Difference Advantage Estimation for Multi-Agent Policy Gradients

Yueheng Li [1] **Guangming Xie** [1 2] **Zongqing Lu** [3 2]

Abstract

Multi-agent policy gradient methods in centralized training with decentralized execution recently witnessed many progresses. During centralized training, multi-agent credit assignment is crucial, which can substantially promote learning performance. However, explicit multi-agent credit assignment in multi-agent policy gradient methods still receives less attention. In this paper, we investigate multi-agent credit assignment induced by reward shaping and provide a theoretical understanding in terms of its credit assignment and policy bias. Based on this, we propose an exponentially weighted advantage estimator, which is analogous to GAE, to enable multi-agent credit assignment while allowing the tradeoff with policy bias. Empirical results show that our approach can successfully perform effective multi-agent credit assignment, and thus substantially outperforms other advantage estimators.

1. Introduction

Many real-world problems can be naturally formulated as cooperative multi-agent reinforcement learning (MARL), where agents learn to maximize the expected return shared by all agents. Cooperative MARL has recently witnessed great promise for solving various tasks, such as autonomous driving (Zhou et al., 2020), traffic signal control (Xu et al., 2021), and inventory management (Anonymous, 2022).

A popular learning paradigm in cooperative MARL is Centralized Training with Decentralized Execution (CTDE) (Oliehoek et al., 2008; Kraemer & Banerjee, 2016). To resolve the non-stationarity in multi-agent settings, CTDE allows each agent to take into account the global state and other agents' actions during centralized training, while learning its individual policy conditioned on only local information. Therefore, the question of how to best exploit the opportunity of centralized training is important and still remains open.

In centralized training, an important concept is called *multi-agent credit assignment* (Chang et al., 2003). Unlike the temporal credit assignment problem in single-agent RL (Sutton & Barto, 2018), multi-agent credit assignment describes the difficulty for each agent to deduce its contribution to the team through shared rewards. Although it is sometimes possible to design individual rewards, these rewards are rarely available in cooperative settings and frequently fail to motivate individuals to sacrifice for the larger good (Foerster et al., 2018). Without or with unsuitable credit assignment, learning in challenging cooperative multi-agent tasks will be substantially impeded and thus lead to poor performance.

On the contrary, as many previous studies pointed out (Tumer & Agogino, 2007; Proper & Tumer, 2012; Wang et al., 2020a), effective multi-agent credit assignment may significantly benefit policy optimization and improve learning performance. However, multi-agent credit assignment still receives much less attention from the community and is rarely discussed explicitly. Among existing cooperative MARL approaches, value-based methods (Sunehag et al., 2018; Rashid et al., 2018; Son et al., 2019; Wang et al., 2021) perform credit assignment through value function factorization, which corresponds to an implicit reward redistribution among agents (Sunehag et al., 2018). However, theoretical understanding of the underlying credit assignment of these approaches is still insufficient (Wang et al., 2020a). On the other hand, besides using value decomposition (Wang et al., 2020b; Zhang et al., 2021), policy-based methods perform credit assignment usually by counterfactual advantage, as proposed in COMA (Foerster et al., 2018). However, empirical results have shown large performance gap between COMA and state-of-the-art methods in challenging tasks such as StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019). This might be because its counterfactual advantage suffers high variance, as some previous studies pointed out (Wang et al., 2020b; Papoudakis et al., 2021).

Recently, MAPPO (Yu et al., 2021), a multi-agent variant of the popular on-policy reinforcement learning algo-

[1]College of Engineering, [2]Institute of Artificial Intelligence, [3]School of Computer Science, Peking University. Correspondence to: Guangming Xie <xiegming@pku.edu.cn>, Zongqing Lu <zongqing.lu@pku.edu.cn>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

rithm PPO (Schulman et al., 2017), achieves strong results compared with state-of-the-art off-policy methods. However, during centralized training, it only utilizes the centralized value function with Generalized Advantage Estimator (GAE) (Schulman et al., 2016) for policy evaluation. This will generally lead to identical advantages for each agent through shared rewards and global information, in which the agents cannot determine their individual contribution to the team's performance.

In this paper, we investigate *explicit* multi-agent credit assignment for multi-agent policy gradient methods by reward shaping. First, we introduce policy-invariance reward shaping by integrating potential-based reward shaping (Ng et al., 1999) and difference rewards, *i.e.*, potential-based difference rewards. Then, we analyze the benefits and potential drawbacks of this reward shaping method and its variants, which indicates a tradeoff between credit assignment and policy bias. Inspired by TD(λ) (Sutton & Barto, 2018) and GAE, we propose an advantage estimator through an exponentially weighted sum of potential-based difference rewards to enable multi-agent credit assignment while allowing the tradeoff with policy bias. We call this estimation scheme *Difference Advantage Estimator* (DAE). Extensive empirical results on matrix game, Multi-Agent Particle Environment (MPE) (Lowe et al., 2017) and SMAC tasks show that DAE can successfully perform effective multi-agent credit assignment, and thus substantially outperforms other advantage estimators including GAE and counterfactual advantage.

Our main contributions, among others, are:

- We provide a theoretical understanding of potential-based difference rewards in terms of policy invariance and N-step multi-agent credit assignment. The latter one directly motivates our method.
- We propose DAE, an advantage estimator, that can easily tradeoff credit assignment and policy bias by a single parameter and further allow the bias-variance tradeoff when combined with GAE.
- We empirically show that DAE outperforms other advantage estimators in a variety of tasks, which immediately indicates its generality for multi-agent policy gradients.

2. Background

2.1. Model

We model the cooperative multi-agent task as a decentralized partially observable MDP (Dec-POMDP), defined by a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, r, \mathcal{P}, \mathcal{O}, \mathcal{Z}, \gamma \rangle$. At each timestep, each agent $i \in \mathcal{N}$ receives a partial observation $o_i \in \mathcal{Z}$ according to $\mathcal{O}(s; i)$ at the state $s \in \mathcal{S}$. Then, agent i chooses an action $a_i \in \mathcal{A}$ according to its policy $\pi_i(a_i|o_i)$. The actions of all agents form a joint action $\boldsymbol{a}$. The state s transitions to next state s' according to $\mathcal{P}(s'|s, \boldsymbol{a})$ and all agents receive a shared reward $r(s, \boldsymbol{a})$. The objective of all agents is to maximize the cumulative return $\mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=0}^{\infty} \gamma^t r_t]$ under the joint policy $\boldsymbol{\pi}$ which is the product of each π_i, where $\gamma \in [0, 1)$ is the discount factor. To settle partial observability, history $\tau_i \in \mathcal{T}_i = (\mathcal{Z} \times \mathcal{A})^*$ is often used to learn the policy $\pi(\cdot|\tau_i)$ instead of observation o_i. Further, for *notation simplicity*, we use $\pi_i(\cdot|s)$ instead. The action-value function and state-value function of the joint policy $\boldsymbol{\pi}$ are defined as:

$$Q^{\boldsymbol{\pi}}(s, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{a}_t) | s_0 = s, \boldsymbol{a}_0 = \boldsymbol{a}] \quad (1)$$

$$V^{\boldsymbol{\pi}}(s) = \mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{a}_t) | s_0 = s]. \quad (2)$$

We consider actor-critic in the CTDE paradigm as the learning framework, where a centralized critic is learned with global information during centralized training, while each agent learns a decentralized policy based on its local information for execution. Following the setting of existing work (Foerster et al., 2018; Yu et al., 2021), all agents share the critic. Further, let θ denote the parameters of all agents' policies that can be shared among agents or not, which does not affect the conclusions made in this paper.

2.2. Multi-Agent Actor-Critic

Multi-agent actor-critic methods usually learn a centralized Q-function or V-function as the critic, for which COMA (Foerster et al., 2018) and MAPPO (Yu et al., 2021) are respectively the representative method.

COMA learns stochastic policies using the following policy gradients:

$$g = \mathbb{E}_{\boldsymbol{\pi}}\big[\sum_i \nabla_\theta \log \pi_i(a_i|s) A_i(s, \boldsymbol{a})\big], \quad (3)$$

where $A_i(s, \boldsymbol{a}) = Q(s, \boldsymbol{a}) - \mathbb{E}_{a_i \sim \pi_i(\cdot|s)}[Q(s, a_i, a_{-i})]$ is the counterfactual advantage and a_{-i} denotes the joint action of all agents except i. The counterfactual baseline is subtracted from $Q(s, \boldsymbol{a})$ to enable multi-agent credit assignment and reduce variance.

The stochastic policies in MAPPO are trained to maximize the objective:

$$L(\theta) = \mathbb{E}_{\boldsymbol{\pi}}\big[\sum_i \min(\rho_i A_i, \operatorname{clip}(\rho_i, 1-\epsilon, 1+\epsilon) A_i)\big], \quad (4)$$

where $\rho_i = \frac{\pi_i(a_i|s)}{\pi_i^{\text{old}}(a_i|s)}$ is the individual importance ratio and A_i is estimated through GAE (Schulman et al., 2016),

$$A_{i,t}^{\text{GAE}(\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{i,t+l}, \quad (5)$$

where $\delta_{i,t+l} = r_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l})$ is the TD residual of $V(s)$ (Sutton & Barto, 2018). GAE is an efficient advantage estimator in single-agent RL that enables bias-variance tradeoff via parameter λ. However, with a shared value function, the advantage estimated by GAE is identical for all agents thereby hard to quantify the contribution of each agent. A simple way to enable multi-agent credit assignment is to substitute the advantage estimated by GAE by the counterfactual baseline. However, this leads to poor performance, which we will discuss in Section 4.3.

Another kind of multi-agent actor-critic methods (Wang et al., 2020b; Zhang et al., 2021; Su et al., 2021; de Witt et al., 2020; Su & Lu, 2021) utilizes value factorization to effectively learn the value function and implicitly performs credit assignment. Previous work (Wang et al., 2020a) analyzes linear factorization implies counterfactual credit assignment. However, more general theoretical understanding of the underlying credit assignment of value decomposition methods is still insufficient.

2.3. Reward Shaping

Reward shaping is a technique for improving a reinforcement learner's performance by integrating expert knowledge into MDP through shaping rewards (Gullapalli & Barto, 1992). It has been popular in single-agent RL, and very recently adopted to the MARL setting as well (Devlin et al., 2011; Xiao et al., 2021).

In single-agent RL, an important principle of reward shaping is *policy invariance* (Ng et al., 1999), which means the reward shaping should not make the agent deviate from the true goal and keeps the optimal policy unchanged. Ng et al. (1999) presented a potential-based reward shaping method to deal with the *temporal* credit assignment problem while preserving policy invariance. The reward after potential-based reward shaping is given by $\tilde{r} = r + f$, where the shaping function f can be written as follows:

$$f(s', s) = \gamma\phi(s') - \phi(s), \tag{6}$$

where $\phi : \mathcal{S} \to \mathbb{R}$ is a real-valued shaping function.

