Learning from Good Trajectories in Offline Multi-Agent Reinforcement Learning

Qi Tian[1], Kun Kuang[1,*], Furui Liu[2], Baoxiang Wang[3]

[1] College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[2] Huawei Noah's Ark Lab, Beijing, China
[3] School of Data Science, Chinese University of Hong Kong (Shenzhen), Shenzhen, China
{tianqics,kunkuang}@zju.edu.cn, liufurui2@huawei.com, bxiangwang@cuhk.edu.cn

Abstract

Offline multi-agent reinforcement learning (MARL) aims to learn effective multi-agent policies from pre-collected datasets, which is an important step toward the deployment of multi-agent systems in real-world applications. However, in practice, each individual behavior policy that generates multi-agent joint trajectories usually has a different level of how well it performs. *e.g.*, an agent is a random policy while other agents are medium policies. In the cooperative game with global reward, one agent learned by existing offline MARL often inherits this random policy, jeopardizing the performance of the entire team. In this paper, we investigate offline MARL with explicit consideration on the diversity of agent-wise trajectories and propose a novel framework called Shared Individual Trajectories (SIT) to address this problem. Specifically, an attention-based reward decomposition network assigns the credit to each agent through a differentiable key-value memory mechanism in an offline manner. These decomposed credits are then used to reconstruct the joint offline datasets into prioritized experience replay with individual trajectories, thereafter agents can share their good trajectories and conservatively train their policies with a graph attention network (GAT) based critic. We evaluate our method in both discrete control (*i.e.*, StarCraft II and multi-agent particle environment) and continuous control (*i.e.*, multi-agent mujoco). The results indicate that our method achieves significantly better results in complex and mixed offline multi-agent datasets, especially when the difference of data quality between individual trajectories is large.

Introduction

Multi-agent reinforcement learning (MARL) has shown its powerful ability to solve many complex decision-making tasks. *e.g.*, game playing (Samvelyan et al. 2019). However, deploying MARL to practical applications is not easy since interaction with the real world is usually prohibitive, costly, or risky (Garcıa 2015), *e.g.*, autonomous driving (Shalev-Shwartz, Shammah, and Shashua 2016). Thus offline MARL, which aims to learn multi-agent policies in the previously-collected, non-expert datasets without further interaction with environments, is an ideal way to cope with practical problems.

*Corresponding author.

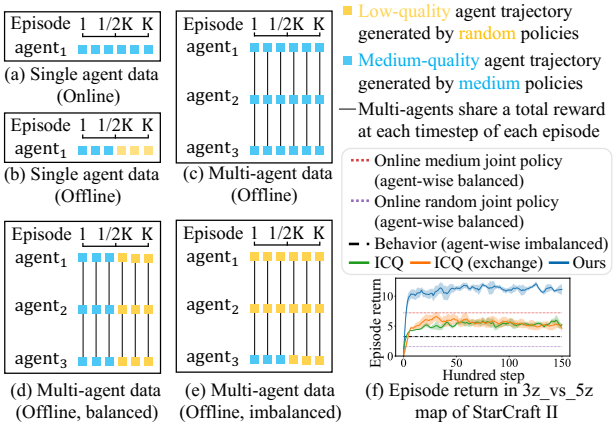


Figure 1: (a)∼(e) Data composition of data replay in episodic online/offline RL/MARL, where the online learning agents in (a) and (c) are assumed to be medium policies. (f) Episode return in 3z_vs_5z map of StarCraft II with the offline data structure of (e).

Recently, Yang et al. (2021) first investigated offline MARL and found that multi-agent systems are more susceptible to extrapolation error (Fujimoto, Meger, and Precup 2019), *i.e.*, a phenomenon in which unseen state-action pairs are erroneously estimated, compared to offline single-agent reinforcement learning (RL) (Fujimoto, Meger, and Precup 2019; Kumar et al. 2019; Wu, Tucker, and Nachum 2019; Kumar et al. 2020; Fujimoto and Gu 2021). Then they proposed implicit constraint Q-learning (ICQ), which can effectively alleviate this problem by only trusting the state-action pairs given in the dataset for value estimation.

In this paper, we point out that in addition to extrapolation error, the diversity of trajectories in the offline multi-agent dataset also deserves attention since the composition of this dataset is much more complex than its single-agent version. For better illustration, we briefly summarize the data quality of data replay in episodic online/offline RL/MARL as is shown in Figure 1. In online settings, since the data replay is updated rollingly based on the learning policy, the quality of all data points is approximately the same as is shown in Figure 1(a) and Figure 1(c), where line connections in Figure

1(c) represent multi-agent systems only provide one global reward at each timestep of each episode. In offline settings, there is no restriction on the quality of collected data, thus the single-agent data replay usually contains multi-source and suboptimal data as is shown in Figure 1(b). Figure 1(d) directly extends the offline single-agent data composition to a multi-agent version. However, in practical tasks, each individual trajectory in an offline multi-agent joint trajectory usually has a different data quality, as is shown in Figure 1(e). For example, consider the task that two robotic arms (agents) work together to lift a large circular object, and the reward is the height of the center of mass. Suppose that two workers operate two robotic arms simultaneously to collect offline data, but they have different levels of proficiency in robotic arm operation. The offline joint data generated in this case is an agent-wise imbalanced multi-agent dataset.

To investigate the performance of the current state-of-the-art offline MARL algorithm, ICQ, on the imbalanced dataset, we test it on the 3s_vs_5z map in StarCraft II (Samvelyan et al. 2019). Specifically, as is shown in Figure 1(f), we first obtain random joint policy (violet dotted) and medium joint policy (red dotted) through online training, and then utilize these two joint policies to construct an agent-wise imbalanced dataset with the data structure of Figure 1(e). We can observe that the performance of ICQ (green) is lower than that of the online medium joint policy (red dotted), as agent_1 and agent_2 may inherit the random behavior policies that generate the data. One might attempt to solve this issue of ICQ by randomly exchanging local observations and actions among agents, so that agent_1 and agent_2 can share some medium-quality individual trajectories of agent_3. Unfortunately, the performance of this alternative (orange) is similar to ICQ, since the proportion of the medium-quality individual data in the data replay does not change under the constraint of the total reward.

