AIIR-MIX: Multi-Agent Reinforcement Learning Meets Attention Individual Intrinsic Reward Mixing Network

Wei Li$^{(\boxtimes)}$ LI-WEI@SEU.EDU.CN
Weiyan Liu   LIUWEIYAN@SEU.EDU.CN
Shitong Shao   SHAOSHITONG@SEU.EDU.CN
Shiyi Huang   HUANGSHIYI@SEU.EDU.CN
School of Instrument Science and Engieering, Southeast University, Nanjing, Jiangsu 210096, China

Editors: Emtiyaz Khan and Mehmet Gönen

Abstract

Deducing the contribution of each agent and assigning the corresponding reward to them is a crucial problem in cooperative Multi-Agent Reinforcement Learning (MARL). Previous studies try to resolve the issue through designing an intrinsic reward function, but the intrinsic reward is simply combined with the environment reward by summation in these studies, which makes the performance of their MARL framework unsatisfactory. We propose a novel method named Attention Individual Intrinsic Reward Mixing Network (AIIR-MIX) in MARL, and the contributions of AIIR-MIX are listed as follows: **(a)** we construct a novel intrinsic reward network based on the attention mechanism to make teamwork more effective. **(b)** we propose a Mixing network that is able to combine intrinsic and extrinsic rewards non-linearly and dynamically in response to changing conditions of the environment. We compare AIIR-MIX with many State-Of-The-Art (SOTA) MARL methods on battle games in StarCraft II. And the results demonstrate that AIIR-MIX performs admirably and can defeat the current advanced methods on average test win rate. To validate the effectiveness of AIIR-MIX, we conduct additional ablation studies. The results show that AIIR-MIX can dynamically assign each agent a real-time intrinsic reward in accordance with their actual contribution.

Keywords: Multi-Agent Reinforcement Learning, Attention Mechanism, Mixing Network, Intrinsic Reward.

1. Introduction

Deep Reinforcement Learning (DRL) is an crucial branch of machine learning. It utilizes neural networks to approximate the optimal action decision or value function of the agent, realizing the generalization of the representation ability. By reason of the powerful fitting ability of deep learning, DRL can be employed as an effective way to tackle agent decision-making problems in complicated environments, such as group decision-making Nguyen et al. (2020), speech recognition Mousavi et al. (2016); Shen et al. (2019), autonomous driving Shen et al. (2019), natural language processing Young et al. (2018), and intelligent control Carlucho et al. (2020).

A multi-agent environment, where multiple agents are present for interaction and learning, is an emerging hot topic in recent years. Unfortunately, single-agent reinforcement learning is not very effective when used in a multi-agent environment, because the joint

Li$^{(\boxtimes)}$ Liu Shao Huang

action space of the agent resulting from fully centralized learning is too large to learn the optimal policy. Therefore, Multi-Agent Reinforcement Learning (MARL) is an extension of deep reinforcement learning from single-agent to multi-agent, and has a list of methods to solve this problem. Centralized Training and Decentralized Execution (CTDE) Foerster et al. (2016); Wang* et al. (2020), an effective and widely applied method in this list. In this paradigm, the central controller manages all the agents' observations, actions, and rewards during training. And the central controller and its value networks are not utilized during execution. Due to the above features and superiority, CTDE has become a widely applied paradigm in MARL, such as COMA Foerster et al. (2018), VDN Sunehag et al. (2017), QMIX Rashid et al. (2018) and QTRAN Son et al. (2019). In our research, we will also follow the CTDE paradigm in our proposed method.

In MARL, the reward function is extremely significant. However, it is difficult to go about formalizing all situations as reward functions in some real-world tasks. For exploring new environments in reinforcement learning, often only sparse rewards can be set. And yet, in practical application scenarios, sparse rewards still face problems such as inefficient samples and difficulty in exploration. This problem is more evident in MARL. Because of the inherent challenges of MARL, such as unstable environment and dimensional catastrophe, extending MARL to sparse reward settings will further increase the difficulty of policy learning. A good method for solving the incentive issue in a multi-agent environment is to assign extrinsic rewards and construct an intrinsic reward for each agent. In recent years, many researchers have started to focus on these directions. LIIR Du et al. (2019) learns an intrinsic reward function for each agent and continuously updates it to maximize the expected accumulated team reward from the environment. GIIR Wu et al. (2021) solves the lazy agent problem by using an intrinsic reward encoder to generate a separate intrinsic reward for each agent. OpenAI Berner et al. (2019) uses artificially set intermediate rewards to accelerate learning. FTW Jaderberg et al. (2019) learns the agent's intrinsic reward through two layers of optimization. However, both approaches overlook the following points: **(a)** building dependencies (i.e., attention mechanisms) between agents can induce more precise rewards; **(b)** integrating intrinsic and extrinsic rewards to facilitate policy learning better.

In this paper, we propose Attention Individual Intrinsic Reward Mixing Network (AIIR-MIX) method to fill this gap. AIIR-MIX includes the generation of precise intrinsic reward network (AIIR) and a non-linear Mixing network (MIX) to combine the intrinsic and extrinsic rewards. In terms of generating intrinsic reward, we propose an intrinsic reward network based on the attention mechanism. We assume that each agent has a separate intrinsic reward. The agents' observations and actions are extremely similar when they perform teamwork. Based on the above, the contribution of each agent in teamwork is calculated from each agent's observation and action using the attention mechanism, and a more accurate intrinsic reward is generated for each agent. The intrinsic reward network is updated to maximize the standard cumulative discounted extrinsic rewards from the environment. In terms of combining intrinsic and extrinsic rewards, we propose a Mixing network that allows intrinsic and extrinsic rewards to be combined in a non-linear manner. The extrinsic reward is fed to the hyper network to generate the weights of the Mixing network. The Mixing network combines weights and intrinsic rewards to output global rewards for each agent. In addition, we apply an intrinsic reward function to the Actor-Critic

algorithm, where each agent's individual policy is updated under the direction of the corresponding proxy critic. Benefitting from these improvements, AIIR-MIX generates a more appropriate global reward and reduces artificial intervention in reward function design.

