ADMN: Agent-Driven Modular Network for Dynamic Parameter Sharing in Cooperative Multi-Agent Reinforcement Learning

Yang Yu¹,², Qiyue Yin¹,², Junge Zhang¹,²*, Pei Xu², Kaiqi Huang¹,²,³*
¹School of Artificial Intelligence, University of Chinese Academy of Sciences
²CRISE, Institute of Automation, Chinese Academy of Sciences
³CAS, Center for Excellence in Brain Science and Intelligence Technology
{yuyang2019, pei.xu}@ia.ac.cn, {qyyin, jgzhang, kqhuang}@nlpr.ia.ac.cn

Abstract

Parameter sharing is a common strategy in multi-agent reinforcement learning (MARL) to make the training more efficient and scalable. However, applying parameter sharing among agents indiscriminately hinders the emergence of agents diversity and degrades the final cooperative performance. To better balance parameter sharing and agents diversity, we propose a novel Agent-Driven Modular Network (ADMN), where agents share a base network consisting of multiple specialized modules, and each agent has its own routing to connect these modules. In ADMN, modules are shared among agents to improve the training efficiency, while the combination of different modules brings rich diversity. The agent routing at different time steps is learned end-to-end to achieve a dynamic and adaptive balance. Specifically, we also propose an information-theoretical regularization between the routing of agents and their behavior to further guarantee the identifiability of different routing. We evaluated ADMN in challenging StarCraft micromanagement games and Google Research Football games, and results demonstrate the superior performance of ADMN, particularly in larger or heterogeneous cooperative tasks.

1 Introduction

Multi-agent reinforcement learning (MARL) has gained substantial attention in recent years, due to its applicability in addressing various real-world cooperative problems, including cooperative games [Berner et al., 2019; Bard et al., 2020], traffic control [Bazzan, 2009; Chu et al., 2019], and robot fleet coordination [Cao et al., 2012; Hüttenerauch et al., 2019]. Nevertheless, effectively learning coordinated policies for complex cooperative tasks remains challenging, as the algorithm must explore the exponentially growing joint action-state space with the increasing number of agents. To make the training more efficient and scalable, recent MARL methods [Peng et al., 2012; Wang et al., 2021a; Foerster et al., 2018; Yu et al., 2021] predominantly employ parameter sharing, where agents share their network parameters.

Parameter sharing indeed facilitates the training in MARL, significantly reducing the total number of trainable parameters and making the training complexity tractable. More importantly, this sharing allows for the reuse of experiences by multiple agents. Although these merits make parameter sharing popular in MARL, having all agents share the same parameters also has negative effects [Li et al., 2021; Hu et al., 2022]. In the real world, the completion of many complex cooperative tasks requires for agents with different abilities. However, fully-shared parameters may cause agents to behave similarly, hindering the emergence of agents diversity, and limiting agents exploration capabilities, thereby degrading the final cooperative performance.

To address the lack of diversity in naive parameter sharing, there are researches encouraging no parameter sharing or selective parameter sharing in MARL [Jiang and Lu, 2021; Li et al., 2021; Christianos et al., 2021; Kim and Sung, 2023]. Instead of sharing parameters across all agents, these methods train individual policy networks or group-based networks to encourage the diversity across agents or agent groups. However, existing methods employ fixed parameter sharing patterns for agents during cooperation, resulting in static diversity that fails to adapt to dynamic cooperative tasks.

*Corresponding authors.
In this paper, to deal with the dilemma between parameter sharing and agents diversity, and adapt to the dynamic environment or different cooperative tasks, we propose a novel framework called Agent-Driven Modular Network (ADMN). In ADMN, a modular network is introduced into the agent architecture, consisting of different specialized modules. During training, these modules are shared among agents to decrease the training complexity, while different routing, i.e., the connecting weights across different modules bring rich diversity. At each time step, the agent configures its routing according to its own observation and identity, which is learned end to end and achieves a dynamic and flexible balance, without relying on human prior knowledge. Specifically, to further guarantee the identification of different routing, an information-theoretical regularization between the routing of agents and their behavior is also proposed under ADMN.

