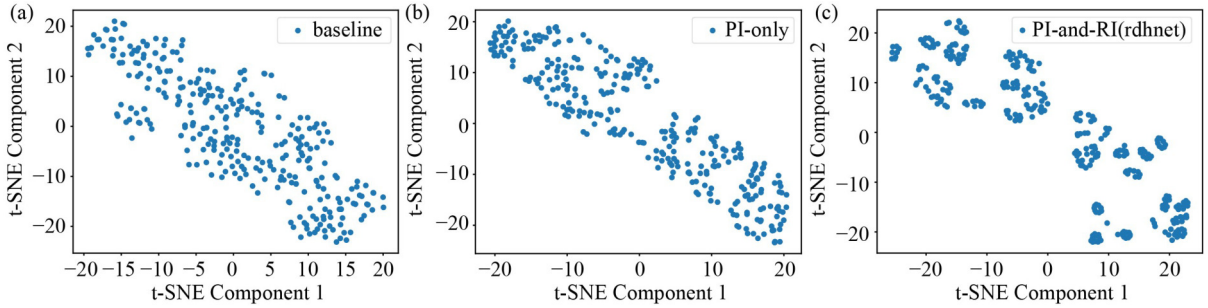

Fig. 8 The t-SNE visualization of features with different methods in latent space. (a) The results of COMIX' features; (b) the results of HPN' features; (c) the results of RDHNet' features

dispersed in the latent space, followed by the baseline augmented with Permutation Invariance (PI). The most compact clustering is produced by the model that incorporates both Permutation Invariance and Rotational Invariance (RI), which is RDHNet. In our experimental setup, the degree of clustering reflects the compactness of the representation space. Thus, we have sufficient reason to believe that integrating both PI and RI in our RDHNet model enables the effective compression of redundant representations, as demonstrated in Fig. 8.

6 Conclusion and discussion

In this paper, we first thoroughly analyze the representation redundancy problem caused by symmetry in multi-agent scenarios, distinguishing between permutation symmetry and rotational symmetry. We then propose the RDHNet architecture, inspired by human observation and problem-solving approaches, to address the issue of rotational symmetry in multi-agent systems. Finally, we validate our approach through experiments that reveal the objective hindrance posed by rotational symmetry to multi-agent learning and demonstrate that RDHNet applies to a wide range of multi-agent scenarios, further supporting our hypothesis: in specific scenarios and environments, Cartesian coordinates are unnecessary, as human intuition perceives distance and angles rather than constructing a Cartesian coordinate system in the mind.

We also acknowledge certain limitations of this work. For instance, the experimental scenarios lack diversity, the algorithm's computational complexity increases quadratically with the number of entities in the system, and we have yet to develop more effective optimization strategies for the actor-critic version of RDHNet. In future research, we also envision a more aggressive computational compression scheme. Specifically, under the POMDP setting, each entity would be able to observe only the k_n nearest entities to itself. As a result, when the number of agents increases, the model's computational complexity will be controlled at approximately $O(n)$ level. This would allow RDHNet to scale easily to large-scale scenarios. Of course, this remains part of our future work.

Acknowledgements The work was supported by the National Natural Science Foundation of China (Grant Nos. 91948303–1, 62106278), the National Key R and D Program of China (2021ZD0140301).

Competing interests The authors declare that they have no competing interests or financial conflicts to disclose.

Appendixes

Appendix A: Analysis of computational complexity

We evaluate the computational complexity of our proposed algorithms, focusing exclusively on the local networks, as the critic and mixer networks exhibit negligible differences across various algorithms.