We evaluate the AIIR-MIX method on StarCraft II micromanagement benchmark Samvelyan et al. (2019). The experimental results show that the AIIR-MIX performs better than the mainstream algorithms such as LIIR and QMIX in both homogeneous and heterogeneous maps. We conduct ablation experiments and demonstrate that both AIIR and MIX perform better than the baseline algorithm when they are used individually. Moreover, we visualize the training process and show the dynamic change process of attention weights and intrinsic reward as time advances in the complete trajectory. The results demonstrate the effectiveness and importance of the intrinsic reward based on the attention mechanism.

2. Related Work

2.1. Sparse Reward

In many Reinforcement Learning (RL) tasks, the reward from the environment is sparse. Therefore, it may take agents too many steps to reach the state with a positive reward. This situation causes many problems such as low efficiency and exploration difficulty.

Researchers have proposed an intrinsic motivation approach to address the sparse reward problem, in which an intrinsic reward function will be designed to generate intrinsic rewards for agents to promote learning efficiency. Pathak et al. (2017) proposed using prediction errors in pixel space as curiosity rewards to drive exploration in a self-supervised manner. Strehl and Littman (2008) recorded counts of accessed state-action pairs in table form and converted the counts into an intrinsic reward, which was additionally added to the reward from environmental feedback. Song et al. (2018) extended generative adversarial imitation learning to the multi-agent field, but the need for expert demonstrations limited its generality. Hao et al. (2019) combined generative adversarial imitation learning and self-limitation learning and applied them to multi-agent systems to facilitate multi-agent cooperation, but still did not fundamentally reduce the difficulty of training when the number of agents was large.

In order to design an intrinsic reward function that can solve the sparse reward issue more efficiently, the attention mechanism, a method of establishing dependencies between agents, can be utilized by us to solve the sparse reward issue in MARL.

2.2. Attention mechanism

The attention mechanism, a method capable of automatically selecting significant information, is widely utilized in computer vision, natural language processing, and reinforcement learning. In recent years, the attention mechanism has been introduced into MARL to facilitate the learning effectiveness of agents to some extent. For example, Multi-Actor-Attention-Critic (MAAC) Iqbal and Sha (2019) applies an attention mechanism to model centralized Critic networks. The Attention communication (ATOC) Jiang and Lu (2018) model proposes a bi-directional LSTM communication channel with an attention layer, whose attention mechanism allows each agent to focus on messages from other agents according to their state-related importance.

Inspired by these work, we propose an intrinsic reward generation network based on an attention mechanism. Unlike existing methods, this paper utilizes an attention mechanism in generating intrinsic rewards for each agent. It adaptively processes historical information from other agents, focuses on each agent's contribution in teamwork, and generates more precise intrinsic rewards. At the same time, the intrinsic reward network is combined with the Actor-Critic architecture to improve its performance.

3. Background

In generally, a fully cooperative multi-agent problem can be described as a decentralized partially observable Markov decision process (Dec-POMDP) Oliehoek and Amato (2016) consisting of a tuple $G =< \mathcal{N}, S, U, P, Z, O, r, \gamma, \rho_0 >$. $\mathcal{N} = \{1, 2, \cdots, n\}$ denote the set of n agents and $i \in \mathcal{N}$. $s \in S$ is the state of the environment which includes global information for all agents. U is the set of actions. Z is the agent's observation set, each agent gets its own observation $o_i \in Z$ according to the observation function $O(s, i) : S \times \mathcal{N} \to Z$. At each timestep, each agent $i \in \mathcal{N} \equiv \{1, \cdots, n\}$ chooses an action u_i through the parameterized policy network $\pi_i(o_i)$ according to the current observation o_i, forming a joint action $u_i \in U \equiv U^n$ and leading to next state s' according to the transition function $P(s'|s, u) : S \times U \times S \to [0, 1]$. $\boldsymbol{\pi} = \{\pi_1, \pi_2, \cdots, \pi_n\}$ denotes the joint policy consists of the policy of each agent. In order to distinguish different rewards, we denote the team reward from the environment as extrinsic reward $\mathbf{r}^{\text{ex}}$. The intrinsic reward set that will be learned as $\mathbf{r}_t^{\text{in}} = \{r_i^{\text{in}}\}_{i=1}^{n}$, where t is the index of the timestep. $\mathbf{r}^{\text{ex}}(s, u_i) : S \times U \to \mathbb{R}$ is the team reward for each agent i from environment. $\rho_0 : S \to \mathbb{R}$ is the distribution of the initial state s_0. In a fully cooperative multi-agent problem, each agent receives the same $\mathbf{r}^{ex}$ to promote cooperative behavior.

The learning objective of the cooperative multi-agent problem is that n agents learn a policy network π_i parameterized by θ_i to maximize the global cumulative discounted reward set $\mathbf{r}_t^{\text{total}}$. That is, when $\mathbf{r}_t^{\text{total}}$ is the largest, the optimal joint policy of all agents is obtained $\boldsymbol{\pi}^* = \underset{\pi}{\operatorname{argmax}} \, \mathbb{E}_{s_0, u_0, \ldots, s_n, u_n} \left[\sum_{t=0}^{T} \gamma^t \mathbf{r}_t^{\text{ex}} \right]$, where $\gamma \in [0, 1)$ is a discount factor and T is the maximum number of steps.

3.1. Policy Gradient

The goal of reinforcement learning is to find an optimal behavior policy for the agent so as to obtain the maximum reward. The main characteristic of the policy gradient method is to model and optimize the policy directly. A policy is usually modeled as a function $\pi_\theta(a|s)$ parameterized by θ. For each training iteration, the parameter θ is changed in the direction given by the gradient $\nabla_\theta J(\theta)$ to find the optimal θ^*, the gradient related to the parameter is expressed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim d^{\pi_\theta}, u \sim \pi_\theta} \left[\boldsymbol{\Psi}(s, u) \nabla_\theta \log \pi_\theta(u|s) \right], \tag{1}$$

where d^{π_θ} denotes a state transition following the policy, and $\boldsymbol{\Psi}(s, u)$ is the trajectory reward related to agent states and actions. The policy gradient algorithm is extended to multi-agent field, and each agent $i \in \{1, \cdots, n\}$ has a policy function $\pi_{\theta_i}(u_i|o_i)$. The multi-agent

policy gradient can be expressed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_{\theta_i} \sim \boldsymbol{\pi_\theta}} \left[\sum_{i=1}^n \boldsymbol{\Psi}(s, \mathbf{u}) \nabla_{\theta_i} \log \pi_{\theta_i}(u_i | o_i) \right], \tag{2}$$

where u_i denotes the action of agent i, o_i denotes the observation of agent i, $\mathbf{u}$ and $\boldsymbol{\pi}$ denotes the joint action and policy of all agents, respectively. There are two training methods for calculating $\boldsymbol{\Psi}(s, \mathbf{u})$. One is to train the policy network through the REINFORCE Williams (1992). The other is to introduce the policy gradient into the Actor-Critic framework, and actors are trained through the gradient of Critic. In this paper, we choose the second method in training. The advantage function, as a common way to measure the value of an agent action, is usually designed as $A_{\boldsymbol{\pi}}(s, \mathbf{u}) = Q_{\boldsymbol{\pi}}(s, \mathbf{u}) - V_{\boldsymbol{\pi}}(s)$, where $Q_{\boldsymbol{\pi}}(s, \mathbf{u})$ is $r(s, \mathbf{u}) + \gamma V_{\boldsymbol{\pi}}(s')$ used to help calculate the advantage function.