Another type of potential-based reward shaping is called shaping advice (Wiewiora et al., 2003), where the potential function depends not only on states but also on actions to guide the agent with more specific information. One form of shaping advice is called look-ahead advice, given by:

$$f(s', a', s, a) = \gamma\phi(s', a') - \phi(s, a). \tag{7}$$

However, to guarantee policy invariance, look-ahead advice needs a biased greedy action selection (Wiewiora et al., 2003).

Although these approaches of reward shaping can be applied in multi-agent settings, they have two major limitations (Subramanian et al., 2021). The first is that the reward shaping must be done by an expert who has complete domain knowledge. It is not always possible to find such experts for complex MARL tasks. Second, the reward shaping is usually given by a fixed function. But better advice should be adaptive during training since other agents' behaviors are also changing.

In MARL, reward shaping can additionally induce multi-agent credit assignment. Difference rewards (Wolpert & Tumer, 2002; Proper & Tumer, 2012) are a powerful way to perform multi-agent credit assignment, where each agent can deduce its contribution to the team by capturing the difference between the shared reward and the reward received if its action is replaced with a default action. Formally, the difference rewards for each agent are given by:

$$\tilde{r}_i(s, \boldsymbol{a}) = r(s, \boldsymbol{a}) - r(s, a_{-i}, c_i), \tag{8}$$

where c_i is a default action for agent i to replace a_i. Considering that it is unclear how to choose the default action, an alternative form of difference rewards has been proposed by using aristocrat utility (Wolpert & Tumer, 2002),

$$\tilde{r}_i(s, \boldsymbol{a}) = r(s, \boldsymbol{a}) - \mathbb{E}_{a_i\sim\pi_i}[r(s, a_{-i}, a_i)]. \tag{9}$$

Clearly, (9) can quantify the contribution of each agent given the state and joint action. However, directly applying difference rewards for each agent does not guarantee policy invariance, which may severely bias agents from the original objective. We will discuss this further in the next section.

3. Method

In this section, we first introduce reward shaping with policy invariance. We then analyze its induced multi-agent credit assignment. After that, we develop an advantage estimator based on the reward shaping, called **D**ifference **A**dvantage **E**stimation (**DAE**), which allows the tradeoff between credit assignment and policy bias.

3.1. Potential-Based Difference Rewards

As previously discussed, multi-agent credit assignment is of great significance to improve learning efficiency and performance; difference rewards capture the contribution of each agent but make agents deviate from the original goal, while potential-based reward shaping indeed keeps policy invariance. Therefore, we consider *potential-based difference rewards*, which are a direct combination of potential-based reward shaping and difference rewards,

$$\begin{aligned}\tilde{r}_i(s, \boldsymbol{a}) = &\underbrace{r(s, \boldsymbol{a})}_{①} + \underbrace{\gamma\mathbb{E}_{a_i'\sim\pi_i}[r(s', a_{-i}', a_i')]}_{②} \\ &\underbrace{-\mathbb{E}_{a_i\sim\pi_i}[r(s, a_{-i}, a_i)]}_{③}.\end{aligned} \tag{10}$$

Unlike the potential based difference rewards in CaP (Devlin et al., 2014), the potential function (②+③) in (10) is related to the actions of other agents, which actually belongs to look-ahead shaping advice. Previous study (Xiao et al., 2021) proves that, with an adjustment in the policy gradient, applying look-ahead shaping advice in multi-agent actor-critic can guarantee the learned policies to be locally optimal in the original MDP. We adopt difference rewards (① + ③) as shaping advice, and the potential function (② + ③) is irrelevant to the action of agent i. Therefore, it can be proved to guarantee policy invariance *without* any adjustments. The following proposition formally shows that our potential-based difference rewards guarantee policy invariance in multi-agent actor-critic.

Proposition 3.1 (Policy Invariance). *Given the reward function $r(s,\boldsymbol{a})$, the policy gradient keeps invariant for each agent using the potential-based difference rewards defined in* (10)*.*

Proof. Let $\phi_i(s,a^{-i}) \doteq \mathbb{E}_{a_i\sim\pi_i}[r(s,a_{-i},a_i)]$ and $Q(s,\boldsymbol{a})$ be the Q-function of the joint policy $\boldsymbol{\pi}$ in the original MDP. For each agent i, the Q-function of the joint policy $\boldsymbol{\pi}$ in the MDP with the shaped reward function is denoted as $\tilde{Q}_i(s,\boldsymbol{a})$, which differs from $Q(s,\boldsymbol{a})$ by the potential function:

$$\begin{aligned}\tilde{Q}_i(s,\boldsymbol{a}) &= \mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=0}^{\infty}\gamma^t(r_t+\gamma\phi_i(s_{t+1},a_{t+1}^{-i})-\phi_i(s_t,a_t^{-i}))]\\ &= \mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=0}^{\infty}\gamma^t r_t] + \mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=1}^{\infty}\gamma^t\phi_i(s_t,a_t^{-i})]\\ &\quad - \mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=0}^{\infty}\gamma^t\phi_i(s_t,a_t^{-i})]\\ &= Q(s,\boldsymbol{a}) - \phi_i(s,a^{-i}).\end{aligned}$$

Then, the policy gradient after reward shaping is given by

$$g = \mathbb{E}_{\boldsymbol{\pi}}[\sum_i \nabla_\theta \log\pi_i(a_i|s)\tilde{Q}_i(s,\boldsymbol{a})], \tag{11}$$

where the expectation is with respect to the state-action distribution induced by the joint policy $\boldsymbol{\pi}$. Let $d^{\boldsymbol{\pi}}(s)$ be the stationary distribution of states, then we have:

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{\pi}}[\sum_i \nabla_\theta\log\pi_i(a_i|s)\phi(s,a^{-i})]\\ &= \sum_s d^{\boldsymbol{\pi}}(s)\sum_{\boldsymbol{a}}\boldsymbol{\pi}(\boldsymbol{a}|s)\sum_i\nabla_\theta\log\pi_i(a_i|s)\phi_i(s,a^{-i})\\ &= \sum_s d^{\boldsymbol{\pi}}(s)\sum_i\sum_{a_{-i}}\pi_{-i}(a_{-i}|s)\sum_{a_i}\nabla_\theta\pi_i(a_i|s)\phi_i(s,a^{-i})\\ &= \sum_s d^{\boldsymbol{\pi}}(s)\sum_i\sum_{a_{-i}}\pi_{-i}(a_{-i}|s)\phi_i(s,a^{-i})\nabla_\theta 1 = 0.\end{aligned}$$

We immediately have

$$\begin{aligned}g &= \mathbb{E}_{\boldsymbol{\pi}}[\sum_i \nabla_\theta\log\pi_i(a_i|s)\tilde{Q}_i(s,\boldsymbol{a})]\\ &= \mathbb{E}_{\boldsymbol{\pi}}[\sum_i \nabla_\theta\log\pi_i(a_i|s)Q(s,\boldsymbol{a})].\end{aligned} \tag{12}$$

Therefore, the policy gradient of each agent keeps invariant after reward shaping using (10). □

Unlike single-agent settings, the concept of an optimal policy in multi-agent settings is not clear (Devlin & Kudenko, 2011). The policy invariance here mainly implies the unbiased policy gradient, which is different from the optimal policy keeping invariant in single-agent settings. In the CTDE paradigm, the optimality of the joint policy may not be guaranteed due to the limited class of the product of individual policies (see Appendix A.2 for further details).

In the following subsection, we analyze how the policy-invariant reward shaping in (10) enables multi-agent credit assignment. Based on the analysis, we introduce the tradeoff between policy bias and credit assignment.

3.2. N-Step Multi-Agent Credit Assignment

Difference rewards induce multi-agent credit assignment, but it is unclear if the potential-based difference rewards do the same. In the following, we give a profound understanding of this.

We denote $\mathcal{M}^{(0)}$ as the original MDP with reward $r_{i,t}^{(0)} = r_t$ for each agent i at each timestep t. Then, applying the potential-based difference rewards will result in a transformed MDP $\mathcal{M}^{(1)}$ with the shaped reward $r_{i,t}^{(1)} = r_t + \gamma\mathbb{E}_{a_i}[r_{t+1}] - \mathbb{E}_{a_i}[r_t]$ according to (10), where $\mathbb{E}_{a_i}[r_t]$ denotes $\mathbb{E}_{a_i\sim\pi_i}[r(s_t,a_{-i,t},a_{i,t})]$ for short. For that, we say $\mathcal{M}^{(0)}$ is transformed into $\mathcal{M}^{(1)}$ after a one-step reward shaping. After that, we can perform another step of reward shaping and have

$$\begin{aligned}r_{i,t}^{(2)} &= r_{i,t}^{(1)} + \gamma\mathbb{E}_{a_i}[r_{i,t+1}^{(1)}] - \mathbb{E}_{a_i}[r_{i,t}^{(1)}]\\ &= r_t + \gamma^2\mathbb{E}_{a_i}[r_{t+2}] - \mathbb{E}_{a_i}[r_t],\end{aligned} \tag{13}$$

where $r_{i,t}^{(2)}$ belongs to MDP $\mathcal{M}^{(2)}$. The intuition of repeatedly applying reward shaping is that if the shaped rewards in $\mathcal{M}^{(1)}$ induce better credit assignment without any drawbacks, then further reward shaping in $\mathcal{M}^{(2)}$ should be better than that in $\mathcal{M}^{(1)}$.

Before analyzing its drawbacks, let us first consider the multi-agent credit assignment induced by applying the reward shaping k times, and following (13) we can easily derive:

$$r_{i,t}^{(k)} = r_t + \gamma^k\mathbb{E}_{a_i}[r_{t+k}] - \mathbb{E}_{a_i}[r_t]. \tag{14}$$

Based on (14), we can observe that the difference rewards in (9) exactly match $\mathcal{M}^{(\infty)}$ with $r_{i,t}^{(\infty)} = r_t - \mathbb{E}_{a_i}[r_t]$, since γ^k is zero when $k \to \infty$. This means achieving the same level of credit assignment as difference rewards may need many steps of the reward shaping.

Specifically, the discounted return of each step of the reward shaping gives a more intuitive formulation:

$$\begin{aligned}
G_{i,t}^{(0)} &= \sum_{l=0}^{\infty} \gamma^l r_{t+l} \\
G_{i,t}^{(1)} &= \sum_{l=0}^{\infty} \gamma^l r_{t+l} - \mathbb{E}_{a_i}[r_t] \\
G_{i,t}^{(2)} &= \sum_{l=0}^{\infty} \gamma^l r_{t+l} - \gamma \mathbb{E}_{a_i}[r_{t+1}] - \mathbb{E}_{a_i}[r_t] \\
&\vdots \\
G_{i,t}^{(\infty)} &= \sum_{l=0}^{\infty} \gamma^l r_{t+l} - \sum_{l=0}^{\infty} \gamma^l \mathbb{E}_{a_i}[r_{t+l}],
\end{aligned} \tag{15}$$

where $G_{i,t}^{(k)}$ is the discounted return of $\mathcal{M}^{(k)}$. It is clear that the discounted return considers one more future step if one more step of reward shaping is performed. We can also observe that $G_{i,t}^{(\infty)}$ actually corresponds to the counterfactual advantage of COMA in (3), which involves the credit assignment of all future steps, and Castellini et al. (2020) proposed a policy gradient method using $G_{i,t}^{(\infty)}$.