In this paper, we propose a novel algorithmic framework called Shared Individual Trajectories (SIT) to address this problem. It first learns an Attention-based Reward Decomposition Network with Ensemble Mechanism (ARDNEM), which assigns credits to each agent through a differentiable key-value memory mechanism in an offline manner. Then these credits are used to reconstruct the original joint trajectories into Decomposed Prioritized Experience Replay (DPER) with individual trajectories, thereafter agents can share their good trajectories and conservatively train their policies with a graph attention network based critic. Extensive experiments on both discrete control (*i.e.*, StarCraft II and multi-agent particle environment) and continuous control (*i.e.*, multi-agent mujoco) indicate that our method achieves significantly better results in complex and mixed offline multi-agent datasets, especially when the difference of data quality between individual trajectories is large.

Related Work

Offline MARL. Recently, Yang et al. (2021) first explored offline MARL and found that multi-agent systems are more susceptible to extrapolation error (Fujimoto, Meger, and Precup 2019) compared to offline single-agent RL (Fujimoto, Meger, and Precup 2019; Kumar et al. 2020; Fujimoto and Gu 2021), and then they proposed implicit constraint Q-learning (ICQ) to solve this problem. Jiang and Lu (2021a) investigated the mismatch problem of transition probabilities in fully decentralized offline MARL. However, they assume that each agent makes the decision based on the global state and the reward output by the environment can accurately evaluate each agent's action. This is fundamentally different from our work since we focus on the partially observable setting in global reward games (Chang, Ho, and Kaelbling 2003), which is a more practical situation. Other works (Meng et al. 2021; Jiang and Lu 2021b) have made some progress in offline MARL training with online fine-tuning. Unfortunately, all the existing methods neglect to investigate the diversity of trajectories in offline multi-agent datasets, while we take the first step to fill this gap.

Multi-agent credit assignment. In cooperative multi-agent environments, all agents receive one total reward. The multi-agent credit assignment aims to correctly allocate the reward signal to each individual agent for a better groups' coordination (Chang, Ho, and Kaelbling 2003). One popular class of solutions is value decomposition, which can decompose team value function into agent-wise value functions in an online fashion under the framework of the Bellman equation (Sunehag et al. 2018; Rashid et al. 2018; Yang et al. 2020b; Li et al. 2021). Different from these works, in this paper, we focus on explicitly decomposing the total reward into individual rewards in an offline fashion under the regression framework, and these decomposed rewards will be used to reconstruct the offline prioritized dataset.

Experience replay in RL/MARL. Experience replay, a mechanism for reusing historical data (Lin 1992), is widely used in online RL (Wang et al. 2020a). Many prior works related to it focus on improving data utilization (Schaul et al. 2016; Zha et al. 2019; Oh et al. 2020). *e.g.*, prioritized experience replay (PER) (Schaul et al. 2016) takes temporal-difference (TD) error as a metric for evaluating the value of data and performs importance sampling according to it. Unfortunately, this metric fails in offline training due to severe overestimation in offline scenarios. In online MARL, most works related to the experience replay focus on stable decentralized multi-agent training (Foerster et al. 2017; Omidshafiei et al. 2017; Palmer et al. 2018), but these methods usually rely on some auxiliary information, *e.g.*, training iteration number, timestamp and exploration rate, which often are not provided by the offline settings. SEAC (Christianos, Schäfer, and Albrecht 2020) proposed by Christianos et al. is most related to our work as it also shares individual trajectories among agents during online training. However, SEAC assumes that each agent can directly obtain a local reward and does not consider the importance of each individual trajectory. Instead, we need to decompose the global reward into local rewards in an offline manner, and then determine the quality of the individual trajectory based on the decomposed rewards for priority sampling.

Preliminaries

A fully cooperative multi-agent task in the global reward game (Chang, Ho, and Kaelbling 2003) can be described as