3.2. QMIX

QMIX Rashid et al. (2018) applies a mixing network that takes the outputs of all agent networks as input, mixes them monotonically, and adds global state information to the training process to improve algorithm performance. In this method, the weights and biases of the mixing network are generated by each hypernetwork based on the input state s. The generated weights ensure that the weights are non-negative by means of an absolute activation function. For each training iteration, the goal of QMIX is to minimize the loss function:

$$\mathcal{L}(\theta) = \sum_{i=1}^b \left[(y_i^{tot} - Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s; \theta))^2 \right], \tag{3}$$

where $y_i^{tot} = r + \gamma \max_{u'} Q_{tot}(\boldsymbol{\tau}, \mathbf{u}', s'; \bar{\theta}_i)$, $\bar{\theta}_i$ represent the parameters of the target networks, b denotes the batch size of transitions sampled from the replay buffer, and $\boldsymbol{\tau}$ denotes the joint action-observation history. Inspired by this paradigm, we adopt a feed-forward neural network as a mixing network, combining $\mathbf{r}_i^{in}$ and $\mathbf{r}^{ex}$ into the global reward $\mathbf{r}_i^{total}$, thus improving the algorithm performance.

3.3. Attention Mechanism

For attention mechanism, the input is some vectors $\mathcal{V} \equiv \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n\}$. For each vector $\mathbf{v}_i$ in $\mathcal{V}$, according to Equation 4, the output of attention mechanism is $\widetilde{\mathbf{v}}_i$.

$$\widetilde{\mathbf{v}}_i = \sum_{j=1}^n \frac{\mathbf{exp}\left(\sigma\left(\mathbf{v}_i, \mathbf{v}_j\right)\right) \mathbf{v}_j}{\sum_{k=1}^n \mathbf{exp}\left(\sigma\left(\mathbf{v}_i, \mathbf{v}_k\right)\right)}, \tag{4}$$

where $\sigma\left(\cdot, \cdot\right)$ denotes a similarity metric function. AIIR-MIX devises an attention information processing mechanism to calculate the correlation of the historical information of the agents, so that it has the ability to adaptively identify and process meaningful information, and enhance team cooperation between agents.

Figure 1: Architecture of the overall AIIR-MIX framework. At timestep t, the total Actor obtains state information (e.g., health, location, shield, agent's sight range, and others) from the environment and generates actions. Then the $\mathbf{r}^{\mathrm{ex}}$ is obtained from the environment and the intrinsic reward set $\mathbf{r}^{\mathrm{in}}_t$ is generated by AIIR according to state information and actions. And Mixing network obtains the global reward set $\mathbf{r}^{\mathrm{total}}_t$ by combining $\mathbf{r}^{\mathrm{in}}_t$ and $\mathbf{r}^{\mathrm{ex}}$ through non-linear operations.

4. Method

In this Section, we introduce a new method called AIIR-MIX, which is based on the Actor-Critic framework. The main contribution of AIIR-MIX is to generate set $\mathbf{r}^{\mathrm{in}}_t$ (w.r.t., the timestep is t) and combine it with $\mathbf{r}^{\mathrm{ex}}$ to generate set $\mathbf{r}^{\mathrm{total}}_t$ nonlinearly. We first present the overall AIIR-MIX framework in Fig. 1, and then we specifically show the relevant details about AIIR-MIX in Fig. 2.

As shown in Fig. 1, the total AIIR-MIX framework consists of five parts: extrinsic Critic network, total Critic network, total Actor network, AIIR, and Mixing network. AIIR and Mixing network make up a module for generating $\mathbf{r}^{\mathrm{total}}_t$, which is the most important part of AIIR-MIX. In particular, the total Actor network is made up of n Actor networks, each Actor network with parameter θ_i generates the policy of its corresponding agent. And we denote the current states s_i as the input and the agent's policy $\pi_{\theta_i}(u_i|s_i)$ as the output of each Actor network. Similarly, each Critic network in total Critic with parameter ω_i evaluates an agent's policy, which can update the Actor network end-to-end. Specifically, each Critic network updates the value function parameters ω_i depending on $Q(s_i, u_i; \omega_i)$ and each Actor network updates the policy parameters θ_i for $\pi_{\theta_i}(u_i|s_i)$ in the direction suggested by its corresponding Critic network. Following the CTDE paradigm, the total

Critic network is only utilized during training. Like Lowe et al. (2017); Foerster et al. (2018), the total Critic network is updated by the Temporal-Difference error (TD-error) as well. But we utilize $\mathbf{r}_t^{\text{total}}$ generated from Mixing network by combining $\mathbf{r}^{\text{ex}}$ and $\mathbf{r}_t^{\text{in}}$ instead of $\mathbf{r}^{\text{ex}}$ to calculate TD-error. To better facilitate teamwork among the agents, we designed AIIR based on an attention mechanism to generate $\mathbf{r}_t^{\text{in}}$. Due to the powerful representational abilities of AIIR-MIX, it can update the total Critic network and the total Actor network better.