For **RDHNet**, consider a scenario with n agents and m entities (where agents are a subset of entities). For each agent, the computational cost of selecting a reference entity and establishing a coordinate system per decision step is denoted by C. Consequently, the total cost for all n agents to establish polar coordinate systems and perform computations for each entity is given by

$$B_{\text{rpi}} = n(m-1)C.$$

We further decompose C by assuming that the computational cost of each hypernetwork for calculating an entity's attributes is C_h. Therefore,

$$C = (m-1)^2 C_h.$$

Substituting this into the expression for B_{rpi}, the total computational cost of RDHNet without optimization becomes

$$B_{\text{rpi}} = n(m-1)^2 C_h.$$

Given that $m > n$, we set $m = n + m_0$. Substituting this into the equation above yields:

$$B_{\text{rpi}} = n(n+m_0-1)^2 C_h = \left(n^3 + 2m_0 n^2 + (m_0-1)^2 n\right) C_h.$$

Since C_h is a constant, the computational complexity of RDHNet is

$$O(n^3).$$

For the **optimized RDHNet-LS**, where the number of reference entities is m_c and in scenarios where $n \gg m_c > 1$, the computational cost is

$$B_{\text{rpi}} = nmm_c C_h = n(n+m_0)m_c C_h = \left(n^2 + m_c m_0 n\right) C_h.$$

Thus, the computational complexity of RDHNet-LS is

$$O(n^2).$$

In contrast, for algorithms that do not consider symmetries, such as **COMIX**, **COVDN**, and **MADDPG**, assuming their local networks are standard MLPs with a computational cost

of C_m, the total cost for n agents is

$$B_{\text{ni}} = nC_m.$$

(Although the MLP size may increase with n, we approximate it as constant.) Therefore, their computational complexity is approximately

$$O(n).$$

For models that account for permutation invariance, such as **HPN**, assuming each hypernetwork has a computational cost C_h, the computational cost per agent is mC_h, and the total cost for n agents is

$$B_{\text{pi}} = nmC_h = n(n+m_0)C_h = \left(n^2 + nm_0\right)C_h.$$

Thus, their computational complexity is

$$O(n^2).$$

In summary, considering computational complexity with respect to the number of agents n, the order from highest to lowest is as follows:

1. **RDHNet**: $O(n^3)$
2. **Permutation-Invariant Models** (e.g., HPN) and **RDHNet-LS**: $O(n^2)$
3. **Non-Symmetry-Aware Models** (e.g., COMIX, COVDN): Approximately $O(n)$

Appendix B: Partial observation setting

The main body of the paper focuses on discussing and validating the rationale and effectiveness of RDHNet, while other issues in MARL were deliberately simplified or omitted. In the tasks discussed in the paper, each agent has a global view, which does not typically align with real-world scenarios. However, from a model design perspective, RDHNet should be applicable to POMDP scenarios.

Consider an extreme case where agent i cannot observe any entities except itself at a given moment. In this situation, no polar coordinate system can be constructed, and we allow the algorithm to take random actions (since there is no input information, the agent's decision can only be random). In all other cases, where the agent can observe at least one entity besides itself, RDHNet can construct at least one polar coordinate system. For entities that are not observed, their features are simply not computed. In such cases, RDHNet operates as intended. We believe that, compared to other methods, this would not significantly affect the performance of RDHNet.

We conducted experiments to evaluate this, and the results are as follows:

From Fig. A1, we can observe that, compared to global observation, RDHNet can still train successfully under the POMDP setting, although the training exhibits larger fluctuations. However, this does not significantly impact the overall performance. This phenomenon can be easily explained: under the POMDP setting, an agent's decision-making sometimes faces incomplete information, which naturally introduces a certain degree of instability in the training process. Nevertheless, overall, RDHNet is not heavily affected, which strongly demonstrates that RDHNet can be effectively applied in POMDP settings.

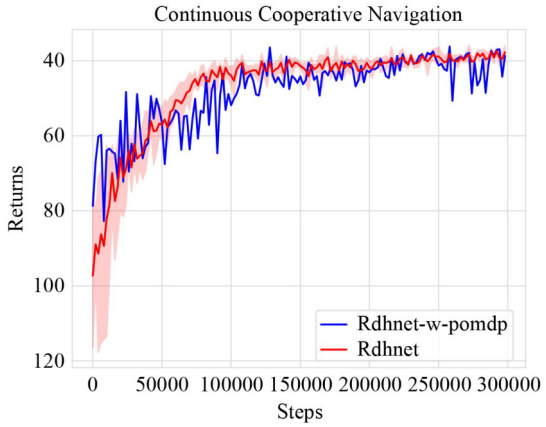

Fig. A1 The performance of RDHNet with POMDP setting

Appendix C: The detail of directional information proccess

Assume that at a certain moment, the observation of agent i is

$$\begin{aligned} o_i &= [E_1, \ldots, E_n] \\ &= [(p_1, z_1), \ldots, (p_n, z_n)] \\ &= [((x_1, y_1), z_1), \ldots, ((x_n, y_n), z_n)], \end{aligned}$$