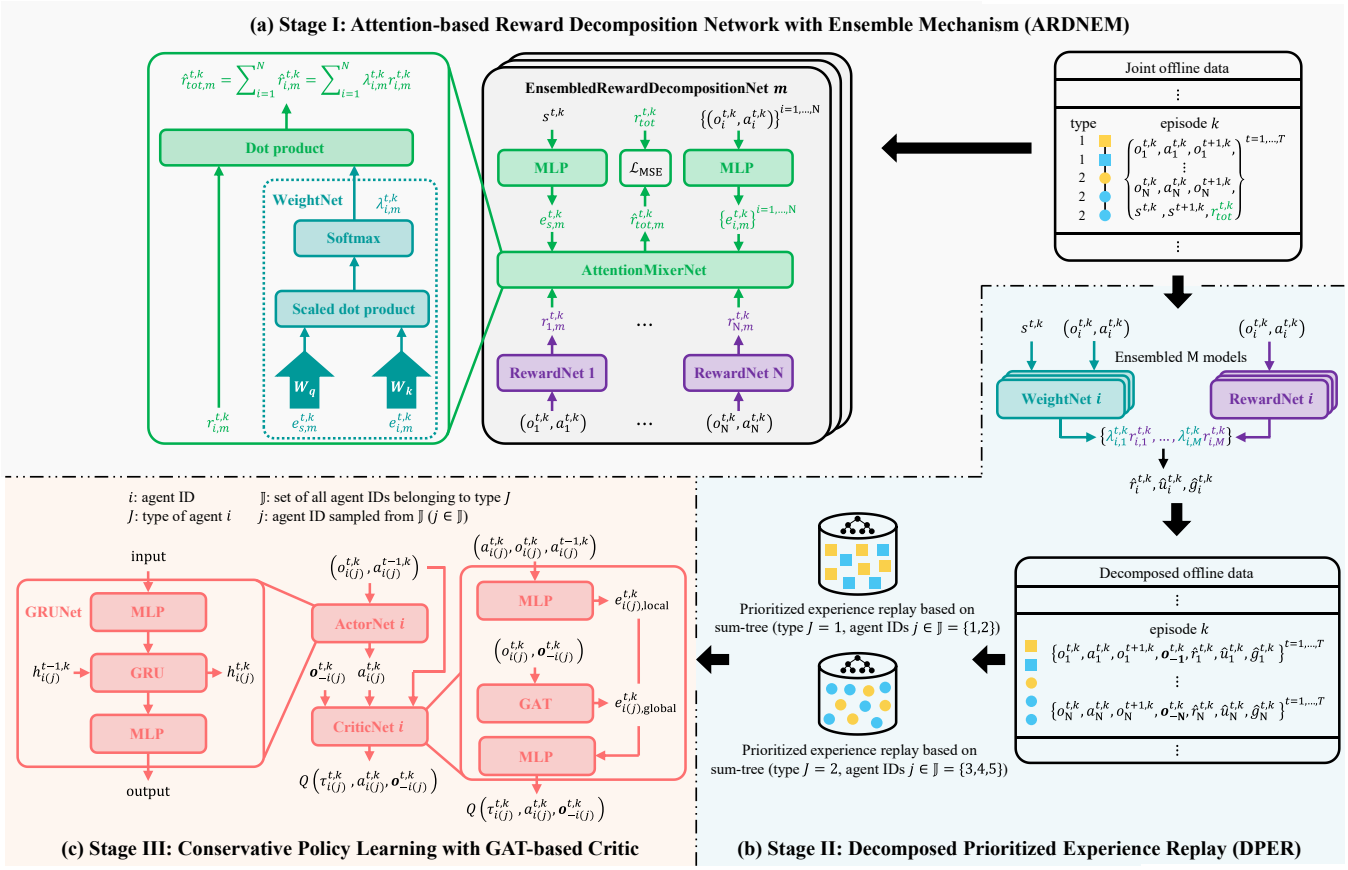


Figure 2: Overall framework of Shared Individual Trajectories (SIT) in offline MARL

a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek and Amato 2016) consisting of a tuple $\langle N, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \Omega, \gamma \rangle$. Let N represent the number of agents and $\mathcal{S}$ represent the true state space of the environment. At each timestep $t \in \mathcal{Z}^+$ of episode $k \in \mathcal{Z}^+$, each agent $i \in N \equiv \{1, \ldots, n\}$ takes an action $a_i \in \mathcal{A}$, forming a joint action $\boldsymbol{a} \in \boldsymbol{\mathcal{A}} \equiv \mathcal{A}^n$. Let $\mathcal{T}(s'|s, \boldsymbol{a}) : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathcal{S}$ represent the state transition function. All agents share the global reward function $r(s, \boldsymbol{a}) : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathcal{R}$ and $\gamma \in [0, 1)$ represents the discount factor. Let J represent the type of agent i, which is the prior knowledge given by the environment (Wang et al. 2020b; Yang et al. 2020a). All agent IDs belonging to type J are denoted as $j \in \mathbb{J}$, where $\mathbb{J}$ represents the set of IDs of all agents isomorphic to agent i (including i). We consider a partially observable setting in which each agent receives an individual observation $o_i \in \Omega$ according to the observation function $\mathcal{O}(s, a) : \mathcal{S} \times \mathcal{A} \to \Omega$. Each agent has an action-observation history $\tau_i \in \boldsymbol{T} \equiv (\Omega \times \mathcal{A})^t$, on which it conditions a stochastic policy $\pi_i(a_i|\tau_i)$. Following Gronauer (2022), we focus on episodic training in this paper.

Methodology

In this section, we propose a new algorithmic framework called Shared Individual Trajectories (SIT) in offline MARL, which can maximally exploit the useful knowledge in the complex and mixed offline multi-agent datasets to boost the performance of multi-agent systems. SIT consists of three stages as shown in Figure 2: Stage I: learning an Attention-based Reward Decomposition Network with Ensemble Mechanism (ARDNEM). Stage II: reconstructing the original joint offline dataset into Decomposed Prioritized Experience Replay (DPER) based on the learned ARDNEM. Stage III: conservative offline actor training with a graph attention network (GAT) based critic on DPER.

Attention-Based Reward Decomposition Network with Ensemble Mechanism

It is a non-trivial problem to evaluate the behavior of individual trajectories in an offline dataset under the global reward games since only one total reward can be accessed in this case. We propose a reward decomposition network, as is shown in Figure 2(a), to solve this problem. Specifically, the individual reward $r_i^{t,k}$ at timestep t of episode k can be approximately estimated from the local observation $o_i^{t,k}$ and action $a_i^{t,k}$ through the reward network f_i. *i.e.*, $r_i^{t,k} = f_i(o_i^{t,k}, a_i^{t,k})$. In order to learn the contribution weight $\lambda_i^{t,k}$ of each individual reward $r_i^{t,k}$ to the total reward, we use the attention mechanism (*i.e.*, the differen-

tiable key-value memory model (Oh et al. 2016)) to achieve this goal. That is, we first encode the global state $s^{t,k}$ and local observation-action pair $(o_i^{t,k}, a_i^{t,k})$ into embedding vectors $e_s^{t,k}$ and $e_i^{t,k}$ with two multi-layer perceptrons (MLPs) respectively, then we pass the similarity value between the global state's embedding vector $e_s^{t,k}$ and the individual features' embedding vector $e_i^{t,k}$ into a softmax

$$\lambda_i^{t,k} \propto \exp((e_s^{t,k})^T W_q^T W_k e_i^{t,k}), \tag{1}$$

where the learnable weight W_q and W_k transform $e_s^{t,k}$ and $e_i^{t,k}$ into the query and key. Thus the estimated total reward $\hat{r}_{tot}^{t,k}$ at each timestep t of episode k can be expressed as

$$\hat{r}_{tot}^{t,k} = \sum_{i=1}^{N} \lambda_i^{t,k} r_i^{t,k} = \sum_{i=1}^{N} \lambda_i^{t,k} \cdot f_i(o_i^{t,k}, a_i^{t,k}). \tag{2}$$

Since misestimated individual rewards will have a large impact on agent learning, we want to obtain the uncertainty corresponding to each estimated value for correcting subsequent training. Following Chua et al. (2018), we introduce ensemble mechanism (Osband et al. 2016) to meet this goal. *i.e.*, M decomposition networks are learned simultaneously. Finally, our Attention-based Reward Decomposition Network with Ensemble Mechanism (ARDNEM) w.r.t. parameters ψ can be trained on the original joint offline dataset with the following mean-squared error (MSE) loss

$$\mathcal{L}_{\mathrm{MSE}}(\psi) = \frac{1}{M} \sum_{m=1}^{M} \left(\sum_{i=1}^{N} \lambda_{i,m}^{t,k} \cdot f_{i,m}(o_i^{t,k}, a_i^{t,k}) - r_{tot}^{t,k} \right)^2, \tag{3}$$

where $r_{tot}^{t,k}$ represents the true total reward in the offline dataset. In practice, the reward network parameters of different agents are shared for the scalability of our method.