Based on the Bellman equation, the loss function of a Critic network can be defined as:

$$\mathcal{L}_{\text{total}}\left(\omega_i\right) = \frac{1}{n} \sum_{i=1}^{n} \left[\left(y_i^{\text{total}} - Q(s_i, u_i; \omega_i)\right)^2 \right], \tag{5}$$

where $i \in \mathcal{N} \equiv \{1, \cdots, n\}$ and $y_i^{\text{total}} = r_i^{\text{total}} + \gamma Q(s_i', u_i'; \omega_i')$. Of particular note is our definition of r_i^{total} is the global reward of agent i, which is an element in $\mathbf{r}_t^{\text{total}}$. Each Actor network is updated by the policy gradient:

$$\nabla_{\theta_i} J\left(\theta_i\right) = E_{\pi_{\theta_i}} \left[A_{\pi_{\theta_i}}\left(u_i | s_i\right) \nabla_{\theta_i} \log \pi_{\theta_i}\left(u_i | s_i\right) \right], \tag{6}$$

where $A_{\pi_{\theta_i}}\left(u_i | s_i\right)$ is the advantage function based on $A_{\boldsymbol{\pi}}(s, \mathbf{u})$ (have been introduced in Section 3.1) and $A_{\pi_{\theta_i}}\left(u_i | s_i\right) = r^{ex}\left(s_i, u_i\right) + \gamma V \pi_{\theta_i}\left(s_i\right) - V \pi_{\theta_i}\left(s_i'\right)$. With a learning rate of α for the Actor network, the update of the parameter θ_i can be defined as:

$$\theta_{i,t+1} \leftarrow \theta_{i,t} + \alpha A_{\pi_{\theta_i}}\left(u_i | s_i\right) \nabla_{\theta_i} \log \pi_{\theta_i}\left(u_i | s_i\right). \tag{7}$$

By referring to Fig. 1, all agent share the extrinsic Critic network. The loss function of this extrinsic Critic network with parameter η can be defined as:

$$\begin{aligned} &\mathcal{L}_{ex}\left(\eta\right) = \left[y^{ex} - Q(\mathbf{s}, \mathbf{u}; \eta)\right]^2, \\ &\textbf{where } \mathbf{s} \equiv \{s_i\}_{i=1}^{n}, \mathbf{u} \equiv \{u_i\}_{i=1}^{n}, y^{ex} = \mathbf{r}^{ex} + \gamma Q(\mathbf{s}', \mathbf{u}'; \eta'). \end{aligned} \tag{8}$$

Then, the generating procedure of $\mathbf{r}_t^{\text{in}}$ and $\mathbf{r}_t^{\text{total}}$ is described in detail. To promote cooperation amongst agents and meet the objective of maximizing the global reward for $\mathbf{r}_t^{\text{in}}$, we design an intrinsic reward generation framework based on an attention mechanism.

As shown in Fig.2c, we first define a feature extractor, to process the state s_i^t and the last action u_i^t of the agent i at timestep t, where the feature extractor is a sequence of consecutive ReLU-FC-ReLU-FC. Then, $\mathbf{v}_i^t$ appears both as an output of the feature extractor and as an input to the attention mechanism, representing the local attention embedding of the agent. With the help of the attention mechanism, we get the correlation of all agent pairs. Commonly, the correlation of agent pairs is obtained by a distance metric function, and we apply cosine similarity to calculate the correlation between the attention embeddings of the agents in this paper:

$$A_{i,j}^t = \frac{<\mathbf{v_i^t}, \mathbf{v_j^t}>}{\parallel \mathbf{v_i^t} \parallel_2 \cdot \parallel \mathbf{v_j^t} \parallel_2} \tag{9}$$

In addition, we perform softmax function on the calculated correlations:

$$\widehat{\mathbf{A}}_{i,j}^t = \text{softmax}\left(A_{i,:}^t\right) = \frac{\exp\left(A_{i,j}^t\right)}{\sum_k \exp\left(A_{i,k}^t\right)}. \tag{10}$$
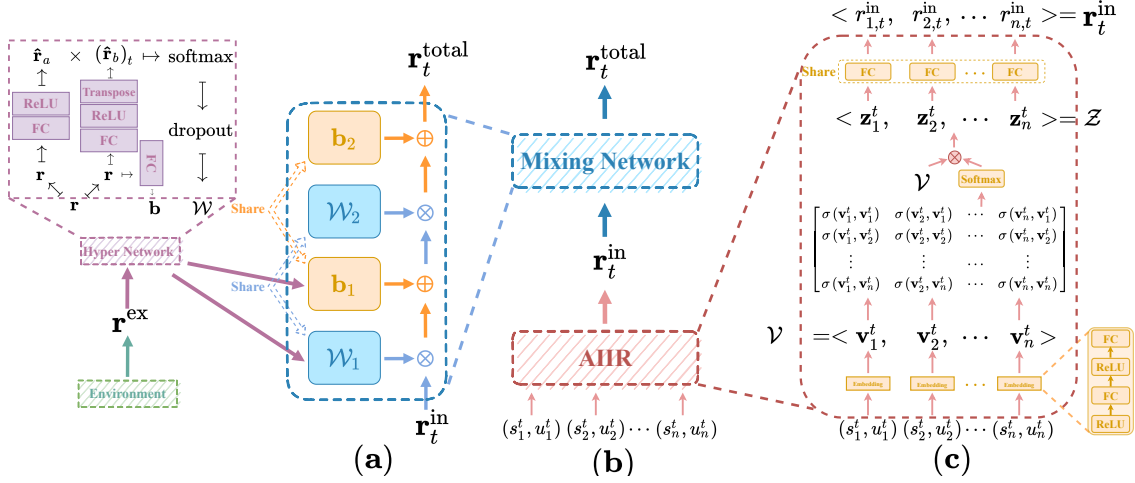
Figure 2: **(a)** Mixing network structure. **(b)** The overall architecture for generating the global reward $\mathbf{r}_t^{\text{total}}$. **(c)** AIIR structure.