where x_k, y_k denotes the Cartesian coordinates corresponding to entity k. Next, we can derive the formula for calculating the angle information as follows:

$$\sin(\theta_{ij}) = \frac{y_j}{\sqrt{x_j^2 + y_j^2}}, \quad \cos(\theta_{ij}) = \frac{x_j}{\sqrt{x_j^2 + y_j^2}},$$

$$\sin(\theta_{ik}) = \frac{y_k}{\sqrt{x_k^2 + y_k^2}}, \quad \cos(\theta_{ik}) = \frac{x_k}{\sqrt{x_k^2 + y_k^2}},$$

$$\sin(\theta_{ijk}) = \sin(\theta_{ik} - \theta_{ij}) = \sin(\theta_{ik}) \cdot \cos(\theta_{ij}) - \cos(\theta_{ik}) \cdot \sin(\theta_{ij}),$$

$$\cos(\theta_{ijk}) = \cos(\theta_{ik} - \theta_{ij}) = \cos(\theta_{ik}) \cdot \cos(\theta_{ij}) + \sin(\theta_{ik}) \cdot \sin(\theta_{ij}),$$

$$\begin{aligned} m_{ijk} &= \left[\sin(\theta_{ijk}), \cos(\theta_{ijk})\right] \\ &= \left[\frac{y_k x_j}{\sqrt{(y_k^2 + y_k^2)(x_j^2 + y_j^2)}} - \frac{x_k y_j}{\sqrt{(x_k^2 + y_k^2)(y_j^2 + y_j^2)}}, \right. \\ &\quad \left. \frac{x_k x_j}{\sqrt{(x_k^2 + y_k^2)(x_j^2 + y_j^2)}} + \frac{y_j y_k}{\sqrt{(y_k^2 + y_k^2)(y_j^2 + y_j^2)}} \right]. \end{aligned}$$

In the above derivation, θ_{ij} represents the angle between the line connecting entity i and entity j and the x-axis of the Cartesian coordinate system. Meanwhile, θ_{ijk} denotes the angle formed by entities i, j, and k, where i is the vertex of the angle. Other notations follow a similar representation.

References

1. Wen M, Lin R, Wang H, Yang Y, Wen Y, Mai L, Wang J, Zhang H, Zhang W. Large sequence models for sequential decision-making: a survey. Frontiers of Computer Science, 2023, 17(6): 176349
2. Mushtaq A, Haq I U, Sarwar M A, Khan A, Khalil W, Mughal M A. Multi-agent reinforcement learning for traffic flow management of