Decomposed Prioritized Experience Replay

As is shown in Figure 2(b), after ARDNEM is learned, we can use it to estimate individual rewards for all local trajectories. Specifically, since ARDNEM adopts the ensemble mechanism, we take the mean of M model output $\lambda_{i,m}^{t,k} r_{i,m}^{t,k}$ as the estimation of the weighted individual reward $\hat{r}_i^{t,k}$:

$$\hat{r}_i^{t,k} = \frac{1}{M} \sum_{m=1}^{M} \lambda_{i,m}^{t,k} r_{i,m}^{t,k}. \tag{4}$$

Its corresponding variance is defined as the uncertainty $\hat{u}_i^{t,k}$ of the model prediction:

$$\hat{u}_i^{t,k} = \sqrt{\frac{1}{M} \sum_{m=1}^{M} \left(\lambda_{i,m}^{t,k} r_{i,m}^{t,k} - \hat{r}_i^{t,k} \right)^2}. \tag{5}$$

To maximally exploit the useful knowledge in the data replay, we need a metric to distinguish the importance of each data. Many previous works in online RL do this through temporal-difference (TD) error (Schaul et al. 2016; Zha et al. 2019; Oh et al. 2020). However, the value function suffers from severe overestimation in offline settings (Fujimoto, Meger, and Precup 2019), thus TD error is not an ideal choice. In this paper, considering that offline datasets are static and limited, we believe that high-quality data should be valued. Therefore, we use Monte Carlo return $\hat{g}_i^{t,k}$ to measure the importance (or quality) of the data at each timestep t of episode k:

$$\hat{g}_i^{t,k} = \sum_{t=t'}^{T} \gamma^{t-t'} \hat{r}_i^{t,k}, \tag{6}$$

where γ represents discounted factor.

After the above efforts, each episode k in the original joint trajectories (*i.e.*, $\{o_1^{t,k}, a_1^{t,k}, o_1^{t+1,k}, \ldots, o_{\mathrm{N}}^{t,k}, a_{\mathrm{N}}^{t,k}, o_{\mathrm{N}}^{t+1,k}, s^{t,k}, s^{t+1,k}, r_{tot}^{t,k}\}^{t=1,\ldots,T}$) can be decomposed to the agent-wise individual trajectories (*i.e.*, $\{\{o_i^{t,k}, a_i^{t,k}, o_i^{t+1,k}, \boldsymbol{o}_{-\boldsymbol{i}}^{t,k}, \hat{r}_i^{t,k}, \hat{u}_i^{t,k}, \hat{g}_i^{t,k}\}^{t=1,\ldots,T}\}^{i=1,\ldots,\mathrm{N}}$), where $\boldsymbol{o}_{-\boldsymbol{i}}^{t,k}$ represents all local observations except for agent i. We then store all these individual trajectories into single/multiple Decomposed Prioritized Experience Replay (DPER) according to their agent type. Since we train multi-agent policies in an episodic manner, the mean of Monte Carlo return $\hat{g}_i^{t,k}$ for each episode k is used as the sampling priority $\hat{p}_i^k = \frac{1}{T} \sum_{t=1}^{T} \hat{g}_i^{t,k}$. To trade off priority sampling and uniform sampling, we use a softmax function with a temperature factor α to reshape the priorities in the decomposed dataset:

$$p_i^k = \frac{e^{\hat{p}_i^k / \alpha}}{\sum_{j \in \mathbb{J}, k}^{K} e^{\hat{p}_j^k / \alpha}}, \tag{7}$$

where $\mathbb{J}$ represents the set of IDs of all agents isomorphic to agent i (including i). Temperature factor α determines how much prioritization is used, with $\alpha \to \infty$ corresponding to the uniform sampling.

In practice, considering the large gap in rewards for different environments, all sampling priority $\hat{p}_i^k$ are linearly scaled to a uniform range, allowing our method to share the same temperature factor α across environments. Meanwhile, we use the sum-tree (Schaul et al. 2016) as the storage structure of DPER to improve the sampling efficiency.

Conservative Policy Learning with GAT-Based Critic

In this subsection, we will use the obtained type-wise DPER for multi-agent policy learning under the centralized training decentralized execution (CTDE) paradigm. As is shown in Figure 2(c), the input of the centralized critic for each agent i consists of local information and global information. Specifically, the former includes the local action-observation history $\tau_i^{t,k} = (o_i^{t,k}, a_i^{t-1,k})$ (Peng et al. 2021) and current action $a_i^{t,k}$ of each agent i. We encode them into the local embedding vector $e_{i,\mathrm{local}}^{t,k}$ with two MLPs f_{local}. *i.e.*, $e_{i,\mathrm{local}}^{t,k} = f_{\mathrm{local}}(\tau_i^{t,k}, a_i^{t,k})$. The latter includes local observations of all agents $(o_i^{t,k}, \boldsymbol{o}_{-\boldsymbol{i}}^{t,k})$. We construct these observations as a fully connected graph and aggregate the global embedding vector $e_{i,\mathrm{global}}^{t,k}$ via a graph attention network (GAT)