We evaluated ADMN in StarCraft Multi-Agent Challenge (SMAC) [Samvelyan et al., 2019] and Google Research Football (GRF) [Kurach et al., 2020]. Compared to baselines, results demonstrate superior performance of our method in challenging cooperative tasks, especially in larger or heterogeneous cooperative tasks. And visualization of agents’ routing across various tasks or time steps underscores the flexibility and reasonability of learned routing policies in ADMN.

2 Related Work

To make the training in MARL efficient and scalable, many methods adopt naive parameter sharing framework, including value-based MARL methods [Rashid et al., 2020; Iqbal et al., 2021; Xu et al., 2023b] and policy-based methods [Lowe et al., 2017; Zhou et al., 2020; Zhang et al., 2021b; Xu et al., 2023a], where all agents share the same value network or policy network, as shown in Figure 2(a). However, naive parameter sharing hinders the diversity of agents and limits the emergence of sophisticated coordinated policies. To encourage the diversity across agents or agent groups, recent approaches introduce no parameter sharing or selective parameter sharing as shown in Figure 2(b,c). For example, EOI [Jiang and Lu, 2021] avoids parameter sharing to prevent similar agent behaviors, CDS [Li et al., 2021] fosters individual diversity with agent-specific heads in addition to a sharing network, and SePS [Christiansos et al., 2021] divides agents into groups with shared network parameters before training. Although these methods produce diverse agent policies, the introduction of agent-based or group-based networks decreases the method’s scalability, whose size of training parameters still increases with the number of agents. More importantly, the diversity introduced in existing works is static, which encourages fixed parameter sharing patterns during cooperation, and cannot adapt to dynamic cooperative tasks. In contrast, the size of ADMN will not increase with the number of agents. Furthermore, as shown in Figure 2(d), the diversity in ADMN comes from the combination of different modules, which is varying at different time steps and learned end to end to achieve a dynamic and adaptive parameter sharing.

Another approach to balance efficiency and diversity in MARL is task decomposition, where the whole task is divided into subtasks, and agents are assigned to different sub-groups for various subtasks [Wooldridge et al., 2000; DeLoach and Garcia-Ojeda, 2010; Lhaksmana et al., 2018]. For example, RODE [Wang et al., 2021b] proposes to decompose the whole task based on the decomposition of the joint-action space, where each subtask contains partial actions. However, this division is limited to specific environments, and does not apply to tasks where basic actions are needed in each subtask. More recently, LDSA [Yang et al., 2022] is proposed to learn the task decomposition and agents assignment, by learning the subtask encoder and the agent trajectory encoder. Although the task decomposition in LDSA is learnable, it still needs human prior knowledge to help design the algorithm, such as the decision of the number of subtasks. On the contrary, in ADMN, the efficiency and diversity are balanced using the modular network, where the sharing of the modular knowledge across agents is learned end to end, without introducing any task-related human prior knowledge.

3 Preliminary

3.1 Problem Formulation

Multi-agent reinforcement learning is the extension of reinforcement learning to help address multi-agent sequential decision problems, especially cooperation games,
which is usually modeled as Dec-POMDP, i.e., $G = \langle S, U, A, P, R, Z, O, n, \gamma \rangle$ [Zhang et al., 2021a]. $S$ is the state space of the environment. At each time step $t$, every agent $u \in U \equiv \{1, \ldots, n\}$ chooses an action $a_u \in A$ which forms the joint action $a = a^n$ $P$ is the state transition function which maps the current state $s$ and the joint action $a$ to the next state $s'$, i.e., $P(s'|s, a) : S \times A \times S \rightarrow [0, 1]$. All agents receive a shared reward $r \in \mathbb{R}$ according to the reward function $R(s,a) : S \times A \rightarrow \mathbb{R}$ and $\gamma \in [0,1)$ is the discount factor. In a partially observable setting, each agent does not have access to the full state and instead samples observations $o_u \in O$ according to observation function $Z(s,u) : S \times U \rightarrow O$. The action-observation history for an agent $u$ is $\tau_u \in T \equiv (O \times A)^*$, on which it conditions its policy $\pi_u(\tau_u) : T \times A \rightarrow [0, 1]$. The joint action-value function is defined as $Q^\pi(s, a) = E_{\tau \sim \pi} [\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0, a_0, \tau]$, where $\pi$ is the joint policy. The object of MARL is to find the joint policy $\pi$ to maximize the expected sum of the discounted team reward.