The global attention embeddings $\mathbf{z}_t^i$ of agent i is obtained by weighting and summing the attention embeddings of all agents according to the weighting coefficients:

$$\mathbf{z}_i^t = \text{attn}\left(\widehat{\mathbf{A}}_{i,:}^t, \mathbf{v}_i^t\right) = \sum_j \widehat{\mathbf{A}}_{i,j}^t \cdot \mathbf{v}_{i,j}^t, \tag{11}$$

where $\mathbf{z}_i^t$ contains the degree of similarity in the states and actions of the agents, learns the correlations between agents and promotes teamwork. After that, the local attention embedding $\mathbf{v}_i^t$ of agent i and the global attention embedding $\mathbf{z}_i^t$ obtained through the attention mechanism are simultaneously provided as inputs to the fully connected layer. Finally, r_i^{in} of agent i and the set of all intrinsic rewards $\mathbf{r}_t^{\text{in}}$ are output:

$$\mathbf{r}_t^{\text{in}} = \left\{r_i^{\text{in}}\right\}_{i=1}^n, \ \text{where} \ r_i^{\text{in}} = \mathbf{FC}(\mathbf{z}_i^t). \tag{12}$$

As shown in Fig. 2a, $\mathbf{r}_t^{\text{in}}$ output by AIIR is non-linearly combined with r^{ex} through the Mixing network to output $\mathbf{r}_t^{\text{total}}$. In previous studies, both LIIR Du et al. (2019) and GIIR Wu et al. (2021) combined $\mathbf{r}_t^{in}$ and $\mathbf{r}^{\text{ex}}$ by weighted summation:

$$\mathbf{r}_t^{\text{total}} = \mathbf{r}^{\text{ex}} + \lambda \mathbf{r}_t^{\text{in}}. \tag{13}$$

In AIIR-MIX, we apply a non-linear approach to combine $\mathbf{r}_t^{\text{in}}$ and r^{ex} to dynamically generate $\mathbf{r}_t^{\text{total}}$, so that the agent can obtain a more accurate reward at each timestep, thus facilitating the agent to select the optimal policy more readily. The weights $\{\mathcal{W}_1, \mathcal{W}_2\}$ and biases $\{\mathbf{b}_1, \mathbf{b}_2\}$ of the Mixing network are generated by a separate hyper network, as shown in the left of Fig. 2a. The r^{ex} is utilized as input to the hyper network, which generates same weights $\{\mathcal{W}_1, \mathcal{W}_2\}$ and biases $\{\mathbf{b}_1, \mathbf{b}_2\}$ through different linear layers and outputs them as weights and biases of the Mixing network, respectively. Combining the gradient descent method and the hyper network to update the Mixing network, allows for

the dynamic adjustment of the weights of the Mixing network, thus enabling to combine $\mathbf{r}_t^{\text{in}}$ and r^{ex}:

$$
\begin{aligned}
\mathcal{W}_1 \quad &= \mathcal{W}_2 \quad = \text{dropout}\left(\text{softmax}\left(\text{FC}\left(\text{ReLU}\left(r^{\text{ex}}\right)\right) \times \text{FC}\left(\text{ReLU}\left(r^{\text{ex}}\right)\right)^T\right)\right), \\
\mathbf{b}_1 \quad &= \mathbf{b}_2 \quad = \text{FC}\left(r^{\text{ex}}\right), \\
\mathbf{r}_t^{total} &= \left\{r_i^{\text{in}} \times \mathcal{W}_1 \times \mathcal{W}_2 + \mathbf{b}_1 \times \mathcal{W}_2 + \mathbf{b}_2\right\}_{i=1}^{n}.
\end{aligned}
\tag{14}
$$

Then $\mathbf{r}_t^{\text{total}}$ is obtained by AIIR-MIX, which is introduced in Fig. 2b in more detail.

5. Experiments

In this Section, we evaluate our AIIR-MIX method on the StarCraft Multi-Agent Challenge (SMAC) environment Samvelyan et al. (2019), which has become a benchmark for evaluating MARL methods. We compare AIIR-MIX with the state-of-the-art MARL methods such as LIIR Du et al. (2019), QMIX Rashid et al. (2018), COMA Foerster et al. (2018), QTRAN Son et al. (2019). We conduct ablation experiments to demonstrate the effectiveness and rationality of AIIR and Mixing network. Finally, in order to analyze the learning process of the agents more clearly, we visualize the attention weights and intrinsic reward at each timestep.

5.1. StarCraft II Micromanagement

The StarCraft Multi-Agent Challenge (SMAC) environment is an experimental environment based on the real-time strategy game StarCraft II. Compared with the full StarCraft II, it focuses more on the micro-strategy of each agent than on macro-operations, that is, SMAC focuses on how to control each agent to defeat the enemy without considering the high-level macro-operations such as how to develop the economy and perform resource scheduling.

Table 1: Maps in different scenarios.

Name	Ally Units	Enemy Units	Type
2s3z	2 Zealots 3 Stalkers	2 Zealots 3 Stalkers	heterogeneous
3s5z	3 Zealots 5 Stalkers	3 Zealots 5 Stalkers	heterogeneous
8m	8 Marines	8 Marines	homogeneous
MMM	1 Medivac 2 Marauders 7 Marines	1 Medivac 2 Marauders 7 Marines	heterogeneous