StarCraft II (SC II)								
		Behavior	BC	QMIX	MABCQ	MACQL	ICQ	Ours
Low Quality	2s_vs_1sc	2.8	3.2±1.2	1.1±0.0	1.7±0.0	4.5±0.1	4.4±0.1	**10.5±0.4**
	3s_vs_5z	2.9	2.9±0.4	2.3±0.2	4.5±0.2	4.5±0.1	5.2±0.1	**11.1±0.8**
	2s3z	3.2	3.1±1.7	2.6±0.0	4.7±0.7	5.7±0.4	9.1±0.3	**12.3±1.6**
	8m	3.1	3.1±0.2	2.0±0.6	5.7±1.2	2.5±0.6	5.5±0.5	**10.8±0.4**
	1c3s5z	5.5	5.9±0.8	2.4±1.9	8.4±1.2	6.1±1.2	9.4±0.0	**13.7±0.3**
	10m_vs_11m	3.8	4.2±0.7	0.7±0.5	4.3±1.8	5.2±0.1	8.0±0.2	**12.3±0.7**
Medium Quality	2s_vs_1sc	9.8	9.8±0.2	3.6±1.6	6.6±1.6	9.8±0.1	10.2±0.0	**16.5±1.8**
	3s_vs_5z	7.0	7.6±0.4	2.6±1.6	5.8±0.8	8.8±1.4	13.5±1.2	**17.8±1.5**
	2s3z	7.0	7.1±0.3	2.3±0.5	5.8±1.2	7.1±0.5	9.1±0.3	**14.8±0.3**
	8m	9.8	9.6±0.3	2.4±0.0	4.0±1.6	10.5±1.2	13.4±0.6	**14.8±1.1**
	1c3s5z	10.3	10.0±0.1	8.1±1.6	11.8±2.0	10.1±0.2	12.7±0.3	**15.4±0.4**
	10m_vs_11m	9.2	9.2±0.3	1.7±0.4	3.5±0.6	9.5±0.5	10.6±0.4	**13.6±1.3**
Multi-agent Particle Environment (MPE)								
		Behavior	BC	QMIX	MABCQ	MACQL	ICQ	Ours
Low Quality	CN_3ls3l	-157.7	161.1±4.2	-231.9±3.0	-174.3±22.7	-217.2±18.1	-117.8±11.5	**-92.4±7.2**
	CN_4ls4l	-278.4	-274.7±6.0	-316.4±17.3	-194.5±8.3	-300.5±7.3	-231.4±2.2	**-120.7±13.3**
	PP_3p1p	-249.5	-253.7±10.8	-336.9±22.9	-239.9±32.3	-326.5±18.6	-227.7±6.7	**-105.6±9.0**
Medium Quality	CN_3ls3l	-107.4	-111.0±1.8	-256.1±3.9	-107.3±13.9	-231.9±13.2	-83.0±3.2	**-54.5±2.9**
	CN_4ls4l	-166.2	-161.4±5.4	-290.0±5.3	-146.0±8.3	-282.8±14.0	-184.6±10.2	**-72.3±2.3**
	PP_3p1p	-155.4	-156.2±7.3	-296.6±28.8	-230.2±27.9	-272.5±14.4	-158.6±5.2	**-80.4±4.7**
Multi-Agent mujoco (MAmujoco)								
		Behavior	BC	FacMAC	MABCQ	MACQL	ICQ	Ours
Low Quality	HalfCheetah_2l	-110.5	-110.3±1.1	-152.5±18.5	-100.9±2.8	-70.8±29.1	-109.3±3.1	**-0.3±0.1**
	Walker_2l	-21.6	-27.9±12.4	-34.3±10.3	-28.7±14.0	16.8±39.1	-21.2±8.5	**105.8±20.8**
Medium Quality	HalfCheetah_2l	41.7	45.3±5.4	-95.3±45.6	64.9±15.0	20.3±34.7	50.4±18.9	**164.1±13.0**
	Walker_2l	71.6	80.3±19.9	-11.7±5.7	87.0±17.1	41.0±21.9	75.8±12.5	**167.1±36.6**

Table 1: The mean and variance of the episodic return on agent-wise imbalanced multi-agent datasets of various maps.

(Veličković et al. 2018), as

$$w_{i,j}^{t,k} = \frac{\exp(\text{LeakyReLU}(W_2^T[W_1 o_i^{t,k}; W_1 o_j^{t,k}]))}{\sum_{k\in N}\exp(\text{LeakyReLU}(W_2^T[W_1 o_i^{t,k}; W_1 o_k^{t,k}]))}$$
$$e_{i,\text{global}}^{t,k} = \textstyle\sum_{j\in N} w_{i,j}^{t,k} W_1 o_j^{t,k}, \tag{8}$$

where W_1 and W_2 represent the learnable weights in GAT. $(\cdot)^T$ represents transposition. $[\cdot;\cdot]$ represents concatenation operation. Then, the centralized critic of each agent i can be expressed as $Q_i(\tau_i^{t,k}, a_i^{t,k}, \boldsymbol{o}_{-\boldsymbol{i}}^{t,k}) = f_{\text{agg}}([e_{i,\text{local}}^{t,k}; e_{i,\text{global}}^{t,k}])$, where f_{agg} represents the aggregation network with two MLPs. In order to simplify the expression, we denote the critic of agent i as Q_i in our subsequent description.

To alleviate the severe extrapolation error in offline agent learning, we plug the filtering mechanism of CRR (Wang et al. 2020c) into individual policy learning. This method can implicitly constrain the forward KL divergence between the learning policy and the behavior policy, which is widely used in offline single-agent learning (Wang et al. 2020c; Nair et al. 2020; Gulcehre et al. 2021) and multi-agent learning (Yang et al. 2021). Formally, suppose that the type of agent i is J, and its corresponding DPER is denoted as $\mathbb{B}_J$. All agent IDs belonging to type J are denoted as $j \in \mathbb{J}$. The priority-based sampling strategy in this dataset is denoted as P_J. When the data at timestep t of episode k is sampled, the actor π_i w.r.t. parameters θ_i and critic Q_i w.r.t. parameters ϕ_i of agent i is trained as follows

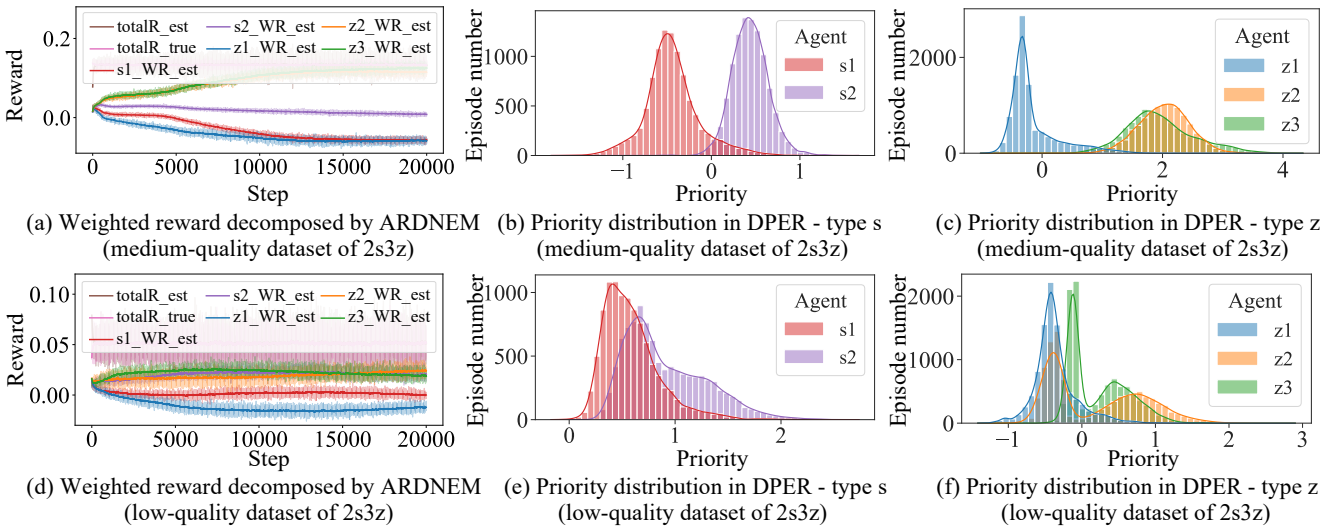
$$\mathcal{L}_{\text{critic}}(\phi_i) = \mathbb{E}_{P_J(\mathbb{B}_J)}\left[\frac{\eta}{\hat{u}_{i(j)}^{t,k}}\left(\hat{r}_{i(j)}^{t,k} + \gamma Q_i' - Q_i\right)^2\right] \tag{9}$$