However, although more steps of the reward shaping enables better credit assignment, it has several drawbacks.

- First, the reward function is usually unknown for MARL tasks, which means without extra simulations, the reward that marginalizes out one agent's action needs to be estimated, such as by a neural network. While $G_{i,t}^{(0)}$ uses unbiased samples to compute the return of a trajectory, the term $\mathbb{E}_{a_i}[r]$ will introduce bias due to the estimation error, and the bias will accumulate over steps of the reward shaping.
- Second, the policy is not invariant for k-step $(k \geq 2)$ reward shaping and generally the difference from the original policy gradient increases as k increases (see Appendix A.3). Therefore, if we conduct too many steps of the reward shaping, the estimation error along with the policy gradient shift may hurt the performance.

We collectively refer to the two drawbacks as ***policy bias***. Here we do not consider its variance. Although the additional estimation error brings variance, it serves as a baseline that reduces variance. Empirically, they have similar variance (Section 4.3), but too many steps of the reward shaping may significantly hurt the performance (Section 4.3).

In the next subsection, we will discuss how to balance the credit assignment and policy bias.

3.3. Difference Advantage Estimation

Inspired by TD(λ) (Sutton & Barto, 2018) and GAE, we introduce a parameter $\beta \in [0, 1]$ and define the β-step reward shaping by exponentially weighted average of the rewards from $\mathcal{M}^{(k)}$ as:

$$\begin{aligned}
r_{i,t}^{(\beta)} &= (1-\beta)(r_{i,t}^{(0)} + \beta r_{i,t}^{(1)} + \beta^2 r_{i,t}^{(2)} + \dots) \\
&= r_t + (1-\beta)\gamma \sum_{l=0}^{\infty} (\gamma\beta)^l \mathbb{E}_{a_i}[r_{t+l+1}] - \mathbb{E}_{a_i}[r_t].
\end{aligned} \tag{16}$$

Although (16) is rather complicated, the corresponding discounted return is quite simple,

$$\begin{aligned}
G_{i,t}^{(\beta)} &= \sum_{l=0}^{\infty} \gamma^l r_{i,t+l}^{(\beta)} \\
&= \sum_{l=0}^{\infty} \gamma^l (r_{t+l} - \beta^{l+1} \mathbb{E}_{a_i}[r_{t+l}]).
\end{aligned} \tag{17}$$

From (17), we can see that compared with the return in the original MDP, the reward r_{t+l} is replaced by $r_{t+l} - \beta^{l+1}\mathbb{E}_{a_i}[r_{t+l}]$. It is analogous to difference rewards in which the reward is subtracted by a reward baseline for each agent that marginalizes out its action. But the baseline here is heavily discounted for future steps, which implies that credit assignment at future steps is less taken into consideration similar to discounted rewards. Therefore, (17) makes a compromise between policy bias and credit assignment, controlled by the parameter β. Larger β considers more credit assignment at the sacrifice of larger policy bias, whereas smaller β obtains less credit assignment but reduces policy bias. There are two notable special cases of this formula: $\beta = 0$ and $\beta = 1$ reflect the original return and the return of difference rewards respectively.

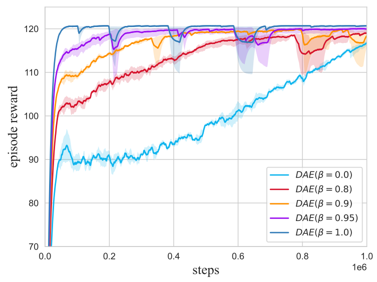
Based on the tradeoff between policy bias and credit assignment, we can build a stochastic gradient ascent algorithm using the gradients estimated by the returns in (17). However, the variance of the gradient estimator scales unfavorably with the time horizon, since the effect of an action is confounded with the effects of past and future actions (Schulman et al., 2016). Using a value function can substantially reduce the variance of policy gradient estimates at the cost of some bias. Therefore, we introduce DAE, an advantage estimator analogous to GAE, which adjusts the bias-variance tradeoff while enabling multi-agent credit assignment. Formally, we define difference advantage estimation as following:

$$A_{i,t}^{\mathrm{DAE}} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{i,t+l}, \tag{18}$$

where $\delta_{i,t+l} = r_{t+l} - \beta^{l+1}\mathbb{E}_{a_i}[r_{t+l}] + \gamma V_{t+l+1} - V_{t+1}$.

$$\|t=1\| \quad \begin{array}{c|c|cc} & & \multicolumn{2}{c}{\text{Agent 2}} \\ & & b_1 & b_2 \\ \hline \text{Agent 1} & a_1 & -\frac{1}{2} & -\frac{1}{2} \\ & a_2 & \frac{1}{2} & \frac{1}{2} \end{array} \quad \Longrightarrow \quad \|t=2\| \quad \begin{array}{c|c|cc} & & \multicolumn{2}{c}{\text{Agent 2}} \\ & & b_1 & b_2 \\ \hline \text{Agent 1} & a_1 & -\frac{1}{2} & \frac{1}{2} \\ & a_2 & -\frac{1}{2} & \frac{1}{2} \end{array}$$

(a) 2-step matrix game



(b) learning curves of DAE with different β

Figure 1. (a) Payoff matrices for 2-step matrix game at $t = 1$ (left) and $t = 2$ (left), where all indices start from 1. (b) Learning curves of DAE with different β on the 16-step matrix game, where larger β gets better performance.

The parameter γ from GAE makes a balance between bias and variance, and the parameter β controls the tradeoff between the policy bias and credit assignment. Similarly, $\beta = 1$ means the original reward is replaced by difference rewards, while DAE degenerates to GAE if $\beta = 0$.

To realize the advantage estimation in (18), besides the value network V_φ parameterized by φ, DAE additionally learns a centralized reward network r_ψ parameterized by ψ to approximate the reward function $r(s, \boldsymbol{a})$. For computational efficiency, the reward network takes as input the state s and a_{-i} and outputs the reward of each action of agent i. Consequently, the expectation of reward can be calculated efficiently by a single forward pass. The reward network is trained by minimizing the MSE regression loss

$$\mathcal{L}(\psi) = \frac{1}{2}(r(s, \boldsymbol{a}) - r_\psi(s, a_{-i}, a_i))^2. \tag{19}$$

There are several advantages for estimating state values and rewards separately rather than Q-values as COMA. First, even though the reward and Q-function have the same dimensionality, learning the reward is easier than Q-function since the regression does not involve many problems like bootstrapping and moving targets (Castellini et al., 2020). Second, as many previous studies (Wang et al., 2020b; Papoudakis et al., 2021) have pointed out, COMA suffers large variance due to the difficulty in learning the Q-values, thus leading to poor performance in SMAC, and the same issue arises with MADDPG (Papoudakis et al., 2021) which also learns the Q-values. Instead, the state-value function has a lower-dimensional input and is generally easier to learn.

Given the fact that the reward signal is usually delayed in many tasks, the reward received currently may be credited to past state-action pairs. A common example is that we usually attach an additional reward at the end of the episode to reflect the win or loss. This additional reward cannot be correctly represented using state-action pair in the final timestep. Therefore, we utilize a recurrent layer in the reward network and train it via Backpropagation Through Time to deal with the delayed reward problem.

For completeness, the training procedure of DAE based on MAPPO is given in Appendix B.

4. Experiments

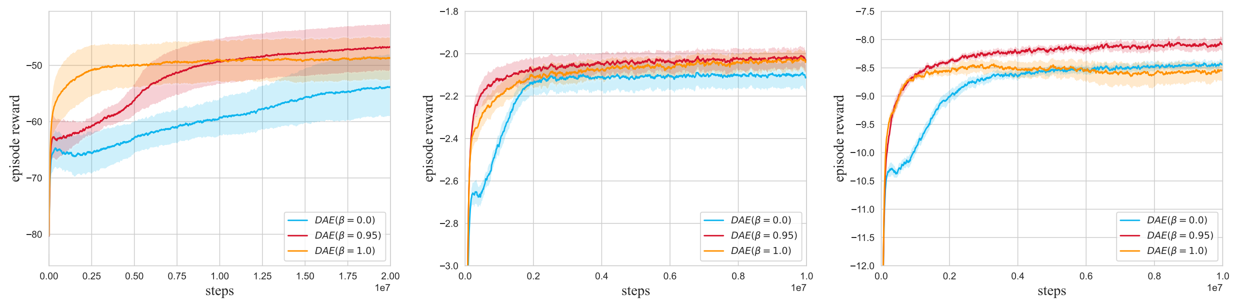
In this section, we perform DAE on a set of experiments, including matrix game, Multi-Agent Particle Environments (MPE) (Lowe et al., 2017) and StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019), to investigate the following questions:

1. *Is credit assignment in DAE helpful to improve the convergence speed and learning performance? How does DAE compare with COMA or directly using difference rewards?*
2. *What is the empirical effect of varying $\beta \in [0, 1]$ when optimizing episodic return using DAE?*

For these experiments, we perform the policy updates using MAPPO (Yu et al., 2021) with parameter sharing. Although the reward network in DAE can be trained with offline samples from the environment, for fair comparison it is trained online using the same epochs and samples as for training the value function. Therefore, in the following experiments, we can compare DAE with GAE and difference rewards by only setting β to 0 and 1 respectively. All the learning curves in the experiments are plotted based on five training runs with different random seeds using mean and standard deviation with confidence internal 95%.

4.1. N-Step Matrix Game

To explicitly investigate multi-agent credit assignment, we design a N-step matrix game. The N-step matrix game has N agents, N actions for each agent, and totally N timesteps. The observation of each agent is a one-hot vector of timestep t, and the shared reward only depends on one agent's action at each timestep t, *i.e.*, $r_t = u_t^{(t)} - \frac{N+1}{2}$, where $u_t^{(t)} = k$ if the t-th agent chooses the k-th action at timestep t. All



(a) cooperative navigation (b) line control (c) formation control

Figure 2. Learning curves for three tasks in MPE using DAE with varying values of β. Each curve is averaged over the settings $N = 3, 4, 5$, and separate results are available in Appendix D.

indices start from 1. Figure 1(a) is an example for $N = 2$ and Figure 1(b) shows the learning curves for $N = 16$ during 1M training steps with different β in DAE. Recall that $\beta = 0$ corresponds to GAE (MAPPO) , and $\beta = 1$ means difference rewards. We can see that in the matrix game larger β has better performance, which implies that credit assignment is important and facilitates the learning.