In the experiment, this paper applies all the default settings in SMAC, including game difficulty settings, shooting range and observation range. Both the shooting range and the observation range are circles with a certain radius. Only the agents within the observation range can enter the field of view, and only the agents within the shooting range can be attacked. As shown in the environment of Fig. 1, the attributes of the agents include weapon cooling down (CD), health point (HP), shield (2S3Z and 3S5Z), unit type, relative distance of the unit being observed, and last action. The action space of an agent consists of four discrete actions: move[direction], attack[enemy id], stop and noop. The agent movement
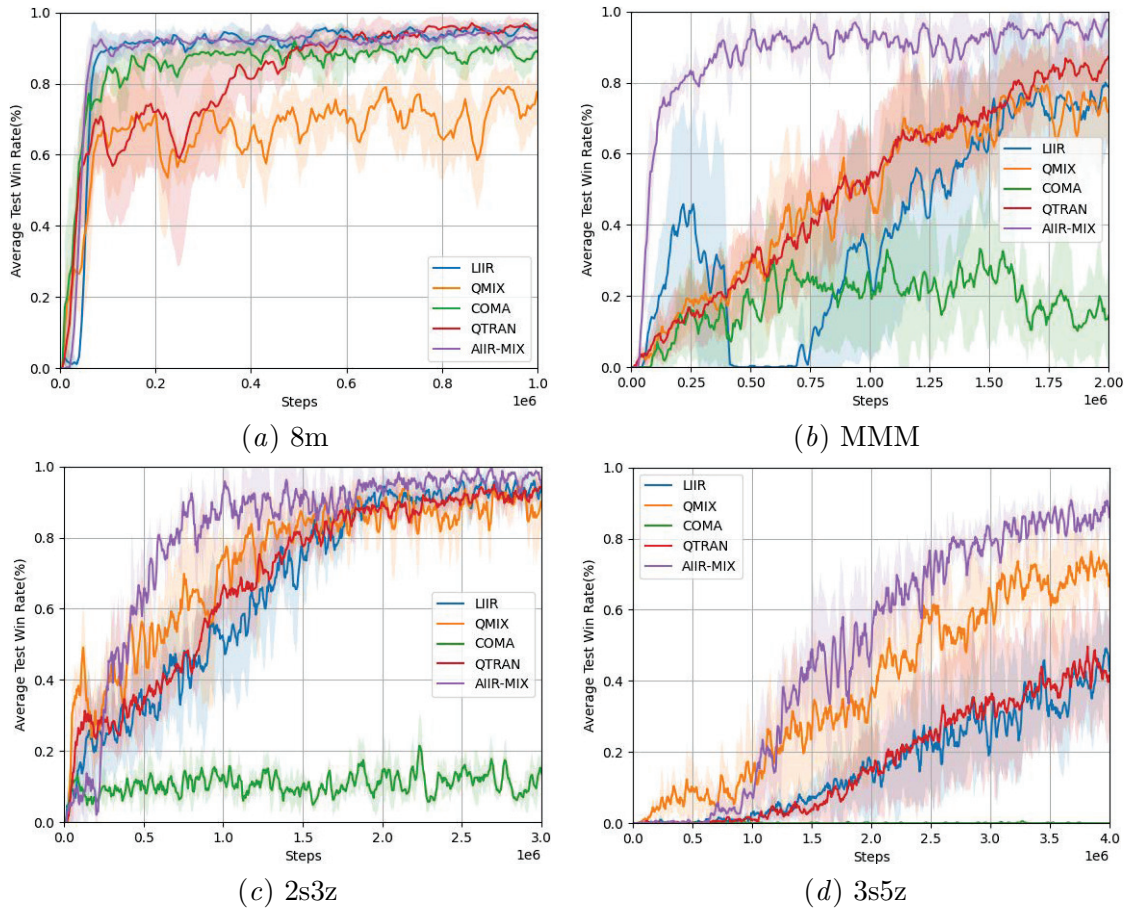
LI[⊠] LIU SHAO HUANG

Figure 3: Average Test Win Rate for LIIR, QMIX, COMA, QTRAN, and AIIT-MIX on Homogeneous and Heterogeneous Scenarios

space includes four directions: east, south, west and north. The attack action requires designating the enemy id within its shooting range. We select four challenging symmetric scenarios, 2s3z, 3s5z, MMM and 8m to evaluate the performance of the algorithms. 2s3z, 3s5z and MMM are heterogeneous maps and 8m is a homogeneous map. The scenario details for different maps of SMAC are shown in Table 1. Different game characters have different health point, attack power and shooting range.

We train these methods in 5 independent runs with different random seeds. During each run, these methods are evaluated every 5000 timesteps of training in 20 independent evaluation episodes. The evaluation episode will be regarded as the winning episode if all enemy units are defeated in time limit, then the percentage of the winning episodes is calculated as the win rate.
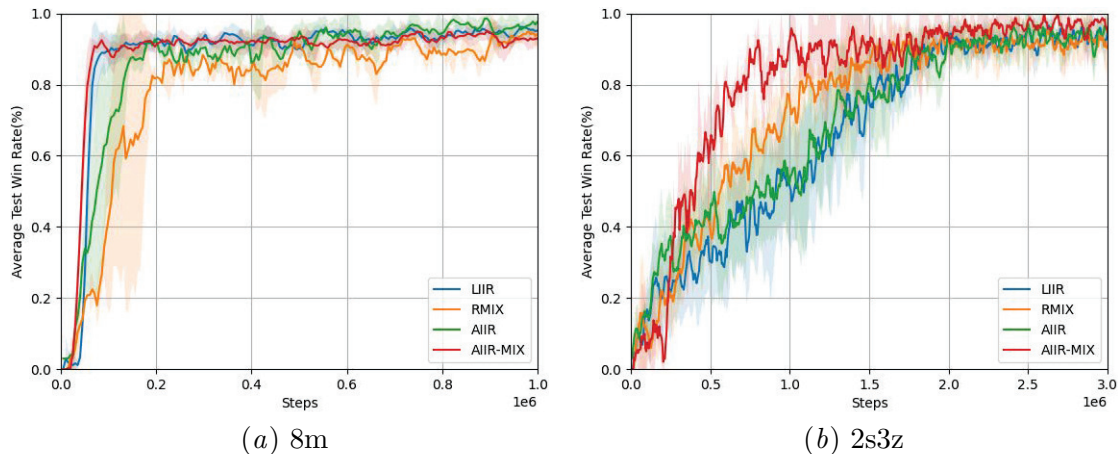
Figure 4: Average Test Win Rate for AIIR-MIX and ablations on 8m and 2s3z.

5.2. Comparison

Fig. 3 demonstrates the performance on 4 different maps in SMAC. Among all the baseline algorithms, QMIX and QTRAN are the most advanced algorithms among the current predominant value decomposition algorithms. In MMM and 2s3z, both QMIX and QTRAN obtain good performance. QMIX obtains a good performance in 3s5z. However, QMIX and QTRAN converge at a much slower pace than other state-of-the-art algorithms and QMIX performs relatively poorly in 8m. These results show that QMIX and QTRAN based on value decomposition can master the heterogeneous scenarios to some extent, but has a relatively poor performance in homogeneous scenarios. COMA fails to perform well in MMM, 2s3z, and 3s5z scenarios, and this result confirms that there is a performance gap between counterfactual policy gradients and TD dominance policy gradients in guiding the Actor network. LIIR can master homogeneous scenarios, such as 8m, but in heterogeneous scenarios, such as in MMM, 2s3z, and 3s5z, it has a slow convergence speed and poor performance. For all scenarios, the AIIR-MIX algorithm consistently outperforms the other algorithms. This result concludes that the Mixing network, which combines the intrinsic reward generated by the attention mechanism and extrinsic reward, can significantly contribute to obtaining better trained policies.

5.3. Ablations

We conduct ablation experiments to investigate the effect of the attention mechanism on AIIR and the necessity of non-linear transformation in Mixing network. For one thing, we analyze the importance of the attention mechanism in AIIR by comparing it with RMIX (i.e., Mixing network), which employs the same intrinsic reward network structure as in LIIR instead of an attention mechanism to generate intrinsic reward. For another, we investigate the necessity of the non-linear Mixing network. We replace the Mixing network with a linear Mixing network in which intrinsic and extrinsic reward of each agent are simply weighted and summed to generate the global reward.