autonomous vehicles. Sensors, 2023, 23(5): 2373

3. Ma C, Li A, Du Y, Dong H, Yang Y. Efficient and scalable reinforcement learning for large-scale network control. Nature Machine Intelligence, 2024, 6(9): 1006−1020
4. Keren S, Essayeh C, Albrecht S V, Morstyn T. Multi-agent reinforcement learning for energy networks: computational challenges, progress and open problems. 2024, arXiv preprint arXiv: 2404.15583
5. Zhai Y, Ding B, Liu X, Jia H, Zhao Y, Luo J. Decentralized multi-robot collision avoidance in complex scenarios with selective communication. IEEE Robotics and Automation Letters, 2021, 6(4): 8379−8386
6. Li Y, Wang H, Ding B, Zhou W. RoboCloud: augmenting robotic visions for open environment modeling using internet knowledge. Science China Information Sciences, 2018, 61(5): 050102
7. OpenAI. Dota 2 with large scale deep reinforcement learning. 2019, arXiv preprint arXiv: 1912.06680
8. Wen M, Kuba J G, Lin R, Zhang W, Wen Y, Wang J, Yang Y. Multi-agent reinforcement learning is a sequence modeling problem. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022, 1201
9. Chen Y, Geng Y, Zhong F, Ji J, Jiang J, Lu Z, Dong H, Yang Y. Bi-DexHands: towards human-level bimanual dexterous manipulation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024, 46(5): 2804−2818
10. Yarahmadi H, Shiri M E, Navidi H, Sharifi A, Challenger M. Bankruptcy-evolutionary games based solution for the multi-agent credit assignment problem. Swarm and Evolutionary Computation, 2023, 77: 101229
11. Yarahmadi H, Shiri M E, Navidi H, Sharifi A, Challenger M. RevAP: a bankruptcy-based algorithm to solve the multi-agent credit assignment problem in task start threshold-based multi-agent systems. Robotics and Autonomous Systems, 2024, 174: 104631
12. Ying D, Zhang Y, Ding Y, Koppel A, Lavaei J. Scalable primal-dual actor-critic method for safe multi-agent RL with general utilities. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2024, 1586
13. Yuan L, Chen F, Zhang Z, Yu Y. Communication-robust multi-agent learning by adaptable auxiliary multi-agent adversary generation. Frontiers of Computer Science, 2024, 18(6): 186331
14. Zhang R, Xu Z, Ma C, Yu C, Tu W W, Huang S, Ye D, Ding W, Yang Y, Wang Y. A survey on self-play methods in reinforcement learning. 2024, arXiv preprint arXiv: 2408.01072
15. Wang W, Yang T, Liu Y, Hao J, Hao X, Hu Y, Chen Y, Fan C, Gao Y. Action semantics network: considering the effects of actions in multiagent systems. In: Proceedings of the 8th International Conference on Learning Representations. 2020
16. Hu S, Zhu F, Chang X, Liang X. UPDeT: universal multi-agent reinforcement learning via policy decoupling with transformers. 2021, arXiv preprint arXiv: 2101.08001
17. Hao J, Hao X, Mao H, Wang W, Yang Y, Li D, Zheng Y, Wang Z. Boosting multiagent reinforcement learning via permutation invariant and permutation equivariant networks. In: Proceedings of the 11th International Conference on Learning Representations. 2023
18. Zhou X, Gao H, Xu X, Zhang X, Jia H, Wang D. PCRL: priority convention reinforcement learning for microscopically sequencable multi-agent problems. In: Proceedings of Deep Reinforcement Learning Workshop NeurIPS 2022. 2022
19. van der Pol E, van Hoof H, Oliehoek F A, Welling M. Multi-agent MDP homomorphic networks. In: Proceedings of the 10th International Conference on Learning Representations. 2022
20. Yu X, Shi R, Feng P, Tian Y, Li S, Liao S, Wu W. Leveraging partial symmetry for multi-agent reinforcement learning. In: Proceedings of the 38th AAAI Conference on Artificial Intelligence. 2024, 17583−17590
21. Yu X, Shi R, Feng P, Tian Y, Luo J, Wu W. ESP: exploiting symmetry prior for multi-agent reinforcement learning. In: Proceedings of the 26th European Conference on Artificial Intelligence Including 12th Conference on Prestigious Applications of Intelligent Systems. 2023, 2946−2953
22. Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th International Conference on Learning Representations. 2017
23. Qi C R, Su H, Mo K, Guibas L J. PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, 77−85
24. Qi C R, Yi L, Su H, Guibas L J. PointNet++: deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 5105−5114
25. Thomas N, Smidt T, Kearnes S, Yang L, Li L, Kohlhoff K, Riley P. Tensor field networks: rotation- and translation-equivariant neural networks for 3D point clouds. 2018, arXiv preprint arXiv: 1802.08219
26. Fuchs F, Worrall D, Fischer V, Welling M. SE(3)-transformers: 3D roto-translation equivariant attention networks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 1970−1981
27. Maron H, Ben-Hamu H, Serviansky H, Lipman Y. Provably powerful graph networks. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019, 193
28. Gasteiger J, Groß J, Günnemann S. Directional message passing for molecular graphs. In: Proceedings of the 8th International Conference on Learning Representations. 2020
29. Rashid T, Samvelyan M, De Witt C S, Farquhar G, Foerster J, Whiteson S. Monotonic value function factorisation for deep multi-agent reinforcement learning. Journal of Machine Learning Research, 2020, 21(178): 1−51
30. Peng B, Rashid T, de Witt C A S, Kamienny P A, Torr P H S, Böhmer W, Whiteson S. FACMAC: factored multi-agent centralised policy gradients. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. 2021, 934
31. de Boer P T, Kroese D P, Mannor S, Rubinstein R Y. A tutorial on the cross-entropy method. Annals of Operations Research, 2005, 134: 19−67
32. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M. Playing Atari with deep reinforcement learning. 2013, arXiv preprint arXiv: 1312.5602
33. Lowe R, Wu Y, Tamar A, Harb J, Abbeel P, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 6382−6393
34. Lillicrap T P, Hunt J J, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. In: Proceedings of the 4th International Conference on Learning Representations. 2016
35. Sunehag P, Lever G, Gruslys A, Czarnecki W M, Zambaldi V, Jaderberg M, Lanctot M, Sonnerat N, Leibo J Z, Tuyls K, Graepel T. Value-decomposition networks for cooperative multi-agent learning based on team reward. In: Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems. 2018, 2085−2087