### 3.2 Centralized Training and Decentralized Execution

To address the non-stationary issue and the scalability problem in cooperative MARL [Tan, 1993; Matignon et al., 2012], existing cooperative MARL algorithms mainly adopt the centralized training and decentralized execution (CTDE) paradigm. CTDE means during training, the algorithm has access to the full state and the action-observation histories of all agents, while during execution each agent makes decisions conditioned on its own local history. In this paper, we also adopt the CTDE paradigm, and base our method on the popular value-based CTDE method QMIX [Rashid et al., 2018]. QMIX tries to factor the joint-action value $Q^{\text{tot}}$ into a monotonically nonlinear combination of individual utilities $Q_u$, where the individual utility is learned via a utility network. A mixer network with nonnegative weights is used for combining agents’ utilities. The nonnegativity in the mixer network ensures that $\frac{\partial Q^{\text{tot}}(s,a)}{\partial \pi_u(\tau_u, a_u)} \geq 0$, which guarantees Individual-Global-Max (IGM) Condition [Son et al., 2019], i.e. the optimal joint actions across agents are equivalent to the collection of individual optimal actions of each agent. QMIX is effective since the maximization can be performed in $O(n |A|)$ as opposed to $O(|A|^n)$, and during execution, each agent can independently make decisions by its own utility network without any global information since IGM condition is satisfied. More details will be provided in Appendix.

### 4 Method

In ADMN, there is a base modular network consisting of multiple modules that can be shared by agents, and a routing network used by each agent to generate its routing, i.e., connecting weights across modules at each time step. In particular, to enhance the identification of different routing, a mutual information regularization between the routing and the behavior of the agent is further proposed. In the following, we will present the network architecture of ADMN and elaborate on the mutual information regularization. The framework of ADMN is provided in Figure 3.

![Figure 3: ADMN consists of a base modular network shown at the left center, and a routing network at the right center with the input of the agent identity and observation. Different color slices are related to routing paths of different agents. The mutual information regularization is shown on the top right to strengthen the influence of different agents routing on the agent observation transitions.](image)

#### 4.1 Routing Network

As depicted in Figure 3, the model of ADMN consists of a base modular network, and a routing network, where the latter assists each agent to configure the module connections in adjacent layers. Specifically, we employ soft instead of hard modularization, where the routing network outputs probabilities to weight the modules instead of determining discrete routing paths, and is trained with the base network together. As discussed in [Rosenbaum et al., 2019; Yang et al., 2020], the joint optimization of the hard routing network and the base network is unstable, and hard modularization introduces higher variation along with the exploration in the environment.

To learn the routing of agents with desired properties, we encode the routing in a stochastic embedding space, and the routing of agent $u$, denoted as, $p_u \in R^{\text{dim}}$, is drawn from a multivariate Gaussian distribution, $N(\mu_{p_u}, \sigma_{p_u})$. At each time step $t$, the inputs of the routing network contain the agent identity $z_u$, which is a one-hot vector representing each agent, and the current observation of the agent $o_u^t$ to adapt the routing to the dynamic environment. First, the observation undergoes processing through a one-layer MLP and a GRU unit to obtain its representation, $f(\phi(z_u)) \in R^D$, and the identity is processed by a one-layer MLP and obtain $h(z_u) \in R^D$. Then $p^t_u$ is calculated based on the representation of its observation and identity. Assuming there are $L$ module layers with each layer $m$ modules, the output dimension of the routing network is $\text{dim} = (L - 1) \times m \times m + m$ (the last $m$ represents the weights of the modules in the last module layer). The output of the routing network for agent $u$ is:

$$
(\mu_{p_u}, \sigma_{p_u}) = G(ReLU(f(\phi(z_u)) \cdot h(z_u))),
$$

$$
p^t_u \sim N(\mu_{p_u}, \sigma_{p_u}),
$$

(1)
where $G$ is the parameters of fully-connected layers. Then the routing vector $p_{l}^{t}$ is subsequently split into the connection weights of different modules in adjacent layers. Additionally, a softmax function is applied to normalize the weights pointed to the same module, that is:

$$p_{l+1}^{t;j,u} = \frac{\exp(p_{l+1}^{t;j,u})}{\sum_{j=1}^{m} \exp(p_{l+1}^{t;j,u})},$$

where $p_{l+1}^{t;j,u}$ is the probability weighting the contribution of the $j$th module in the $l$ layer to the $i$th module in the $l+1$ layer, utilized by agent $u$ at time step $t$. With Eq. 1 and Eq. 2, we can obtain weights connecting modules in adjacent layers for each agent at each time step.