Li^(✉) Liu Shao Huang

(*a*) Intrinsic reward
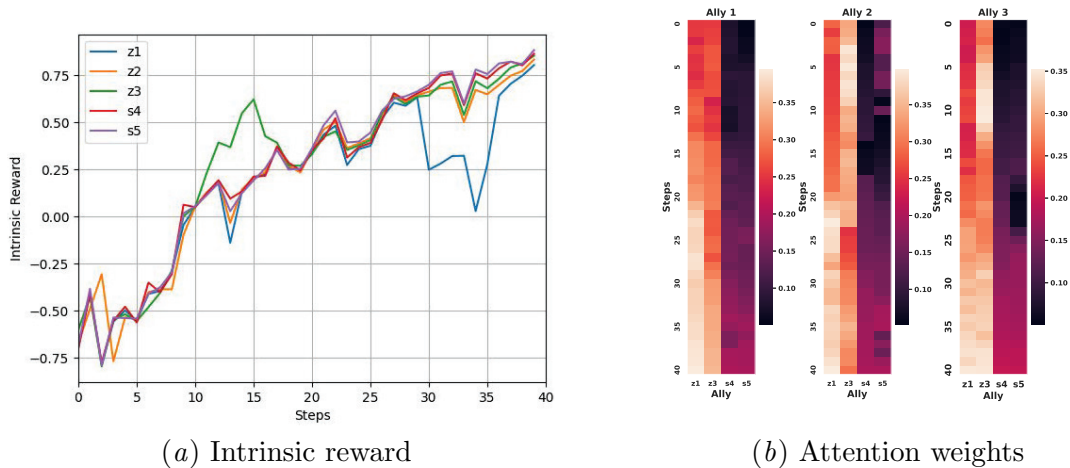
(*b*) Attention weights

Figure 5: Intrinsic Reward and Attention Weights in AIIR-MIX.

Fig. 4 shows the results of AIIR-MIX and its ablations on SMAC benchmark 8m and 2s3z. It can be seen that AIIR-MIX outperforms all its ablations in the experiments. Fig. 4a shows that in homogeneous scenarios, RMIX and AIIR slow down the learning speed compared with the baselines algorithm LIIR due to the complexity of the architecture. However, when the attention-based intrinsic reward function and the non-linear Mixing network are together in AIIR-MIX, the improvement is capable of making up for the decrease in training speed results from the complex architecture. Fig. 4b shows that in heterogeneous scenarios, a non-linear Mixing network is required to achieve better performance. Also, the performance of AIIR-MIX compared with RMIX demonstrates the importance of precise intrinsic reward and the combination with extrinsic reward through a non-linear approach.

5.4. Visualization

In addition to evaluate the performance of the trained policy in Section 5.2, we are more curious about how much the attention mechanism contributes to learning intrinsic reward, and the learned intrinsic reward contributes to policy learning. In order to figure out what is learned in the above two processes, we propose to explicitly visualize the attention weights and intrinsic reward at each timestep in the trajectory. For clarity, we choose scenario 2s3z which has a relatively small number of agents for our analysis. Fig. 5 shows the attention weights and intrinsic rewards of the agents. And Fig. 6 shows some auxiliary snapshots of the agents. The above figures include agent type (s means Stalker, z means Zealot) and agent id (from 1 to 5).

It can be seen in Fig. 5a that after timestep 10, the intrinsic reward of ally Zealot 3 increases a lot. The reason for this phenomenon is clearly illustrated in Fig. 6a and Fig. 6b. Before timestep 10, ally Zealot 3 with a low HP is attacking the enemy Zealot 3. After timestep 10, while ally Zealot 2 starts attacking the enemy, ally Zealot 3 stops firing and flees. Avoiding attacking the enemy head-on is certainly a good behavior when the agent does not have enough HP. After timestep 30, the intrinsic reward of ally Zealot 1 decreases
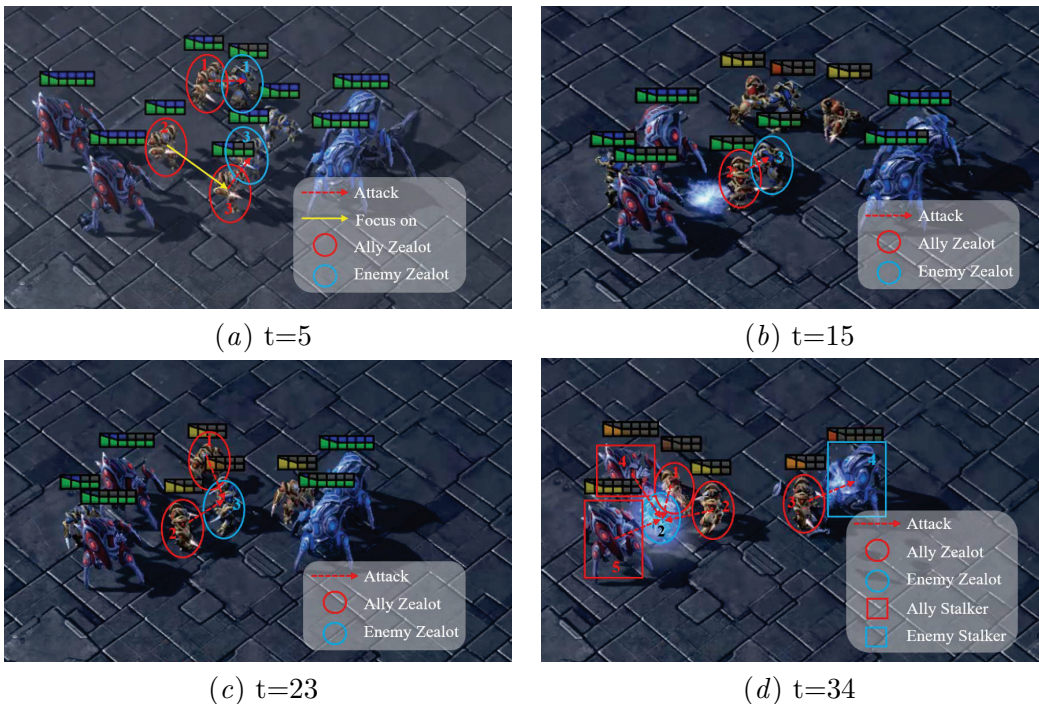
Figure 6: Some auxiliary snapshots in 2s3z.

a lot. As shown in Fig. 6d, ally Zealot 1 is attacking enemy Zealot 2 along with ally Zealot 2, 4 and 5. However, ally Zealot 1's HP is very low at this point, and it is not a good policy to continue attacking the enemy. In general, the intrinsic reward increases when the agent chooses a good policy, and decreases when the agent chooses a bad policy. This more precise reward contributes significantly in helping the agent learn good policies quickly.