Dongzi WANG received MS degrees from National University of Defense Technology, China in 2020. He is currently studying as a doctoral student in Electronic Science and Technology at the National University of Defense Technology. His research interests include machine learning, reinforcement learning and multi-agent reinforcement learning. His research focuses on improving the learning efficiency of multi-agent systems by enhancing knowledge-sharing mechanisms.

Lilan HUANG received the BS in Atmospheric Science from Sun Yat-sen University, China in 2018. She graduated from National University of Defense Technology with a MS in Computer Science and Technology in 2021. She is currently studying as a doctoral student in Software Engineering at the National University of Defense Technology, China. Since 2018, she has been working on data assimilation using machine learning. Using deep learning and reinforcement learning to enable data assimilation, improve the performance of hybrid data assimilation methods, and promote numerical weather forecasting.

Muning WEN is currently a third-year PhD student at Shanghai Jiao Tong University, China under the supervision of Professor Weinan Zhang. He has extensive theoretical and practical experience in multi-agent systems. His research interests mainly focus on reinforcement learning, multi-agent reinforcement learning, and reinforcement learning for LLM agents. Wen Muning has published over twenty papers in prestigious conferences such as NeurIPS, ICML, and ICLR, and has been actively serving as a reviewer for these conferences since 2023.

Yuanxi PENG received the BS degree in computer science from Sichuan University, China in 1988, and the MS and PhD degrees in computer science from the National University of Defense Technology (NUDT), China in 1998 and 2001, respectively. He has been a professor at the College of Computer Science and Technology, National University of Defense Technology (NUDT) since 2011. His research interests are in the areas of hyperspectral image processing, high-performance computing, multi and many-core architectures, on-chip networks, cache coherence protocols, and architectural support for parallel programming.

Minglong LI received the MS degree in computer science from the National University of Defense Technology (NUDT), China in 2017 and the PhD degree in software engineering from the National University of Defense Technology (NUDT), China in 2020, respectively. He is an assistant research fellow in College of Computer Science and Technology, National University of Defense Technology (NUDT). His research interests are in the areas of Artificial Intelligence (AI), Robotics, and Multi-agent Systems.

Teng LI received the BS, MS, and PhD degrees from National University of Defense Technology, China in 2013, 2015, and 2020, respectively. His research interests include machine learning, hyperspectral image processing and representation learning. He has also conducted research on the integration of multi-agent reinforcement learning and graph neural networks in recent years.