### 4.2 Base Modular Network

As mentioned above, the base modular network contains $L$ layers, each with $m$ modules, and connecting weights of modules are generated by the routing network. We denote the input for the $j$th module in the $l$ layer as a $d$-dimensional feature $g_{j;u}^{l} \in \mathbb{R}^{d}$, then the input feature for the $i$th module in the $l+1$ layer can be expressed as:

$$g_{i;u}^{l+1} = \sum_{j=1}^{m} p_{i;j,u}^{l} \left( \text{ReLU}(W_{j}^{l}g_{j;u}^{l}) \right),$$

where $W_{j}^{l}$ represents the parameters of the $j$th module in the $l$ layer. And the output of local utility function of agent $u$ is:

$$Q_{u} = F \left( \sum_{j=1}^{m} p_{i;j,u}^{l} \left( \text{ReLU}(W_{j}^{l+1}g_{j;u}^{l+1}) \right) \right),$$

where $F$ is the parameter of the last layer MLP in the local utility function network, and $p_{i;j,u}^{l}$ represents the weights of module outputs in the $L$ layer to the last layer MLP in the local utility function network. With Eq. 3 and Eq. 4, we can calculate the weighted sum of different module outputs by the probabilities produced by the routing network. Consequently, driven by the observation and the identity of each agent, the inputs of different agents will flow into similar or different forward paths, yielding corresponding outputs, i.e., their action-value utilities $Q_{u}$.

### 4.3 Mutual Information Regularization

In ADMN, we want agents with different routing paths to generate diverse behavior and prevent different routing from collapsing into similar behavior. To further strengthen the influence of different routing, we introduce an auxiliary objective to maximize the mutual information between the routing $p_{l}^{t}$ and the agent’s behavior.

As discussed in [Eysenbach et al., 2018], different actions may yield similar effects on the environment and are indistinguishable, like different force values applied by a robotic arm to grasp a cup which does not move. In this paper, to enable the identification of routing based on the agent’s behavior, and make the influence of the routing visible, we associate the agent routing with observable agent observation transitions rather than agent actions. Concretely, we introduce a conditional mutual information restriction between the agent routing and its next observation conditioned on the current observation of the agent, which is:

$$I(p_{u}^{t}; o_{u}^{t+1} | o_{u}^{t}) = H(p_{u}^{t} | o_{u}^{t}) - H(p_{u}^{t} | o_{u}^{t+1}, o_{u}^{t})$$

$$= H(o_{u}^{t+1} | o_{u}^{t}) - H(o_{u}^{t+1} | p_{u}^{t}, o_{u}^{t})$$

$$= E_D \left[ \log p(o_{u}^{t+1} | p_{u}^{t}, o_{u}^{t}) - \log p(o_{u}^{t+1} | o_{u}^{t}) \right],$$

where $D$ is the replay buffer.

Since the probabilities $p(o_{u}^{t+1} | p_{u}^{t}, o_{u}^{t})$ and $p(o_{u}^{t+1} | o_{u}^{t})$ are unknown, we introduce the variational distribution $q_{\xi}(o_{u}^{t+1} | p_{u}^{t}, o_{u}^{t})$ and $q_{\zeta}(o_{u}^{t+1} | o_{u}^{t})$ parameterized by $\xi$ and $\zeta$ as a proxy, obtaining an approximation of the optimization objective:

$$I(p_{u}^{t}; o_{u}^{t+1} | o_{u}^{t}) \approx E_D \left[ \log q_{\xi}(o_{u}^{t+1} | p_{u}^{t}, o_{u}^{t}) - \log q_{\zeta}(o_{u}^{t+1} | o_{u}^{t}) \right].$$

To affect the training of all parameters in the agent network, we derive an auxiliary reward to optimize the above objective:

$$r_{t}^{i} = E_u \left[ \log q_{\xi}(o_{u}^{t+1} | p_{u}^{t}, o_{u}^{t}) - \log q_{\zeta}(o_{u}^{t+1} | o_{u}^{t}) \right],$$

which will be combined with the extrinsic reward to guide agents training. More details are discussed in Appendix.