There are two reasons for this result. First, the agents receive the same reward, and thus without credit assignment all agents obtain the same advantage. This means only one of the agents obtains the right direction of the gradients since only one agent has made a real contribution to the team. Note that the joint action space is $\mathcal{O}(N^N)$ which can be very large, *i.e.*, $N = 16$ in this game. The irrelevant action from the other agent imposes great noise on the policy gradient that will slow the convergence. Instead, if we use difference rewards for each agent, the other irrelevant agent will receive zero reward and thus give a right estimate of the policy gradient. The second reason is that the reward function in the matrix game is rather simple to learn. And the policy actually keeps invariance in this game (see Appendix A.3). This means there is no need to tune β down to reduce policy bias.

4.2. Multi-Agent Particle Environment

We consider three tasks in MPE, including cooperative navigation, formation control and line control (Agarwal et al., 2020). In cooperative navigation task, N agents must cooperate to cover N landmarks while avoiding collusion. In formation control task, N agents are required to position themselves into an N-sided regular polygonal formation, with one landmark at its centre. In line control task, N agents should position themselves equally spread out in a line between the two landmarks. In each task, agents observe the relative positions of other agents and landmarks, and are collectively rewarded based on the relative positions. The global state is formed by concatenating all local observations. For each task, we evaluate on the settings $N = 3, 4, 5$.

Figure 2 shows the performance of different β averaged over the settings $N = 3, 4, 5$ in these three tasks. Separate results on the settings are available in Appendix D. There are two notable aspects to these results. First, $\beta = 1$ and 0.95 generally converge faster than $\beta = 0$. Second, although it may converge quickly in the beginning, the final performance of $\beta = 1$ is usually worse than $\beta = 0.95$ and even $\beta = 0$. The first aspect reveals the same conclusion in the matrix game that the credit assignment induced by DAE can substantially facilitate the learning. We believe the second aspect, which is different from the matrix game, is mainly caused by the policy bias. Recall that the policy bias consists of two parts, the estimation error of the reward function and the policy shifts. The estimation error of the reward mainly depends on how complex the task is. In MPE, the reward is directly related to the observed relative distance of each agent and the dimension of the joint state-action space is low. Therefore, the bias induced by the estimation error may not be enough to impair the performance. This conclusion can also be observed from the result that $\beta = 1$ and 0.95 converge faster. However, the second part of the policy bias actually influences the final performance. Although larger β enables better credit assignment, it also more severely biases the learned policy, making the agent deviate from the original goal. For difference rewards ($\beta = 1$), it is also known as self-consistency problem (Wolpert & Tumer, 2002). Therefore, the performance of $\beta = 1$ is worse than $\beta = 0.95$. The result also empirically shows that DAE can effectively overcome this problem by slightly reducing β.

4.3. StarCraft Multi-Agent Challenge

We evaluate DAE on SMAC in nine maps including three easy maps: 2s3z, 1c3s5z and MMM, three hard maps: 3s5z, 3s_vs_5z and 5m_vs_6m, and three super hard maps: 3s5z_vs_3s6z, MMM2 and corridor. We train the algorithm for 1M, 5M, 10M environment steps and use 15, 10, 5 epochs for easy, hard, super hard maps, respectively. All

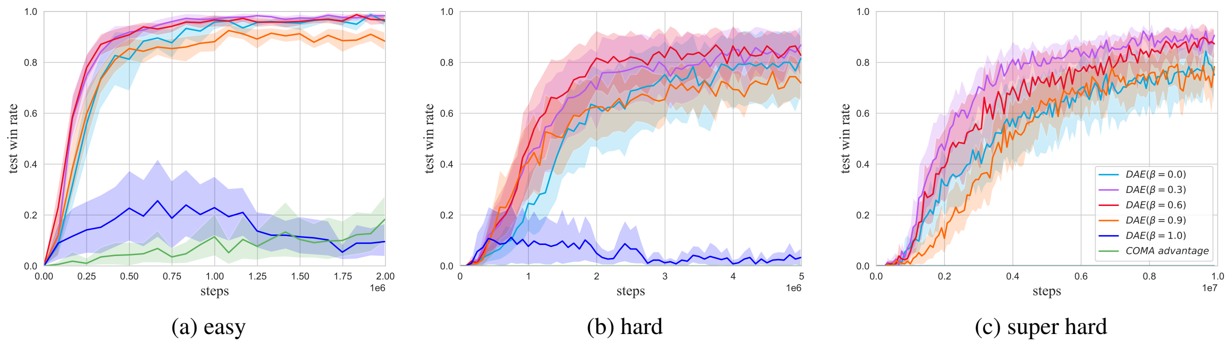


Figure 3. Learning curves in terms of win rates of DAE with different β and COMA advantage in 'easy', 'hard' and 'super hard', each averaged over three corresponding maps. Separate results in each map are available in Appendix D.

other parameters are fixed across the nine maps (see Appendix C). Results are shown in Figure 3, where 'easy' corresponds to the win rate averaged on three easy maps. And the 'hard' and 'super hard' are the same respectively. Separate results in each map are available in Appendix D. Moreover, we also choose COMA advantage as a baseline of advantage estimator on MAPPO.

In general, the results show that DAE with proper β can substantially improve the convergence speed and stability. We can observe that $\beta = 1$ and COMA advantage are very unstable and have poor performance in all the maps. Note that COMA advantage and DAE ($\beta = 1$) are two similar ways realizing difference rewards. We believe the main reason is also similar, *i.e.*, the high estimation error for the joint state-action space results in highly biased advantages. Different from the matrix game and MPE tasks, SMAC has a much higher dimensional joint state-action space which means the reward function is difficult to learn and the Q-function is even harder. As for COMA, although some previous studies (Wang et al., 2020b; Papoudakis et al., 2021) consider its poor performance comes from the large variance of policy gradient, we found that another important factor is that COMA advantage is actually a high-bias advantage estimator (see Appendix D for further details). Compared with $\beta = 1$ which has large policy bias, $\beta = 0.9$ can efficiently discount the bias and significantly improve the performance. Although the high estimation error of the reward mainly affects the performance, the policy shifts can also be observed, *i.e.*, in easy and hard maps, $\beta = 0.9$ converges faster than $\beta = 0$, but its final performance is weaker. Therefore, to reduce the policy bias, we evaluate smaller β (0.3 and 0.6) and they show a better tradeoff between credit assignment and policy bias in this environment. In hard maps, $\beta = 0.6$ is the best, while in super hard maps, $\beta = 0.3$ is the best. The empirical results across all the experiments in Section 4 show that, to enable a good tradeoff between credit assignment and policy bias, β needs to be reduced generally with the difficulty of the task.

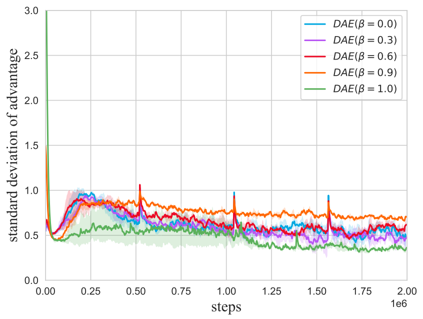


Figure 4. Standard deviation of advantages of DAE with different β on 2s3z in SMAC.

For DAE, the estimated reward introduces additional estimation errors that may cause large variance. However, the term $\mathbb{E}_{a_i}[r_t]$ in (18) also serves as a baseline which will reduce variance. Figure 4 shows the standard deviation of advantages of DAE with different β on map 2s3z. In general, their variances are similar and on the same scale, which implies DAE does not affect the variance much.

5. Conclusion

We propose DAE, an advantage estimator for multi-agent policy gradients, which enables multi-agent credit assignment while allowing the tradeoff with policy bias. The key idea lies in the integration of potential based difference rewards and the exponentially weighted average, analogous to TD(λ). We start from the policy invariance reward shaping (Section 3.1) and extend it to N-step reward shaping (Section 3.2) to perform better credit assignment. To tackle the drawback of policy bias, we introduce a parameter β (Section 3.3) which allows to smoothly interpolate between high bias ($\beta = 1$) and low bias ($\beta = 0$) estimations. Empirical results show that slightly reducing the parameter β can reduce the policy bias while enabling credit assignment, and thus significantly improves the performance in a variety of cooperative multi-agent tasks.

Acknowledgements

We would like to thank the anonymous reviewers for their useful comments to improve our work. This work is supported in part by NSF China under grant 61872009.

References

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 2019.

Agarwal, A., Kumar, S., Sycara, K., and Lewis, M. Learning transferable cooperative behavior in multi-agent teams. In *International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

Anonymous. Multi-agent reinforcement learning with shared resource in inventory management. In *Submitted to The Tenth International Conference on Learning Representations*, 2022.

Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J., and Hochreiter, S. Rudder: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, 2019.

Castellini, J., Oliehoek, F. A., Savani, R., and Whiteson, S. The representational capacity of action-value networks for multi-agent reinforcement learning. In *International Conference on Autonomous Agents and MultiAgent Systems*, 2019.

Castellini, J., Devlin, S., Oliehoek, F. A., and Savani, R. Difference rewards policy gradients. *arXiv preprint arXiv:2012.11258*, 2020.

Chang, Y.-h., Ho, T., and Kaelbling, L. All learning is local: Multi-agent learning in global reward games. In *Advances in Neural Information Processing Systems*, 2003.

de Witt, C. S., Peng, B., Kamienny, P.-A., Torr, P., Böhmer, W., and Whiteson, S. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint arXiv:2003.06709*, 2020.

Devlin, S. and Kudenko, D. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *International Conference on Autonomous Agents and MultiAgent Systems*, 2011.

Devlin, S., Kudenko, D., and Grześ, M. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems*, 14 (02):251–278, 2011.

Devlin, S., Yliniemi, L., Kudenko, D., and Tumer, K. Potential-based difference rewards for multiagent reinforcement learning. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 2014.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*, 2018.

Gullapalli, V. and Barto, A. G. Shaping as a method for accelerating reinforcement learning. In *IEEE international symposium on intelligent control*, 1992.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, 2002.

Kraemer, L. and Banerjee, B. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 2017.

Ng, A. Y., Harada, D., and Russell, S. J. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, 1999.

Oliehoek, F. A., Spaan, M. T., and Vlassis, N. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Papoudakis, G., Christianos, F., Schäfer, L., and Albrecht, S. V. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Advances in Neural Information Processing Systems*, 2021.

Proper, S. and Tumer, K. Modeling difference rewards for multiagent learning. In *International Conference on Autonomous Agents and MultiAgent Systems*, 2012.

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

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. The starcraft multi-agent challenge. In *International Conference on Autonomous Agents and MultiAgent Systems*, 2019.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *International Conference on Learning Representations*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2019.

Su, J., Adams, S., and Beling, P. Value-decomposition multi-agent actor-critics. In *AAAI Conference on Artificial Intelligence*, 2021.

Su, K. and Lu, Z. Divergence-regularized multi-agent actor-critic. *arXiv preprint arXiv:2110.00304*, 2021.