$$\mathcal{L}_{\text{actor}}(\theta_i) = \mathbb{E}_{P_J(\mathbb{B}_J)}\left[-\frac{\eta}{\hat{u}_{i(j)}^{t,k}}\frac{e^{Q_i/\beta}}{Z}\, Q_i|_{a=a_{i(j)}^{t,k}}\right], \tag{10}$$

where $(\cdot)_{i(j)}^{(\cdot)}$ indicates that although the original data is sampled from the trajectory of agent j, it is used for training the network of agent i. Q_i' represents target critic. $e^{Q_i/\beta}/Z$ is the filtering trick in CRR, where Z is the normalization coefficient within a mini-batch and β is used to control how conservative the policy update is. The uncertainty $1/\hat{u}_{(\cdot)}^{t,k}$ indicates that policy learning should value individual rewards $\hat{r}_{i(j)}^{t,k}$ that are precisely estimated, since a small $\hat{u}_{(\cdot)}^{t,k}$ means that the reward network has high confidence for the corresponding predicted reward. η is used to control the importance weight of the uncertainty on actor-critic learning.

Experiments

In this section, we first introduce the data generation method for agent-wise imbalanced multi-agent datasets in StarCraft II (Samvelyan et al. 2019), multi-agent particle environment (MPE) (Lowe et al. 2017) and multi-agent mujoco (MAmujoco) (Peng et al. 2021). Then, We evaluate our method SIT



(a) Weighted reward decomposed by ARDNEM (medium-quality dataset of 2s3z)
(b) Priority distribution in DPER - type s (medium-quality dataset of 2s3z)
(c) Priority distribution in DPER - type z (medium-quality dataset of 2s3z)
(d) Weighted reward decomposed by ARDNEM (low-quality dataset of 2s3z)
(e) Priority distribution in DPER - type s (low-quality dataset of 2s3z)
(f) Priority distribution in DPER - type z (low-quality dataset of 2s3z)

Figure 3: Intermediate results of ARDNEM and DPER modules on 2s3z datasets.

on these datasets. Finally, we conduct analytical experiments to better illustrate the superiority of our method.

Data Generation for Agent-Wise Imbalanced Multi-Agent Datasets

We construct the agent-wise imbalanced multi-agent datasets based on six maps in StarCraft II and three maps in MPE for discrete control, and two maps in MAmujoco for continuous control. In order to obtain diverse behavior policies for generating these imbalanced datasets, we first online train the joint policies based on QMIX (Rashid et al. 2018) (discrete) or FacMAC (Peng et al. 2021) (continuous) in each environment and store them at fixed intervals during the training process. Then, these saved joint policies are deposited into random, medium and expert policy pools according to their episode returns. Since the policy levels among the individual agents during online training are balanced (*e.g.*, the policy level of each individual agent in a medium joint policy is also medium), we can directly sample the required individual behavior policies from different policy pools to generate the agent-wise imbalanced datasets.

For the convenience of expression, each dataset is represented by the type and policy level corresponding to all individual behavior policies. *e.g.*, in 3s_vs_5z map of StarCraft II, the agent-wise imbalanced multi-agent dataset $100\%[\mathrm{s}_1^{\mathrm{r}}, \mathrm{s}_2^{\mathrm{m}}, \mathrm{s}_3^{\mathrm{e}}]$ indicates that the policy levels of three Stalkers (agents) that generate this data are random, medium and expert, respectively. In practice, since we are interested in non-expert data, we generate low-quality and medium-quality agent-wise imbalanced datasets based on the average episode return for all environments

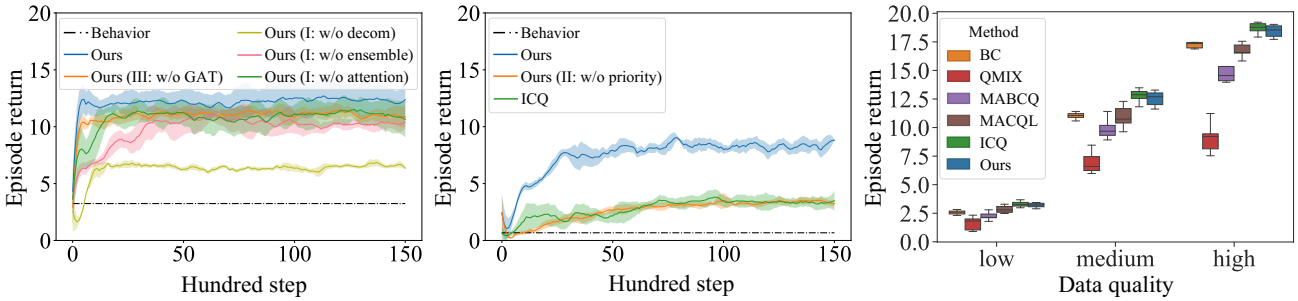
Evaluation on Agent-Wise Imbalanced Multi-Agent Datasets

We compare our proposed method against QMIX (discrete), FacMAC (continuous), behavior cloning (BC), multi-agent version of BCQ (Fujimoto, Meger, and Precup 2019) (MABCQ) and CQL (Kumar et al. 2020) (MACQL), and existing start-of-the-art algorithm ICQ (Yang et al. 2021). The value decomposition structure of MABCQ and MACQL follows Yang et al. (2021). To better demonstrate the effectiveness of our method, we employ the find-tuned hyperparameters provided by the authors of BCQ and CQL.

Table 1 shows the mean and variance of the episodic return of different algorithms with 5 random seeds on tested maps, where the result corresponding to Behavior represents the average episode return of the offline dataset. It can be found that our method significantly outperforms all baselines in all maps and is even 2x higher than the existing start-of-the-art method ICQ in some maps (*e.g.*, low-quality dataset based on 3s_vs_5z in StarCraft II, medium-quality dataset based on CN_3p1p in MPE and all datasets in MAmujoco), which demonstrates that our method can effectively find and exploit good individual trajectories in agent-wise imbalanced multi-agent datasets to boost the overall performance of multi-agent systems. Note that since the online algorithm QMIX or FacMAC cannot handle the extrapolation error in offline scenarios, its performance is much lower than other methods.