### 4.4 Overall Learning Objective

In this section, we introduce the overall learning objective of ADMN. Specifically, we add the auxiliary reward discussed in Eq. 7 to the extrinsic environmental rewards $r$, and utilize the following temporal-difference (TD) loss [Sutton and Barto, 2018] to optimize the model parameters in ADMN:

$$L_{TD} = (y_{t}^{tot} - Q_{tot}(\tau_{t}, a_{t}))^{2},$$

where $Q_{tot}$ is the output of the mixing network with the input of the utility of each agent $Q_{u}$, calculated in Eq. 4. And $y_{t}^{tot} = \hat{r}_{t} + \gamma Q_{tot}(\tau_{t+1}, \arg \max_{a_{t+1}} Q_{tot}(\tau_{t+1}, a_{t+1}))$, where $\hat{r}_{t}$ is the combination reward of the auxiliary reward $r_{t}^{i}$ and the environment reward $r_{t}$ with combination weight $\beta$, and $Q_{tot}$ is the target network periodically copied from $Q_{tot}$ for stable training as proposed in [Mnih et al., 2015]. The action input to the target network is chosen by $Q_{tot}$ as advised by [Van Hasselt et al., 2016].

## 5 Experiments

We conduct experiments to answer the following questions:

1. Is ADMN more effective in balancing efficiency and diversity in complex cooperative tasks?
2. How does each portion contribute to ADMN, such as the design of the modular network, and the information-theoretical regularization?
3. Is the routing learned by each agent dynamic and reasonable?
4. What is the influence of different routing on the agent behavior?

In the following, we will first introduce the environment used for evaluating our method and the baseline methods for comparison, then answer the questions above one by one.

### 5.1 Experimental Setup

We evaluate ADMN and baselines in the challenging StarCraft II micromanagment tasks (SMAC) and the Google
Research Football (GRF). The task in SMAC is a two-team battle game, where the algorithm needs to guide controlled alley agents to eliminate all enemy units in the other team. GRF provides an environment where agents are trained to play football in a physics-based 3D simulation. We evaluated our method on 3 challenging GRF offensive scenarios, academy_3_vs_1_with_keeper, academy_counterattack_easy, and academy_counterattack_hard, where agents need to coordinate to organize the offense in different initial positions. Different from the dense rewards setting in SMAC, agents in GRF will only be rewarded when scoring goals.

To evaluate the effectiveness of ADMN, firstly, we compare it with current state-of-the-art methods that focus on the parameter sharing issue. We provide the results of QMIX [Rashid et al., 2018], which adopts naive parameter sharing mechanism, CDS [Li et al., 2021], which incorporates agent diversity with individual network heads, and SePS, which utilizes selective parameter sharing by grouping agents before cooperative learning. Specially, we migrate SePS from the policy-based algorithm to the value-based algorithm named SePS-V for a fair comparison, and provide the comparison among policy-based algorithms in Appendix. Additionally, we also provide the comparison with methods related to task decomposition. Specifically, we compare with RODE [Wang et al., 2021b], which decomposes the task according to the division of the action space, and LDSA [Yang et al., 2022], which first learns several sub-task representations, then assigns agents to the subtask according to learned agent representation. Results of the baselines we compare with are obtained using the publicly available code released by their authors. Referring to existing MARL methods [Samvelyan et al., 2019; Wang et al., 2021b; Li et al., 2021], each algorithm is evaluated using 5 independent training runs with different random seeds, and the resulting plots include the median performance shown in dark color as well as the 25%-75% percentiles shown in the shaded area.