Subramanian, S. G., Taylor, M. E., Larson, K., and Crowley, M. Multi-agent advisor q-learning. *arXiv preprint arXiv:2111.00345*, 2021.

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. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *International Conference on Autonomous Agents and MultiAgent Systems*, 2018.

Sutton, R. S. and Barto, A. G. Reinforcement learning: An introduction, 2018.

Tumer, K. and Agogino, A. Distributed agent-based air traffic flow management. In *International Conference on Autonomous Agents and MultiAgent Systems*, 2007.

Wang, J., Ren, Z., Han, B., Ye, J., and Zhang, C. Towards understanding linear value decomposition in cooperative multi-agent q-learning. *arXiv preprint arXiv:2006.00587*, 2020a.

Wang, J., Ren, Z., Liu, T., Yu, Y., and Zhang, C. Qplex: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2021.

Wang, Y., Han, B., Wang, T., Dong, H., and Zhang, C. Off-policy multi-agent decomposed policy gradients. In *International Conference on Learning Representations*, 2020b.

Wiewiora, E., Cottrell, G. W., and Elkan, C. Principled methods for advising reinforcement learning agents. In *International Conference on Machine Learning*, 2003.

Wolpert, D. H. and Tumer, K. Optimal payoff functions for members of collectives. In *Modeling complexity in economic and social systems*, pp. 355–369. World Scientific, 2002.

Xiao, B., Ramasubramanian, B., and Poovendran, R. Shaping advice in deep multi-agent reinforcement learning. *arXiv preprint arXiv:2103.15941*, 2021.

Xu, B., Wang, Y., Wang, Z., Jia, H., and Lu, Z. Hierarchically and cooperatively learning traffic signal control. In *AAAI Conference on Artificial Intelligence*, 2021.

Xu, T., Li, Z., and Yu, Y. Error bounds of imitating policies and environments. In *Advances in Neural Information Processing Systems*, 2020.

Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

Zhang, T., Li, Y., Wang, C., Xie, G., and Lu, Z. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2021.

Zhou, M., Luo, J., Villela, J., Yang, Y., Rusu, D., Miao, J., Zhang, W., Alban, M., Fadakar, I., Chen, Z., Huang, A. C., Wen, Y., Hassanzadeh, K., Graves, D., Chen, D., Zhu, Z., Nguyen, N. M., Elsayed, M., Shao, K., Ahilan, S., Zhang, B., Wu, J., Fu, Z., Rezaee, K., Yadmellat, P., Rohani, M., Nieves, N. P., Ni, Y., Banijamali, S., Cowen-Rivers, A. I., Tian, Z., Palenicek, D., Bou-Ammar, H., Zhang, H., Liu, W., Hao, J., and Wang, J. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. In *Conference on Robot Learning*, 2020.

A. Mathematical Details

In this section, we start with some useful lemmas in single-agent RL as well as in imitation learning (Xu et al., 2020; Agarwal et al., 2019). Although these lemmas are not for MARL, we can regard them as lemmas for the joint policy and value function in centralized training of MARL. Therefore, most of them are directly applicable in MARL, and the rest which may not be applicable will be discussed in the section A.2. For simplicity, all the notations in the following are in single-agent setting unless otherwise specified.

We assume the reward function is normalized, i.e., $r(s,a) \in [0,1]$, and define $V^\pi(\mu) = \mathbb{E}_{s_0\sim\mu}[V^\pi(s_0)]$ as the expected value under the initial state distribution μ. The objective is to maximize the expected value from the initial state distribution,

$$\max_{\pi_\theta} V^{\pi_\theta}(\mu). \tag{20}$$

We define the discounted state visitation distribution d^π of a policy π as

$$d^\pi_{s_0}(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t Pr(s_t = s|\pi, s_0), \tag{21}$$

and $d^\pi_\mu(s) = \mathbb{E}_{s_0\sim\mu}[d^\pi_{s_0}(s)]$ as the discounted state visitation distribution under initial state distribution μ. The discounted state-action visitation distribution is defined by $\rho^\pi_\mu(s,a) = d^\pi_\mu(s)\pi(a|s)$. Then the policy gradient is given by

$$\begin{aligned}\nabla_\theta V^{\pi_\theta}(\mu) &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_\theta}_\mu}\mathbb{E}_{a\sim\pi_\theta}[\nabla_\theta \log\pi_\theta(a|s)Q^{\pi_\theta}(s,a)]\\ &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_\theta}_\mu}\mathbb{E}_{a\sim\pi_\theta}[\nabla_\theta \log\pi_\theta(a|s)A^{\pi_\theta}(s,a)].\end{aligned} \tag{22}$$

A.1. Lemmas

The following three lemmas (Lemma A.1, A.2, A.3) come from Xu et al. (2020).

Lemma A.1. *For two policies π and π' we have that*

$$D_{TV}(d^\pi_\mu, d^{\pi'}_\mu) \le \frac{\gamma}{1-\gamma}\mathbb{E}_{s\sim d^\pi_\mu}[D_{TV}(\pi(\cdot|s), \pi'(\cdot|s))]. \tag{23}$$

Proof. By definition, we have

$$\begin{aligned}d^\pi_\mu(s) &= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t Pr(s_t = s|\pi,\mu)\\ &= (1-\gamma)(\mathrm{I}-\gamma P_\pi)^{-1}\mu,\end{aligned}$$

where $P_\pi(s'|s) = \sum_a Pr(s'|s,a)\pi(a|s)$. Then we can obtain that

$$\begin{aligned}d^{\pi'}_\mu - d^\pi_\mu &= (1-\gamma)[(\mathrm{I}-\gamma P_{\pi'})^{-1} - (\mathrm{I}-\gamma P_\pi)^{-1}]\mu\\ &= (1-\gamma)(M_{\pi'} - M_\pi)\mu\\ &= (1-\gamma)M_{\pi'}(M_\pi^{-1} - M_{\pi'}^{-1})M_\pi\mu\\ &= (1-\gamma)\gamma M_{\pi'}(P_{\pi'} - P_\pi)M_\pi\mu\\ &= \gamma M_{\pi'}(P_{\pi'} - P_\pi)d_\pi,\end{aligned}$$

where $M_\pi = (\mathrm{I}-\gamma P_\pi)^{-1}$. Therefore,

$$\begin{aligned}D_{TV}(d^\pi_\mu, d^{\pi'}_\mu) &= \frac{1}{2}\|\gamma M_{\pi'}(P_{\pi'} - P_\pi)d_\pi\|_1\\ &\le \frac{\gamma}{2}\|M_{\pi'}\|_1\|(P_{\pi'} - P_\pi)d^\pi_\mu\|_1,\end{aligned}$$

where $\|M_{\pi'}\|_1$ is bounded:

$$\|M_{\pi'}\|_1 = \|\sum_{t=0}^{\infty} \gamma^t P_{\pi'}^t\|_1 \leq \sum_{t=0}^{\infty} \gamma^t \|P_{\pi'}\|_1^t \leq \frac{1}{1-\gamma}$$

and $\|(P_{\pi'} - P_\pi)d_\mu^\pi\|_1$ is also bounded:

$$\begin{aligned}
\|(P_{\pi'} - P_\pi)d_\mu^\pi\|_1 &\leq \sum_{s,s'} |P_{\pi'}(s'|s) - P_\pi(s'|s)| d_\mu^\pi(s) \\
&= \sum_{s,s'} |\sum Pr(s'|s,a)(\pi'(a|s) - \pi(a|s))| d_\mu^\pi(s) \\
&\leq \sum_{s,a,s'} Pr(s'|s,a)|\pi'(a|s) - \pi(a|s)| d_\mu^\pi(s) \\
&= \sum_s d_\mu^\pi(s) \sum_a |\pi'(a|s) - \pi(a|s)| \\
&= 2\mathbb{E}_{s\sim d_\mu^\pi}[D_{TV}(\pi(\cdot|s), \pi'(\cdot|s))].
\end{aligned}$$

This completes the proof. □

Lemma A.2. *For two policies π and π' we have that*

$$D_{TV}(\rho_\mu^\pi, \rho_\mu^{\pi'}) \leq \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^\pi}[D_{TV}(\pi(\cdot|s), \pi'(\cdot|s))]. \tag{24}$$

Proof. By definition, we have

$$\begin{aligned}
D_{TV}(\rho_\mu^\pi, \rho_\mu^{\pi'}) &= \frac{1}{2}\sum_{s,a} |\pi(a|s)d_\mu^\pi(s) - \pi'(a|s)d_\mu^{\pi'}(s)| \\
&\leq \frac{1}{2}\sum_{s,a} |\pi(a|s) - \pi'(a|s)| d_\mu^\pi(s) + \frac{1}{2}\sum_{s,a} \pi'(a|s)|d_\mu^\pi(s) - d_\mu^{\pi'}(s)| \\
&= \mathbb{E}_{s\sim d_\mu^\pi}[D_{TV}(\pi(\cdot|s), \pi'(\cdot|s))] + D_{TV}(d_\mu^\pi, d_\mu^{\pi'}) \\
&\leq \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^\pi}[D_{TV}(\pi(\cdot|s), \pi'(\cdot|s))],
\end{aligned}$$

where the last inequality follows Lemma A.1. □

Lemma A.3. *For two policies π and π' we have that*

$$|V^\pi(\mu) - V^{\pi'}(\mu)| \leq \frac{2}{(1-\gamma)^2}\mathbb{E}_{s\sim d_\mu^\pi}[D_{TV}(\pi, \pi')] \tag{25}$$

Proof. Recall that for any policy π, the value can be written as $V^\pi(\mu) = \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim\rho_\mu^\pi}[r(s,a)]$. Then we have

$$\begin{aligned}
|V^\pi(\mu) - V^{\pi'}(\mu)| &= |\frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim\rho_\mu^\pi}[r(s,a)] - \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim\rho_\mu^{\pi'}}[r(s,a)]| \\
&\leq \frac{1}{1-\gamma}\sum_{s,a} |(\rho_\mu^\pi(s,a) - \rho_\mu^{\pi'}(s,a))r(s,a)| \\
&\leq \frac{2}{1-\gamma}D_{TV}(\rho_\mu^\pi, \rho_\mu^{\pi'}) \\
&\leq \frac{2}{(1-\gamma)^2}\mathbb{E}_{s\sim d_\mu^\pi}[D_{TV}(\pi, \pi')]
\end{aligned}$$

where the last inequality follows Lemma A.2. □

Lemma A.4. *(The performance difference lemma (Kakade & Langford, 2002)) For any two policies π, π', we have*

$$V^{\pi}(s) - V^{\pi'}(s) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_s^{\pi}}\mathbb{E}_{a\sim\pi}[A^{\pi'}(s,a)] \tag{26}$$