As shown in Fig. 5a, the intrinsic reward are more fluctuating for Zealot-type agents. For a more detailed analysis, we visualize the attention weights for Zealot-type agents. Fig. 5b shows the attention weights of three Zealots to other allied agents. It is clear that Zealots always pay more attention to the other two Ally Zealots. The elevated attention weights indicate the occurrence of cooperative behavior between agents, including ally Zealot 1 to ally Zealot 2 at timestep 23 and ally Zealot 2 to ally Zealot 3 at timestep 5 in Fig. 6. That is, the attention mechanism facilitates the cooperative behavior between agents in a certain extent.

6. Conclusion

This paper presents AIIR-MIX, a novel multi-agent RL algorithm that learns an individual intrinsic reward through an attention mechanism for each agent so that the agent can obtain different rewards to facilitate the learning of the agent. Besides, we design a non-linear Mixing network to combine intrinsic and extrinsic rewards instead of a simple linear summation for the first time, thereby dynamically generating global rewards for each agent according to the changing environment. Our empirical results of the experiments carried out on the battle games in StarCraft II demonstrate that our approach induces trained

policy compared with a few state-of-the-art MARL methods better. We further conduct ablation studies to confirm the effectiveness of the intrinsic reward network based on the attention mechanism and the non-linear Mixing network. And we visualize the intrinsic rewards and attention weights to illustrate how the intrinsic reward network assigns each agent an appropriate reward and promotes cooperation amongst agents.

Acknowledgments

References

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019. URL https://doi.org/10.48550/arXiv.1912.06680.

Ignacio Carlucho, Mariano De Paula, and Gerardo G Acosta. An adaptive deep reinforcement learning approach for mimo pid control of mobile robots. *ISA Transactions*, 102: 280–294, 2020. doi: 10.1016/j.isatra.2020.02.017. URL https://doi.org/10.1016/j.isatra.2020.02.017.

Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019. URL https://proceedings.neurips.cc/paper/2019/file/07a9d3fed4c5ea6b17e80258dee231fa-Paper.pdf.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 29, 2016. URL https://proceedings.neurips.cc/paper/2016/file/c7635bfd99248a2cdef8249ef7bfbef4-Paper.pdf.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, New Orleans, Louisiana, Apr. 2018. AAAI. URL https://ojs.aaai.org/index.php/AAAI/article/view/11794.

Xiaotian Hao, Weixun Wang, Jianye Hao, and Yaodong Yang. Independent generative adversarial self-imitation learning in cooperative multiagent systems. *arXiv preprint arXiv:1909.11468*, 2019. URL https://doi.org/10.48550/arXiv.1909.11468.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970, Long Beach, United States, Jun. 2019. PMLR. URL https://proceedings.mlr.press/v97/iqbal19a.html.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019. doi: 10.1126/science.aau6249. URL https://doi.org/10.1126/science.aau6249.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems*, 31, 2018. URL https://proceedings.neurips.cc/paper/2018/file/6a8018b3a00b69c008601b8becae392b-Paper.pdf.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017. URL https://doi.org/10.48550/arXiv.1706.02275.

Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. Deep reinforcement learning: an overview. In *Proceedings of SAI Intelligent Systems Conference*, pages 426–440, London, UK, Sep. 2016. Springer. doi: 10.1007/978-3-319-56991-8_32. URL https://doi.org/10.1007/978-3-319-56991-8_32.

Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, 50(9):3826–3839, 2020. doi: 10.1109/TCYB.2020.2977374. URL https://doi.org/10.1109/TCYB.2020.2977374.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016. URL https://doi.org/10.1007/978-3-319-28929-8.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787, Sydney, NSW, Australia, Aug. 2017. PMLR. URL https://proceedings.mlr.press/v70/pathak17a.html.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, volume 80, pages 4295–4304, Stockholm, Sweden, Sep. 2018. PMLR. URL https://doi.org/10.48550/arXiv.1803.11485.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019. URL https://doi.org/10.48550/arXiv.1902.04043.

Yih-Liang Shen, Chao-Yuan Huang, Syu-Siang Wang, Yu Tsao, Hsin-Min Wang, and Tai-Shih Chi. Reinforcement learning based speech enhancement for robust speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and*

Signal Processing (ICASSP), pages 6750–6754, Brighton, UK, May. 2019. IEEE. doi: 10. 1109/ICASSP.2019.8683648. URL https://doi.org/10.1109/ICASSP.2019.8683648.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, volume 97, pages 5887–5896, Long Beach, United States, Sep. 2019. PMLR. URL https://doi.org/10.48550/arXiv.1905.05408.

Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. *Advances in neural information processing systems*, 31, 2018. URL https://proceedings.neurips.cc/paper/2018/file/240c945bb72980130446fc2b40fbb8e0-Paper.pdf.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008. doi: 10.1016/j.jcss.2007.08.009. URL https://doi.org/10.1016/j.jcss.2007.08.009.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017. URL https://doi.org/10.48550/arXiv.1706.05296.

Tonghan Wang*, Jianhao Wang*, Yi Wu, and Chongjie Zhang. Influence-based multi-agent exploration. In *International Conference on Learning Representations*, Addis Ababa, Ethiopia, Apr. 2020. ICLR. URL https://openreview.net/forum?id=BJgy96EYvr.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992. URL https://doi.org/10.1007/BF00992696.

Haolin Wu, Hui Li, Jianwei Zhang, Zhuang Wang, and Jianeng Zhang. Generating individual intrinsic reward for cooperative multiagent reinforcement learning. *International Journal of Advanced Robotic Systems*, 18(5):17298814211044946, 2021. doi: 10.1177/17298814211044946. URL https://doi.org/10.1177/17298814211044946.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational IntelligenCe Magazine*, 13(3):55–75, 2018. doi: 10.1109/MCI.2018.2840738. URL https://doi.org/10.1109/MCI.2018.2840738.