A Closer Look at SIT

ARDNEM and DPER are two important modules of our SIT. The former is used to estimate individual rewards, and the latter constructs type-wise prioritized experience replays. To better demonstrate the effectiveness of our method, we investigate two questions: 1) Can the learned ARDNEM correctly decompose the global reward into individual rewards? and 2) Can the priority in DPER accurately reflect the quality of individual trajectories? To answer these two questions, we show some intermediate results of our method on medium-quality and low-quality datasets of 2s3z, where the composition of the former is



(a) Ablation study for stage I and stage III (low-quality dataset of 2s3z)

(b) Ablation study for stage II (extreme low-quality dataset of 2s3z)

(c) Evaluation on agent-wise balanced multi-agent datasets with different data quality (2s3z)

Figure 4: Ablation experiments and the performance on agent-wise balanced datasets.

$100\%[\mathrm{s}_1^{\mathrm{r}}, \mathrm{s}_2^{\mathrm{e}}, \mathrm{z}_1^{\mathrm{r}}, \mathrm{z}_2^{\mathrm{e}}, \mathrm{z}_3^{\mathrm{e}}]$ and the composition of the latter is $50\%[\mathrm{s}_1^{\mathrm{r}}, \mathrm{s}_2^{\mathrm{r}}, \mathrm{z}_1^{\mathrm{r}}, \mathrm{z}_2^{\mathrm{r}}, \mathrm{z}_3^{\mathrm{r}}] + 50\%[\mathrm{s}_1^{\mathrm{r}}, \mathrm{s}_2^{\mathrm{m}}, \mathrm{z}_1^{\mathrm{r}}, \mathrm{z}_2^{\mathrm{m}}, \mathrm{z}_3^{\mathrm{m}}]$.

Figure 3(a) illustrates the decomposed weighted reward $\hat{r}$, estimated by ARDNEM (denoted as WR in the legend) during training on the medium-quality dataset. We can observe that s_1's reward (red) is lower than s_2's reward (violet), and z_1's reward (blue) is lower than z_2's reward (orange) and z_3's reward (green). This decomposition result of ARDNEM agrees with our expectation because all individual trajectories of s_1 and z_1 are generated by random behavior policies, while the individual trajectories of other agents are generated by expert behavior policies. Figure 3(d) illustrates the decomposed weighted reward on the low-quality dataset. Similarly, the comparisons also agree with the intended decomposition we desire to achieve.

Since 2s3z map has two types of agents, we store the individual trajectories of each agent into the corresponding prioritized experience replays by their type. For the medium-quality dataset, Figure 3(b) and Figure 3(c) show the priority distribution of all individual trajectories in each prioritized experience replay. It demonstrates that most individual trajectories of s_1 have lower priority than s_2, while z_1 has lower priority than z_2 and z_3. This comparison is fully consistent with the quality of the individual trajectories, which indicates the priorities in DPER is as intended. Figure 3(e) and Figure 3(f) draw similar conclusions on the low-quality dataset. A deeper look into Figure 3(f) finds that the priority distribution has multiple modals, which indicates that it correctly captures the distribution of the quality even when the variance of the quality is large. *e.g.*, 50% $\mathrm{z}_2^{\mathrm{r}}$ + 50% $\mathrm{z}_2^{\mathrm{m}}$ (orange).

Analytical Experiment

We conduct some analytical experiments to better understand our proposed method. All experiments are evaluated on the low-quality dataset of 2s3z unless otherwise stated.

Ablation study. Figure 4(a) illustrates the ablation study about stage I and stage III, where '(w/o decom)' represents that the critic of each agent is directly trained based on the total reward without decomposition. '(w/o attention)' represents that the weight λ of each estimated reward is always equal to 1 instead of a learnable value. '(w/o ensemble)' removes the ensemble mechanism. '(w/o GAT)' removes the GAT aggregation. The result shows that each part is important in our SIT. To better illustrate the role of priority sampling in stage II, we construct an extreme low-quality dataset based on 2s3z, *i.e.*, $99.5\%[\mathrm{s}_1^{\mathrm{r}}, \mathrm{s}_2^{\mathrm{r}}, \mathrm{z}_1^{\mathrm{r}}, \mathrm{z}_2^{\mathrm{r}}, \mathrm{z}_3^{\mathrm{r}}] + 0.5\%[\mathrm{s}_1^{\mathrm{r}}, \mathrm{s}_2^{\mathrm{m}}, \mathrm{z}_1^{\mathrm{r}}, \mathrm{z}_2^{\mathrm{m}}, \mathrm{z}_3^{\mathrm{m}}]$. The result in Figure 4(b) shows that the priority sampling in stage II plays a key role when the good individual trajectories are sparse in the dataset.

Evaluation on agent-wise balanced multi-agent datasets. To evaluate the performance of our method on agent-wise balanced multi-agent datasets, we generate datasets with random, medium and expert quality on the 2s3z for evaluation. Figure 4(c) shows our method can achieve similar performance to ICQ on these datasets and is significantly higher than other baselines, which indicates that our method can still work well in agent-wise balanced datasets.

Computational complexity. According to the experiments, the running time of Stage I (supervised learning) is only 10% of ICQ as it does not contain some complex calculation operations (*e.g.* CRR trick) during gradient backpropagation. The running time of Stage II is negligible (about 10s) as it only needs the inference of the ARDNEM. The running time of Stage III is similar to ICQ.

Conclusion

In this paper, we investigate the diversity of individual trajectories in offline multi-agent datasets and empirically show the current offline algorithms cannot fully exploit the useful knowledge in complex and mixed datasets. *e.g.*, agent-wise imbalanced multi-agent datasets. To address this problem, we propose a novel algorithmic framework called Shared Individual Trajectories (SIT). It can effectively decompose the joint trajectories into individual trajectories, so that the good individual trajectories can be shared among agents to boost the overall performance of multi-agent systems. Extensive experiments on both discrete control (*i.e.*, StarCraft II and MPE) and continuous control (*i.e.*, MAmujoco) demonstrate the effectiveness and superiority of our method in complex and mixed datasets.