Proof. Using a telescoping argument, we have

$$\begin{aligned}
V^{\pi}(s) - V^{\pi'}(s) &= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)] - V^{\pi'}(s) \\
&= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t (r(s_t,a_t) + V^{\pi'}(s_t) - V^{\pi'}(s_t))] - V^{\pi'}(s) \\
&= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t (r(s_t,a_t) + \gamma V^{\pi'}(s_{t+1}) - V^{\pi'}(s_t))] \\
&= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t (r(s_t,a_t) + \gamma \mathbb{E}[V^{\pi'}(s_{t+1})|s_t,a_t] - V^{\pi'}(s_t))] \\
&= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t A^{\pi'}(s_t,a_t)] = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_s^{\pi}}\mathbb{E}_{a\sim\pi}[A^{\pi'}(s,a)].
\end{aligned}$$

□

Lemma A.5. *(Agarwal et al., 2020) For the softmax policy class, i.e., $\pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a'}(\exp\theta_{s,a'})}$ where $\theta \in \mathbb{R}^{|S||A|}$, we have*

$$\frac{\partial V^{\pi_\theta}}{\partial \theta_{s,a}} = \frac{1}{1-\gamma} d_\mu^{\pi_\theta}(s)\pi_\theta(a|s)A^{\pi_\theta}(s,a). \tag{27}$$

Proof. First note that

$$\frac{\partial log \pi_\theta(a|s)}{\partial \theta_{s',a'}} = 1[s=s'](1[a=a'] - \pi_\theta(a'|s)).$$

We have

$$\begin{aligned}
\frac{\partial V^{\pi_\theta}}{\partial \theta_{s,a}} &= \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t 1[s_t=s]\Big(1[a_t=a]A^{\pi_\theta}(s,a) - \pi_\theta(a|s)A^{\pi_\theta}(s_t,a_t)\Big)\right] \\
&= \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t 1[(s_t,a_t)=(s,a)]A^{\pi_\theta}(s,a)\right] - \pi_\theta(a|s)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\pi}\Big[1[s_t=s]A^{\pi_\theta}(s_t,a_t)\Big] \\
&= \frac{1}{1-\gamma}\mathbb{E}_{s'\sim d_\mu^{\pi_\theta}}\mathbb{E}_{a'\sim\pi_\theta}\Big[1[(s',a')=(s,a)]A^{\pi_\theta}(s,a)\Big] - 0 \\
&= \frac{1}{1-\gamma}d_\mu^{\pi_\theta}(s)\pi_\theta(a|s)A^{\pi_\theta}(s,a),
\end{aligned}$$

where the second to last step comes from that for any policy $\sum_a \pi(a|s)A^{\pi}(s,a) = 0$. □

A.2. Optimality in MARL

The concept of an optimal policy in MARL is not clear as (Devlin & Kudenko, 2011) in single-agent settings. Generally, the optimal policy in multi-agent tasks can be defined as some Nash equilibrium. Specially, in the fully cooperative setting with shared reward, the optimal policy is Pareto optimality which is also one of the Nash equilibrium. Other Nash equilibrium represents sub-optimal policy. As a result, the ultimate goal of cooperative MARL is Pareto optimality which maximizes the shared expected return. However, the optimal policy of the team is not always available with individual policies.

Formally, let us consider the results in LemmaA.3. Denote the optimal policy as π^*, then for any policy π, we have $V^{\pi} \geq V^* - \frac{2}{(1-\gamma)^2}\mathbb{E}_{s\sim d_\mu^*}[D_{TV}(\pi^*,\pi)]$. This means if we can learn a nearly optimal policy, i.e. $\mathbb{E}_{s\sim d_\mu^*}[D_{TV}(\pi^*,\pi)] \leq \epsilon$,

V^{π} can be very close to the optimal value when ϵ is sufficiently small. However, in multi-agent settings with decentralized policies, we may not be able to obtain a very small ϵ due to the limited class of the policy. To validate this, we only need a simple example. Consider a repeated game where two agents have two actions a_1 and a_2. The joint policy $\pi(a_i, a_j) = \pi_1(a_i) \cdot \pi_2(a_j)$, $i, j = 1, 2$, where π_1 and π_2 are individual policies for the two agents respectively. If the optimal policy is given by $\pi^*(a_1, a_1) = \pi^*(a_2, a_2) = \frac{1}{2}$ and $\pi^*(a_1, a_2) = \pi^*(a_2, a_1) = 0$, it is easy to show that $D_{TV}(\pi^*, \pi_1\pi_2) \geq \sqrt{2} - 1$ for any policy π_1 and π_2, which means that $\epsilon \geq \sqrt{2} - 1$. Note that the bound in Lemma A.3 is tight up to a constant C (Xu et al., 2020). Therefore, in the worse case, we can only have $V^{\pi} \geq V^* - \frac{2(\sqrt{2}-1)}{(1-\gamma)^2} C$, which means the value V^{π} can have a large gap to the optimal value for any policy π.

A.3. Policy Bias

In Section 3.1, we discussed the k-step ($k \geq 2$) reward shaping enables credit assignment more as k increases, but also shifts the policy gradients more. Here we give some theoretical analysis. Considering k-step reward shaping $r_{i,t}^{(k)} = r_t + \gamma^k \mathbb{E}_{a_i}[r_{t+k}] - \mathbb{E}_{a_i}[r_t]$, we have

$$\hat{Q}_i^{(k)}(s, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=0}^{\infty} \gamma^t r_{i,t}^{(k)}] = Q(s, \boldsymbol{a}) - \sum_{t=1}^{k-1} \gamma^t \phi_i^{(t)}(s, \boldsymbol{a}) - \phi_i^{(0)}(s, a_{-i}), \tag{28}$$

where $\phi_i^{(k)}(s, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{\pi}}[r_k]$ represents the expected reward in the k-th step after the given state and actions.

As discussed in Section A.2, dealing with multi-agent policy gradient directly will not yield a tighter bound than $\frac{2(\sqrt{2}-1)}{(1-\gamma)^2} C$. This is because the joint policy class is constrained by individual policies. If individual policies, for example, are softmax parameterized, the joint policy is restrictive and will not be a softmax function. To circumvent this issue and simplify the theoretical analysis, we consider the policy gradient of one single agent with softmax policy and some entropy regularization term, i.e.,

$$L(\theta) = V^{\pi_\theta}(\mu) + \lambda H(\pi_\theta), \tag{29}$$

where $H(\pi_\theta) = -\mathbb{E}_{s \sim \text{Unif}_{\mathcal{S}}}[D_{KL}(\text{Unif}_{\mathcal{A}}, \pi_\theta(\cdot|s))]$. In this case, other agents' policies can be considered as fixed, and we can apply the results in Agarwal et al. (2019) to get a lower bound of the value function. The following theorem formally states that the bound between the optimal value and learned value becomes looser as k increase.

Theorem A.6. *For the biased pilicy gradient*

$$\nabla_\theta \hat{V}^{\pi_\theta} = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\rho^{\pi_\theta}} \mathbb{E}_{a \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)(\hat{A}_k^{\pi_\theta}(s, a)], \tag{30}$$

where $\hat{A}_k = \hat{Q}^{(k)} - \mathbb{E}_{\pi}[\hat{Q}^{(k)}]$. *Suppose the policy gradient is such that:*

$$\|\nabla_\theta \hat{L}(\theta)\|_2 \leq \epsilon \tag{31}$$

where $\nabla_\theta \hat{L}(\theta) = \hat{V}^{\pi_\theta} + \lambda H(\pi_\theta)$ *and* $\epsilon \leq \lambda / 2|\mathcal{S}|\,|\mathcal{A}|$. *Then we have for any initial distribution* μ'*:*

$$V^{\pi_\theta}(\mu') \geq V^*(\mu') - \frac{2}{1-\gamma}\left(\lambda \left\| \frac{d_{\mu'}^{\pi^*}}{\mu} \right\|_{\infty} + \sum_{t=1}^{k-1} \gamma^t \right). \tag{32}$$

Proof. The proof consists of showing that $\max_a \hat{A}^{\pi_\theta}(s, a) \leq 2\lambda / \mu(s)|\mathcal{S}|$ for all states. To see that this is sufficient, observe that by the performance difference lemma (Lemma A.4),

$$\begin{aligned} V^*(\mu') - V^{\pi_\theta}(\mu') &= \frac{1}{1-\gamma} \sum_{s,a} d_{\mu'}^{\pi^*}(s) \pi^*(a|s) A^{\pi_\theta}(s, a) \\ &\leq \frac{1}{1-\gamma} \sum_{s} d_{\mu'}^{\pi^*}(s) \max_a A^{\pi_\theta}(s, a). \end{aligned}$$

Note that

$$\begin{aligned}\hat{A}^{\pi_\theta}(s,a) &= A^{\pi_\theta}(s,a) - \sum_{t=1}^{k-1}\gamma^t(\phi_i^{(k)}(s,a) - \mathbb{E}_\pi[\phi_i^{(k)}(s,a)]) \\ &\geq A^{\pi_\theta}(s,a) - \sum_{t=1}^{k-1}\gamma^t(|\phi_i^{(k)}(s,a)| + |\mathbb{E}_\pi[\phi_i^{(k)}(s,a)]|) \\ &\geq A^{\pi_\theta}(s,a) - 2\sum_{t=1}^{k-1}\gamma^t,\end{aligned} \tag{33}$$

we have

$$\begin{aligned}V^*(\mu') - V^{\pi_\theta}(\mu') &\leq \frac{1}{1-\gamma}\sum_s d_{\mu'}^{\pi^*}(s)(\frac{2\lambda}{\mu(s)\mathcal{S}} + 2\sum_{t=1}^{k-1}\gamma^t) \\ &\leq \frac{2\lambda}{1-\gamma}\max_s\Big(\frac{d_{\mu'}^{\pi^*}(s)}{\mu(s)}\Big) + \frac{2}{1-\gamma}\sum_{t=1}^{k-1}\gamma^t.\end{aligned} \tag{34}$$

which then completes the proof.

We now proceed to show that $\max_a \hat{A}^{\pi_\theta}(s,a) \leq 2\lambda/\mu(s)|\mathcal{S}|$. For this, it suffices to bound $\hat{A}^{\pi_\theta}(s,a)$ for any state-action pair s,a where $\hat{A}^{\pi_\theta}(s,a) > 0$ else the claim is trivially true. Consider a (s,a) pair such that $\hat{A}^{\pi_\theta}(s,a) > 0$. Using the policy gradient expression for the softmax parameterization (Lemma A.5),

$$\frac{\partial \hat{L}(\theta)}{\partial \theta_{s,a}} = \frac{1}{1-\gamma}d_\mu^{\pi_\theta}(s)\pi_\theta(a|s)\hat{A}^{\pi_\theta}(s,a) + \frac{\lambda}{|\mathcal{S}|}(\frac{1}{|\mathcal{A}|} - \pi_\theta(a|s)), \tag{35}$$

For the first term, note that Lemma A.5 only requires $\sum_a \pi(a|s)A^\pi(s,a) = 0$. Therefore it holds with $\hat{A}^{\pi_\theta}$, i.e., the biased policy gradient. The second term comes from that $H(\pi) = \frac{1}{|\mathcal{S}||\mathcal{A}|}\sum_{s,a}\log\pi(a|s) + \log|\mathcal{A}|$.