We conduct the experiment of ADMN with the setting that the number of module layers $L$ is 2, the number of modules in each layer $m$ is 2, and each module outputs a $d = 32$ representation. And $β$ weighing the environment rewards and the auxiliary rewards is 0.05. More details about the environment and training are provided in Appendix.

5.2 Performance on SMAC

To evaluate the balance of ADMN between efficiency and diversity, we evaluate ADMN and baselines in homogeneous SMAC maps with varying numbers of agents, as well as heterogeneous maps with different numbers of unit types. In homogeneous tasks, necessary parameter sharing is crucial for training efficiency, while in heterogeneous tasks, necessary diversity is essential to achieve sophisticated coordination. Results are shown in Figure 4, where in larger homogeneous or more heterogeneous cooperative tasks, such as 27m_vs_30m and 1c3s8z_vs_1c3s9z, ADMN outperforms both naive parameter sharing methods like QMIX, and diversity-based methods like CDS or SePS-V. This comparison highlights that, a dynamic and adaptive balance is effective both for tasks that demand training efficiency and for tasks that require diverse agent capabilities. Besides, we found that CDS exhibits better performance in small-scale homogeneous cooperative tasks but falls short in larger tasks. This result justifies that such fixed diversity pattern with individual networks may sacrifice the efficiency of parameter sharing, and is inefficient when scaled to tasks with a large number of agents. Results on other hard SMAC tasks are provided in Appendix, where ADMN is still comparable or even better than baselines.
5.3 Performance on GRF

We further compare ADMN with baselines in challenging football games GRF, and results are presented in Figure 5. In Figure 5, ADMN consistently outperforms baselines in these three scenarios. Considering that tasks in GRF are in a sparse-reward setting and ask for more sophisticated coordination, ADMN’s superior performance underscores its effectiveness in striking a more efficient balance, enabling agents to discover advanced coordinated policies. Besides, we notice that task decomposition methods like RODE, which performs relatively better in SMAC tasks, do not fare as well in GRF tasks. We posit that this phenomenon arises because RODE decomposes the entire task based on the decomposition of the action space, which is beneficial for SMAC tasks but does not apply to others. On the contrary, without introducing any task-related prior knowledge, ADMN finds an adaptive trade-off between the efficiency and diversity in multiple tasks in an end-to-end manner.

5.4 Ablation Studies

To investigate the significance of different portions in ADMN, we conduct ablation studies. First, considering that the introduction of the modular network enlarges the size of the agent network, we present the results of QMIX_Large, i.e., QMIX with similar size of the agent network in ADMN. Then for the routing network, we ablate the mutual information regularization in ADMN (ADMN_No_MI). Additionally, we ablate the input of the routing network, namely the agent observation and identity and obtain ADMN_no_obs and ADMN_no_id. Figure 6(a) illustrates the comparison of different ablations in SMAC task 7sz. Results indicate that the performance gains from increasing the network parameters are limited. Besides, the mutual information regularization indeed impacts the learning efficiency. Furthermore, note that both ADMN_no_obs and ADMN_no_id perform worse than ADMN, highlighting the importance of both identity and observation for the routing network in this SMAC task.

Besides, we conduct ablation studies on the base modular network design in ADMN. ADMN utilizes a modular network with 2 layers, and each layer has 2 modules with 32 neural units. We varied the network depth by replacing the number of layers with 1 (ADMN_Layer_1) and 4 (ADMN_Layer_4). Additionally, we explored configurations with 4 modules in each layer, featuring either 16 units (ADMN_Module_4, maintaining the same unit count as ADMN) or 32 units (ADMN_Wider). Figure 6(b) reveals that all ablations outperform the baseline QMIX. However, unlike modular networks with shallow layers, the deeper modular network (ADMN_Layer_4) is a little hard to train. Furthermore, ADMN_Module_4, the algorithm with a smaller number of unit sizes, is inferior to ADMN_Wider and ADMN, illustrating that the unit size in each module is crucial for ensuring adequate module specialization.