Acknowledgments

This work was supported in part by National Key Research and Development Program of China (2022YFC3340900), National Natural Science Foundation of China (No. 62037001, U20A20387, No. 62006207), Young Elite Scientists Sponsorship Program by CAST(2021QNRC001), Project by Shanghai AI Laboratory (P22KS00111), the StarryNight Science Fund of Zhejiang University Shanghai Institute for Advanced Study (SN-ZJU-SIAS-0010), Natural Science Foundation of Zhejiang Province (LQ21F020020), Fundamental Research Funds for the Central Universities (226-2022-00142, 226-2022-00051). Baoxiang Wang is partially supported by National Natural Science Foundation of China (62106213, 72150002) and Shenzhen Science and Technology Program (RCBS20210609104356063, JCYJ20210324120011032).

References

Bazzan, A. L. 2009. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18(3): 342–375.

Chang, Y.-H.; Ho, T.; and Kaelbling, L. 2003. All learning is local: Multi-agent learning in global reward games. In *NeurIPS*.

Christianos, F.; Schäfer, L.; and Albrecht, S. 2020. Shared experience actor-critic for multi-agent reinforcement learning. In *NeurIPS*.

Chua, K.; Calandra, R.; McAllister, R.; and Levine, S. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*.

Foerster, J.; Nardelli, N.; Farquhar, G.; Afouras, T.; Torr, P. H.; Kohli, P.; and Whiteson, S. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *ICML*.

Fujimoto, S.; and Gu, S. S. 2021. A minimalist approach to offline reinforcement learning. In *NeurIPS*.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-policy deep reinforcement learning without exploration. In *ICML*.

Garcıa, J. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1): 1437–1480.

Gronauer, S. 2022. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2): 895–943.

Gulcehre, C.; Colmenarejo, S. G.; Wang, Z.; Sygnowski, J.; Paine, T.; Zolna, K.; Chen, Y.; Hoffman, M.; Pascanu, R.; and de Freitas, N. 2021. Regularized behavior value estimation. In *arXiv preprint arXiv:2103.09575*.

Hüttenrauch, M.; Adrian, S.; Neumann, G.; et al. 2019. Deep reinforcement learning for swarm systems. *Journal of Machine Learning Research*, 20(54): 1–31.

Jiang, J.; and Lu, Z. 2021a. Offline decentralized multi-agent reinforcement learning. In *arXiv preprint arXiv:2108.01832*.

Jiang, J.; and Lu, Z. 2021b. Online Tuning for Offline Decentralized Multi-Agent Reinforcement Learning. In *openreview*.

Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; and Levine, S. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. In *NeurIPS*.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. In *NeurIPS*.

Li, J.; Kuang, K.; Wang, B.; Liu, F.; Chen, L.; Wu, F.; and Xiao, J. 2021. Shapley counterfactual credits for multi-agent reinforcement learning. In *SIGKDD*.

Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3): 293–321.

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

Meng, L.; Wen, M.; Yang, Y.; Le, C.; Li, X.; Zhang, W.; Wen, Y.; Zhang, H.; Wang, J.; and Xu, B. 2021. Offline Pre-trained Multi-Agent Decision Transformer: One Big Sequence Model Conquers All StarCraftII Tasks. In *arXiv preprint arXiv:2112.02845*.

Nair, A.; Gupta, A.; Dalal, M.; and Levine, S. 2020. Awac: Accelerating online reinforcement learning with offline datasets. In *arXiv preprint arXiv:2006.09359*.

Oh, J.; Chockalingam, V.; Lee, H.; et al. 2016. Control of memory, active perception, and action in minecraft. In *ICML*.

Oh, Y.; Lee, K.; Shin, J.; Yang, E.; and Hwang, S. J. 2020. Learning to Sample with Local and Global Contexts in Experience Replay Buffer. In *ICLR*.

Oliehoek, F. A.; and Amato, C. 2016. *A concise introduction to decentralized POMDPs*. Springer.

Omidshafiei, S.; Pazis, J.; Amato, C.; How, J. P.; and Vian, J. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *ICML*.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep exploration via bootstrapped DQN. In *NeurIPS*.

Palmer, G.; Tuyls, K.; Bloembergen, D.; and Savani, R. 2018. Lenient Multi-Agent Deep Reinforcement Learning. In *AAMAS*.

Peng, B.; Rashid, T.; Schroeder de Witt, C.; Kamienny, P.-A.; Torr, P.; Böhmer, W.; and Whiteson, S. 2021. Facmac: Factored multi-agent centralised policy gradients. In *NeurIPS*.

Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*.

Samvelyan, M.; Rashid, T.; Schroeder de Witt, C.; Farquhar, G.; Nardelli, N.; Rudner, T. G.; Hung, C.-M.; Torr, P. H.; Foerster, J.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge. In *AAMAS*.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2016. Prioritized Experience Replay. In *ICLR*.

Shalev-Shwartz, S.; Shammah, S.; and Shashua, A. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. In *arXiv preprint arXiv:1610.03295*.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V. F.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *AAMAS*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Wang, H.-n.; Liu, N.; Zhang, Y.-y.; Feng, D.-w.; Huang, F.; Li, D.-s.; and Zhang, Y.-m. 2020a. Deep reinforcement learning: a survey. *Frontiers of Information Technology & Electronic Engineering*.

Wang, W.; Yang, T.; Liu, Y.; Hao, J.; Hao, X.; Hu, Y.; Chen, Y.; Fan, C.; and Gao, Y. 2020b. Action semantics network: Considering the effects of actions in multiagent systems. In *ICLR*.

Wang, Z.; Novikov, A.; Zolna, K.; Merel, J. S.; Springenberg, J. T.; Reed, S. E.; Shahriari, B.; Siegel, N.; Gulcehre, C.; Heess, N.; et al. 2020c. Critic regularized regression. In *NeurIPS*.

Wu, Y.; Tucker, G.; and Nachum, O. 2019. Behavior regularized offline reinforcement learning. In *arXiv preprint arXiv:1911.11361*.

Yang, Y.; Hao, J.; Chen, G.; Tang, H.; Chen, Y.; Hu, Y.; Fan, C.; and Wei, Z. 2020a. Q-value path decomposition for deep multiagent reinforcement learning. In *ICML*.

Yang, Y.; Hao, J.; Liao, B.; Shao, K.; Chen, G.; Liu, W.; and Tang, H. 2020b. Qatten: A general framework for cooperative multiagent reinforcement learning. In *arXiv preprint arXiv:2002.03939*.

Yang, Y.; Ma, X.; Chenghao, L.; Zheng, Z.; Zhang, Q.; Huang, G.; Yang, J.; and Zhao, Q. 2021. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. In *NeurIPS*.

Zha, D.; Lai, K.-H.; Zhou, K.; and Hu, X. 2019. Experience Replay Optimization. In *IJCAI*.