Then, the gradient norm assumption $||\nabla_\theta \hat{L}(\theta)||_2 \leq \epsilon$ implies that:

$$\begin{aligned}\epsilon \geq \frac{\partial \hat{L}(\theta)}{\partial \theta_{s,a}} &= \frac{1}{1-\gamma}d_\mu^{\pi_\theta}(s)\pi_\theta(a|s)\hat{A}^{\pi_\theta}(s,a) + \frac{\lambda}{|\mathcal{S}|}(\frac{1}{|\mathcal{A}|} - \pi_\theta(a|s)) \\ &\geq \frac{\lambda}{|\mathcal{S}|}(\frac{1}{|\mathcal{A}|} - \pi_\theta(a|s)),\end{aligned} \tag{36}$$

where we have used $\hat{A}^{\pi_\theta}(s,a) > 0$. Rearranging and using our assumption $\epsilon \leq \lambda/(2|\mathcal{S}||\mathcal{A}|)$,

$$\pi_\theta(a|s) \geq \frac{1}{|\mathcal{A}|} - \frac{\epsilon|\mathcal{S}|}{\lambda} \geq \frac{1}{2|\mathcal{A}|}.$$

Solving for $\hat{A}^{\pi_\theta}(s,a)$ in (36), we have

$$\begin{aligned}\hat{A}^{\pi_\theta}(s,a) &= \frac{1-\gamma}{d_\mu^{\pi_\theta}(s)}\left(\frac{1}{\pi_\theta(a|s)}\frac{\partial \hat{L}(\theta)}{\partial \theta_{s,a}} + \frac{\lambda}{|\mathcal{S}|}\left(1 - \frac{1}{\pi_\theta(a|s)|\mathcal{A}|}\right)\right) \\ &\leq \frac{1-\gamma}{d_\mu^{\pi_\theta}(s)}\left(2|\mathcal{A}|\epsilon + \frac{\lambda}{|\mathcal{S}|}\right) \\ &\leq 2\frac{1-\gamma}{d_\mu^{\pi_\theta}(s)}\frac{\lambda}{|\mathcal{S}|} \leq 2\lambda/\mu(s)|\mathcal{S}|,\end{aligned}$$

where the penultimate step uses $\epsilon \leq \lambda/(2|\mathcal{S}||\mathcal{A}|)$ and the final step uses $d_\mu^{\pi_\theta}(s) \geq (1-\gamma)\mu(s)$. This completes the proof. □

Theorem A.6 implies that the policy becomes more biased as k increase. However, compared with the unbiased case ($k = 1$), the bias is not minimal even for $k = 2$, where the bound is loosened by the term $\frac{2\gamma}{1-\gamma}$. To smoothly interpolate between the biased and unbiased policy gradient, we can introduce a parameter β and do an exponentially weighted sum of the k-step reward shaping as in (16). The bias can then be smoothly controlled by selecting different $beta$ values.

Corollary A.7. *Suppose the policy gradient is such that*

$$\|\nabla_\theta \hat{L}_\beta(\theta)\|_2 \leq \epsilon, \tag{37}$$

and $\epsilon \leq \lambda/2|\mathcal{S}||\mathcal{A}|$. *Then we have that:*

$$V^{\pi_\theta}(\mu') \geq V^*(\mu') - \frac{2}{1-\gamma}\left(\lambda \left\| \frac{d_{\mu'}^{\pi^*}}{\mu} \right\|_\infty + \frac{\beta\gamma}{1-\beta\gamma}\right). \tag{38}$$

Proof. Observe that the Q function after an exponentially weighed average of k-step reward shaping is given by:

$$\hat{Q}_i^{(\beta)}(s,\boldsymbol{a}) = Q(s,\boldsymbol{a}) - \sum_{t=1}^{\infty}(\beta\gamma)^t \phi_i^{(t)}(s,\boldsymbol{a}) - \phi_i^{(0)}(s,a_{-i}). \tag{39}$$

Comparing (39) with (28), we can let $k=\infty$ and replace γ with $\beta\gamma$ in the last term of (32). Then we get

$$\begin{aligned} V^{\pi_\theta}(\mu') &\geq V^*(\mu') - \frac{2}{1-\gamma}\left(\lambda \left\| \frac{d_{\mu'}^{\pi^*}}{\mu} \right\|_\infty + \sum_{t=1}^{\infty}(\beta\gamma)^t\right) \\ &= V^*(\mu') - \frac{2}{1-\gamma}\left(\lambda \left\| \frac{d_{\mu'}^{\pi^*}}{\mu} \right\|_\infty + \frac{\beta\gamma}{1-\beta\gamma}\right). \end{aligned}$$

□

This corollary gives a lower bound of the learned value function using the weighted-sum of k-step reward shaping. It should be noted that this does not imply that utilizing the gradient after reward shaping will result in poor policy.

In addition, we can expect a tighter bound if the transition function is not very action-dependent, which implies the term $\phi_i^{(k)}(s,a) - \mathbb{E}_\pi[\phi_i^{(k)}(s,a)]$ in (33) will be smaller. An extreme example is that $P(s'|s,\boldsymbol{a}) = P(s'|s)$, then we have

$$\phi_i^{(k)}(s_t,\boldsymbol{a_t}) = \mathbb{E}_{s_{t+k}\sim P(s_{t+k}|s_t)}\mathbb{E}_{\boldsymbol{a_{t+k}}\sim\boldsymbol{\pi}}[r(s_{t+k},\boldsymbol{a_{t+k}})] = \phi_i^{(k)}(s_t)$$

Then the policy gradient (30) is unbiased since ϕ is only related to state. This is the case with the matrix game (Section 4.1) where the reward shaping keeps policy invariant.

B. Pseudo Code

We describe the details of DAE based on MAPPO in Algorithm 1. Specifically, in line 17 we additionally store the action distribution $\boldsymbol{p_t}$ of all agents to compute the reward expectation as $\mathbb{E}_{a_t^{(i)}\sim p_t^{(i)}}[R_t^{(i)}]$ ($\sum_{a_t^{(i)}} p_t^{(i)} R_t^{(i)}$ for discrete actions) for each agent i at time step t. The target value $\hat{V}$ in line 20 is computed using TD(λ) backup: $\hat{V}_t^{(i)} = V_{\phi_{old}}^{(i)} + \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^{(i)}$, as in MAPPO. Consequently, the critic network with parameter φ is trained to minimize the loss function

$$L(\varphi) = \max[(V_\phi^{(i)} - \hat{V}^{(i)})^2, (\text{clip}(V_{\phi_{\text{old}}}, V_{\phi_{\text{old}}} - \epsilon, V_{\phi_{\text{old}}} + \epsilon) - \hat{V}^{(i)})^2]. \tag{40}$$

C. Hyperparameter Settings for Experiments

The implementation of MAPPO follows the original paper (Yu et al., 2021). The hyperparameters are basically default setting in MAPPO as presented in Table 1, Table 2 and Table 3.

D. Additional Results

D.1. Separated Results on MPE and SMAC

Figure 5 shows the separated results on MPE for three tasks and $N=3,4,5$ respectively.
Figure 6 shows the separated results on SMAC for nine maps respectively.

Algorithm 1. Recurrent-MAPPO with DAE

```
1: Initialize $\theta$, the parameters for policy $\pi$ and $\varphi$, the parameters for critic $V$ and $\psi$ the parameters for reward $R$
2: while $step \leq step\_max$ do
3:   set data buffer $D = \{\}$
4:   for $j = 1$ to $batch\_size$ do
5:     $\tau = []$ empty list
6:     initialize $h^{(1)}_{0,\pi}, ..., h^{(n)}_{0,\pi}$ actor RNN states
7:     initialize $h^{(1)}_{0,V}, ..., h^{(n)}_{0,V}$ critic RNN states
8:     initialize $h^{(1)}_{0,R}, ..., h^{(n)}_{0,R}$ reward RNN states
9:     for $t = 1$ to $T$ do
10:      for all agent $i$ do
11:        $p^{(i)}_t, h^{(i)}_{t,\pi} = \pi(o^{(i)}_t, h^{(i)}_{t-1,\pi}; \theta)$
12:        $a^{(i)}_t \sim p^{(i)}_t$
13:        $v^{(i)}_t, h^{(i)}_{t,V} = V(s_t, h^{(i)}_{t-1,V}; \varphi)$
14:        $R^{(i)}_t, h^{(i)}_{t,R} = R(s_t, h^{(i)}_{t-1,R}, \boldsymbol{a_t}; \psi)$
15:      end for
16:      Execute actions $\boldsymbol{a_t}$, observe $r_t, s_{t+1}, o_{t+1}$
17:      $\tau += [s_t, \boldsymbol{o_t}, \boldsymbol{h_{t,\pi}}, \boldsymbol{h_{t,V}}, \boldsymbol{h_{t,R}}, \boldsymbol{a_t}, \boldsymbol{p_t}, r_t, \boldsymbol{R_t}, s_{t+1}, \boldsymbol{o_{t+1}}]$
18:    end for
19:    Compute advantage estimate $\hat{A}$ via DAE on $\tau$
20:    Compute target value $\hat{V}$ on $\tau$
21:    Split trajectory $\tau$ into chunks of length $L$
22:    for $l = 1$ to $T//L$ do
23:      $D = D \bigcup (\tau[l : l+T], \hat{A}[l : l+T], \hat{V}[l : l+T])$
24:    end for
25:  end for
26:  for mini-batch $k = 1, ..., K$ do
27:    $b \leftarrow$ random mini-batch from $D$ with all agent data
28:    for each data chunk $c$ in the mini-batch $b$ do
29:      update RNN hidden states for $\pi$,$V$ and $R$ from first hidden state in data chunk
30:    end for
31:  end for
32:  Adam updates $\theta$,$\varphi$ and $\psi$ with mini-batch $b$
33: end while
```

D.2. Additional Results for COMA

As an advantage estimator, using Q-values subtracted a baseline is highly biased (Schulman et al., 2016). The bias comes from the estimation error of Q-values. Therefore, the estimation difficulty of the joint Q-values in COMA yields large bias to the policy gradients. A simple way to reduce bias at the cost of some variance is to replace the original advantage by n-step TD residual:

$$\delta^n_t = r_t + \gamma r_{t+1} + ... + \gamma^n Q(s_{t+n+1}, \boldsymbol{a}_{t+n+1}) - V(s_t), \tag{41}$$

where $V(s) = \mathbb{E}_{a_i \sim \pi} Q(s, \boldsymbol{a})$ for each agent i in COMA. That is, the $Q(s_t, \boldsymbol{a}_t)$ in the original advantage is expanded to the n-step Q: $r_t + \gamma r_{t+1} + ... + \gamma^n Q(s_{t+n+1}, \boldsymbol{a}_{t+n+1})$, which consists of two terms, an n-step unbaised but high-variance return, and a lower biased term due to the discount factor γ^n. As shown in Figure 7, we found this modification can significantly improves COMA's performance. However, there is still a large performance gap between COMA and DAE.