We extend above ablation studies to the GRF task academy_counterattack_easy in Figure 6(c, d). Similarly, results show that naively increasing the network parameters can not improve the performance effectively, and mutual information regularization is important. Notably, in the routing network, the absence of agent observation has a more adverse impact on performance than the absence of agent identity. This underscores the heightened importance of observation input for adaptive coordination in such dynamic tasks. In the design of the base modular network in the GRF environment, considering tasks in GRF are based on physics engine
and own more complex observation and action space, ADMN is relatively sensitive. While most ablations improve performance compared to QMIX, increased oscillation is observed, and a deeper modular network contributes minimally. More ablation studies are provided in Appendix.

5.5 Visualizaton of Routing

We analyze agents routing and visualize the probability $p_a$ of each agent via t-SNE [Van der Maaten and Hinton, 2008]. Results are presented in Figure 7. The first column depicts initial positions and learned routing embedding of agents in task 7sz. The last three columns show the states and t-SNE of routing samples about agents in 10m_vs_11m at different time steps. Notably, colored circles in snapshots represent agents controlled by ADMN, green bars depict the health of controlled agents at different time steps, and black circles represent the enemies controlled by the environment.

In task 7sz, circles in red and blue represent two unit types (zealot and stalker) separately. Results demonstrate that ADMN enables agents to learn distinct routing policies based on their unit type. In homogeneous task 10m_vs_11m, when $t$ is 0, our controlled agents gather together, and ADMN learns 10 separate clusters one for each agent. When $t$ is 5, agents adopt a loose formation to gain a larger attack range, with routing policies of agents in close positions visually converging. When $t$ is 28, with the battle goes on, agents vary in health and positions. Notably, routing of died agents is mixed. For surviving agents, routing embedding of healthier agents, like agents 4 and 6, is close yet distinct from agents with closer positions but lower health values. This observation indicates that ADMN learns to categorize agents based on both position and health, facilitating adaptable parameter sharing. The visualization of agents’ routing at different time steps or tasks justifies that ADMN adeptly generates dynamic and reasonable routing for agents based on their conditions.

In addition to visualizing agents’ routing, we aim to further investigate the impact of different routing strategies on agents. Considering that ADMN employs a soft routing paradigm with an infinite number of possible routing, we sample a subset and analyze their influence. Figure 8 illustrates examples of routing focusing more on the left branch (blue), the right branch (orange) and both branches (green). Note that the mutual information regularization in Eq. 5 encourages the relation of the routing and the observation transition of agents, hence we present the t-SNE results of observation changes in consecutive time steps of agents in the bottom left in Figure 8. The visualization reveals that distinct routing results in different observation transition distributions.

To delve deeper into the detailed influence of different routing, we plot the average change in observation related to these three routing on the right of Figure 8. Results indicate that the routing focusing on both branches has no impact on the agent observation, which might be adopted by dead agents. Besides, left-branch-focused routing reduces the distance between the agent and its allies, and increases the observed health of enemies. Given the partial observability of agents, the rise in enemies’ health signals increased chances of agent-enemy encounters, where enemies become visible with positive health values instead of zero. And this routing may lead to behaviors resembling support for attacked allies. Conversely, right-branch-focused routing enlarges the distance to both allies and enemies, considering the changes in enemy health are nearly opposite to the left branch, suggesting behaviors akin to recuperation away from the battlefield. The visualization in Figure 8 illustrates the nuanced influence of different routing on agents, contributing to the enrichment of behavioral diversity.

6 Conclusion

In this paper, we introduce ADMN, a novel framework incorporating an agent-driven modular network to address the challenge of balancing training efficiency and agent diversity in multi-agent cooperation. ADMN enables agents to dynamically configure similar or distinct module routings in an end-to-end manner, offering adaptability to diverse cooperative tasks and dynamic environments. And the mutual information regularization within ADMN enhances the identification of different routing strategies. While ADMN demonstrates superior performance in challenging cooperative tasks, the interpretability of the learned modules remains an area for improvement. Future work will focus on investigating the specific roles and analyzing the importance of different modules in ADMN for enhanced practical applications.
Acknowledgments
This work is supported in part by the National Science and Technology Major Project (Grant No.2022ZD0116403), the National Natural Science Foundation of China (Grant No.62176255), the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No.XDA27010103), the Youth Innovation Promotion Association CAS, and the Postdoctoral Fellowship Program of CPSF (Grant No.GZC20232995).

References