E. Future Work

In this paper, we realize DAE using a reward network with recurrent layers. For the reward learning, many approaches can be employed to further improve the performance, such as factorization (Castellini et al., 2019) and redistribution

Table 1. Common hyperparameters used across all environments.

hyperparameters	value	hyperparameters	value	hyperparameters	value
gamma	0.99	optimizer	Adam	actor hidden dim	64
gae lamda	0.95	network	rnn	value hidden dim	64
num mini-batch	1	ppo-clip	0.2	reward hidden dim	128
max grad norm	10	activation	ReLU	hidden layer	1MLP+1GRU

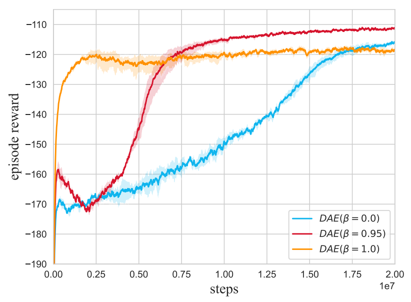
Table 2. Hyperparameters used in Matrix Game (MG) and MPE.

hyperparameters	value	hyperparameters	value
actor lr	7e-4	episode length (MG)	16
critic lr	7e-4	episode length (MPE)	25
rollout threads	128	epoch	10

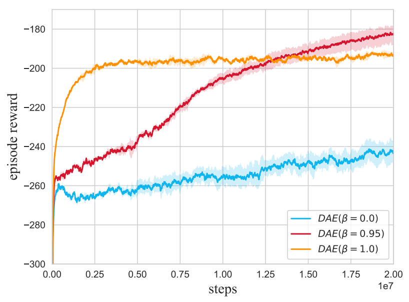
Table 3. Hyperparameters used in SMAC.

hyperparameters	value	hyperparameters	value
actor lr	5e-4	episode length	400
critic lr	5e-4	rollout threads	8

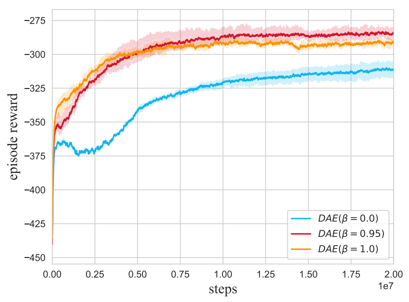
(Arjona-Medina et al., 2019). Another possible option is to utilize an additional Q-network which learns a heavier discounted return. We will investigate this in future work.



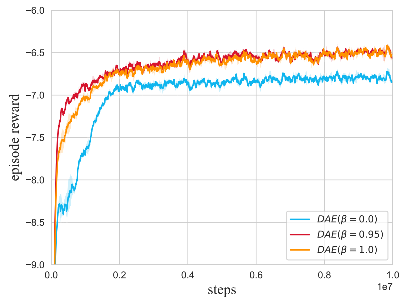
(a) cooperative navigation ($N = 3$)



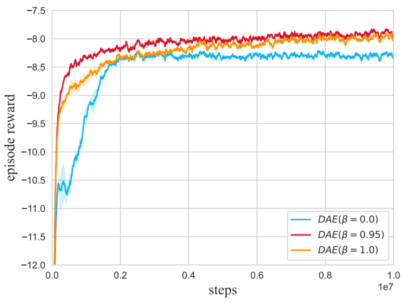
(b) cooperative navigation ($N = 4$)



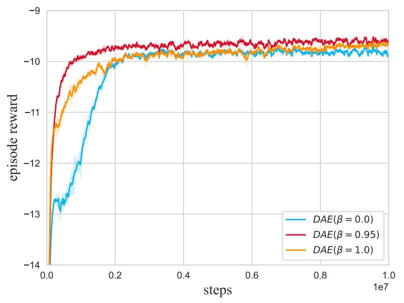
(c) cooperative navigation ($N = 5$)



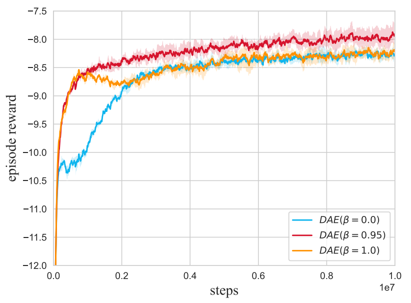
(d) line control ($N = 3$)



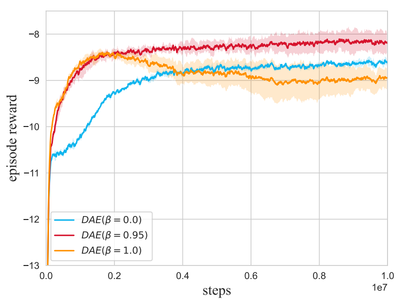
(e) line control ($N = 4$)



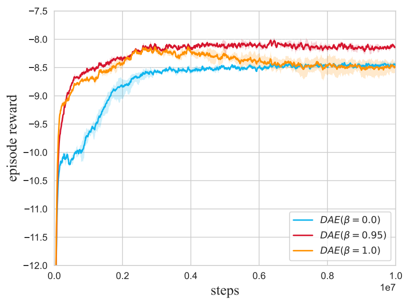
(f) line control ($N = 5$)



(g) formation control ($N = 3$)

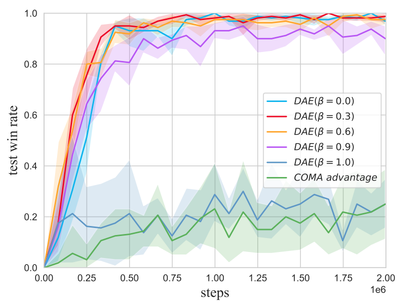


(h) formation control ($N = 4$)

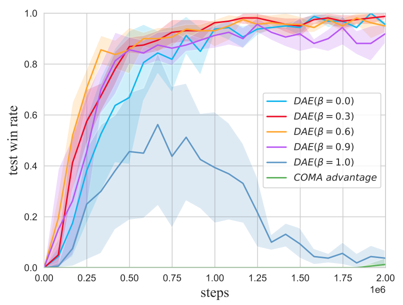


(i) formation control ($N = 5$)

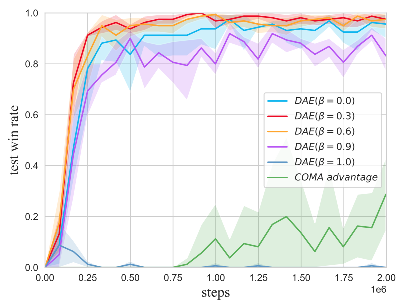
Figure 5. Separated results on MPE tasks.



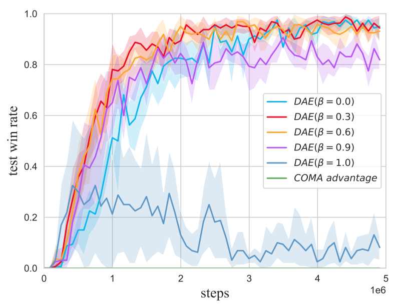
(a) 2s3z



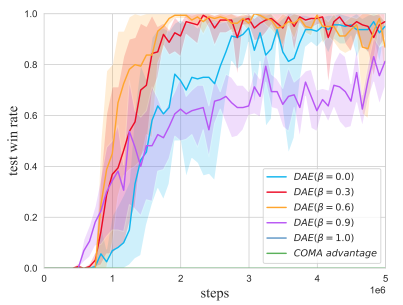
(b) 1c3s5z



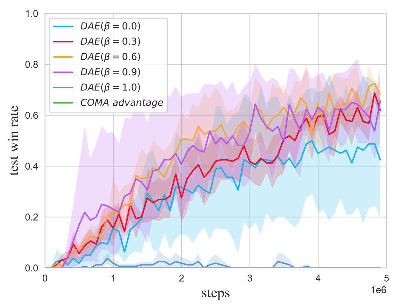
(c) MMM



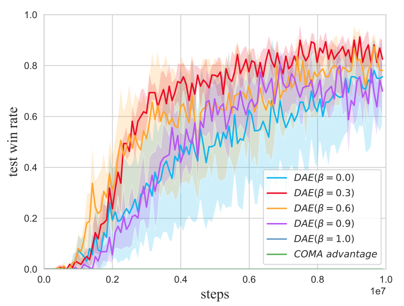
(d) 3s5z



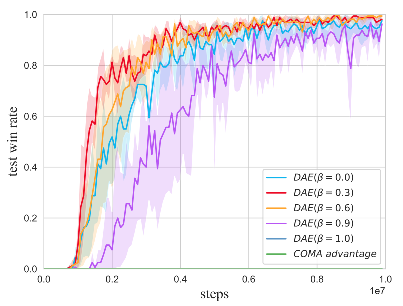
(e) 3s_vs_5z



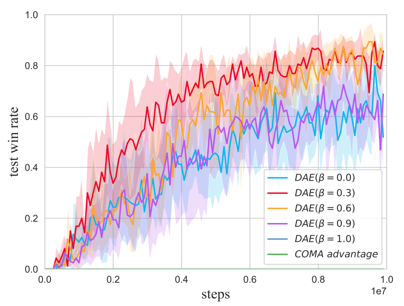
(f) 5m_vs_6m



(g) 3s5z_vs_3s6z

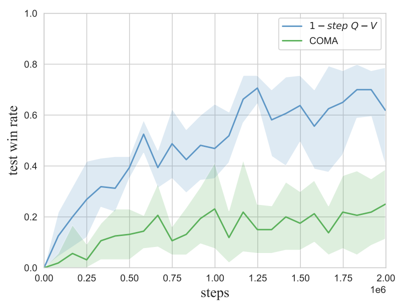


(h) corridor

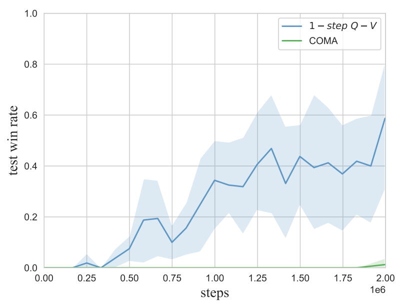


(i) MMM2

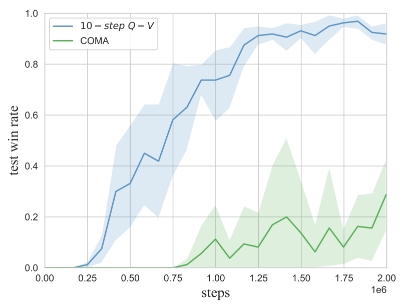
Figure 6. Separated results on nine SMAC maps.



(a) 2s3z



(b) 1c3s5z



(c) MMM

Figure 7. Additional results for COMA, where we compare the advantage estimator $Q - V$ as in COMA and n-step $Q - V$ as in (41).