

# Generating individual intrinsic reward for cooperative multiagent reinforcement learning

*International Journal of Advanced  
Robotic Systems*

September–October 2021: 1–8

© The Author(s) 2021

Article reuse guidelines:

sagepub.com/journals-permissions

DOI: 10.1177/17298814211044946

journals.sagepub.com/home/arx



Haolin Wu<sup>1</sup>, Hui Li<sup>2</sup> , Jianwei Zhang<sup>2</sup>, Zhuang Wang<sup>1</sup>  
and Jianeng Zhang<sup>1</sup>

## Abstract

Multiagent reinforcement learning holds considerable promise to deal with cooperative multiagent tasks. Unfortunately, the only global reward shared by all agents in the cooperative tasks may lead to the lazy agent problem. To cope with such a problem, we propose a generating individual intrinsic reward algorithm, which introduces an intrinsic reward encoder to generate an individual intrinsic reward for each agent and utilizes the hypernetworks as the decoder to help to estimate the individual action values of the decomposition methods based on the generated individual intrinsic reward. Experimental results in the StarCraft II micromanagement benchmark prove that the proposed algorithm can increase learning efficiency and improve policy performance.

## Keywords

Cooperative multiagent reinforcement learning, lazy agent problem, global reward, intrinsic reward, generation

Date received: 20 May 2021; accepted: 16 August 2021

Topic: AI in Robotics; Human Robot/Machine Interaction

Topic Editor: Alessandro Di Nuovo

Associate Editor: Joni Zhong

## Introduction

Many real-world tasks are cooperative multiagent problems, in which all agents work together to achieve a common goal, such as distributed logistics,<sup>1</sup> crewless aerial vehicles,<sup>2</sup> autonomous driving,<sup>3</sup> and network packet routing.<sup>4</sup> Multiagent reinforcement learning (MARL) holds considerable promise to deal with such tasks.

However, the sparse reward is a long-standing problem in the reinforcement learning (RL) field. Worse still, this problem will be more server in the cooperative MARL tasks, in which all agents usually share a global reward. Specifically, the sparse global reward not only reduces the learning efficiency but also may lead to the lazy agent problem in the multiagent field,<sup>5,6</sup> which means that it is difficult for each agent to confirm its contribution to the team's success (i.e. the shared global reward). As a consequence, if an agent learns the decentralized policy based on

the global reward directly, it will take the global reward originated from its teammates' behavior as its contribution, thus encouraging its current meaningless action.

To address the lazy agent problem, decomposition methods<sup>5,7,8</sup> first learn a joint action value (JAV) for all agents based on the global reward and the joint experiences of all agents (i.e. joint observations and the joint actions). Then, each agent learns the individual action value (IAV) implicitly from the JAV decomposition rather than from the

<sup>1</sup> College of Computer Science, Sichuan University, Chengdu, China

<sup>2</sup> National Key Laboratory of Fundamental Science on Synthetic Vision, Sichuan University, Chengdu, China

### Corresponding author:

Hui Li, National Key Laboratory of Fundamental Science on Synthetic Vision, Sichuan University, Chengdu, China.

Email: lihuib@scu.edu.cn



Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (<https://creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work without

further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

global reward directly. Finally, the per-agent decentralized policy can be determined based on its IAVs. To this end, value-decomposition networks (VDN)<sup>5</sup> additively decomposes the JAV into IAVs across agents, QMIX<sup>7</sup> replace the additivity with the monotonicity, and QTRAN<sup>8</sup> is free from the additivity/monotonicity structural constraints to make the decomposition method more general. Thereby, the IAVs are trained end-to-end with the optimization of the JAV.

Another approach to address the lazy agent problem is assigning each agent an individual reward. Reward shaping<sup>9</sup> aims to manually design individual reward functions for each agent. However, it needs heavy and careful manual work, which is hard to ensure that the optimal policy will not be changed even in the single-agent field.<sup>10,11</sup> Thus, a group of recent single-agent RL methods aims to learn parametrized intrinsic reward functions to replace the manually designed reward functions.<sup>12–14</sup> Inspired by the concept, learning individual intrinsic reward (LIIR)<sup>15</sup> learns each agent an intrinsic reward function and then combines the intrinsic reward and the global reward to optimize the policy via the actor-critic<sup>16,17</sup> algorithm.

In this article, we propose a generating individual intrinsic reward (GIIR) algorithm in MARL, which constructs a connection between the learned individual intrinsic reward (IIR) and the decomposition methods. We assume that there exists an individual reward function for each agent, thus introducing an *intrinsic reward encoder* to generate the IIR  $r_i^j$  based on per-agent local experience (i.e. local partial observation and local action). Then, the generated IIR is used to participate in learning the IAVs of the corresponding agent  $i$ . The key insight of our work is that by ensuring the goal of the IIR is increasing the expected global reward, we can make true that the decentralized policy determined according to the IIR-based IAVs can obtain greater teamwork success, that is, gain more global reward. To this end, the parameters of the *intrinsic reward encoder* are optimized by maximizing the expected sum of the global reward. Besides, similar to LIIR,<sup>15</sup> the generated IIR does not have any physical meanings and is only used to learn the IIR-based IAVs. Thus, we take the generated IIR as the input of *hypernetworks*<sup>18</sup> to output the parameters of the agent network and use the agent network to output the IAVs based on local experience. We test the performance of the proposed GIIR in the benchmark environment StarCraft multiagent challenge (SMAC),<sup>19</sup> and the experimental results prove that the proposed GIIR can outperform both decomposition methods and LIIR.

## Related work

The naive MARL method to address the cooperative multiagent tasks is joint action learning (JAL),<sup>20</sup> which takes all agents as an agent and learns a JAV based on the joint experience of all agents. However, the number of actions increases exponentially with the number of agents, which

makes it intractable. Thus, individual learning views other agents as parts of the environment and learns an IAV based on per-agent local experience. Then, each agent performs the decentralized policy without considering other agents. However, because of ignoring the strategy changes of other agents, it fails to coordinate with other agents efficiently. Besides, all agents often share a global reward in cooperative multiagent tasks. Under such circumstances, it is hard for each agent to confirm its contribution to the team's success. Thus, each agent may view the rewards that originated from its teammates' behavior as its contribution, which may lead to the lazy agent problem.<sup>5,6</sup>

One approach to cope with the lazy agent problem is the decomposition method,<sup>5,7,8</sup> which aims to learn IAV implicitly from the JAV decomposition rather than from the global reward directly. Another approach is providing each agent an individual reward. Reward shaping<sup>9</sup> manually designs the individual reward for each agent, but it requires heavy handwork to accurately assign rewards to each agent and is difficult to handle in practice. LIIR<sup>15</sup> adopts the actor-critic algorithm<sup>17,16</sup> to address the MARL tasks and learns each agent an IIR, then the actor uses the IIR to learn each agent's policy. The emergence of individuality (EOI)<sup>21</sup> gives each agent an intrinsic reward, but it aims to learn the individuality based on the intrinsic reward to drive agents to behave differently. Thus, the intrinsic reward is outputted from a classifier in EOI and the role is to distinguish different agents. Our work is closely related to the decomposition methods and LIIR. We learn a parameterized intrinsic reward and introduce it into the decomposition methods to help to improve the learning efficiency.

Our work is also related to the single-agent works about the intrinsic reward. Most works take curiosity as the intrinsic reward either to encourage the agent to explore novel states<sup>22,23</sup> or to encourage the agent to reduce the uncertainty in predicting the consequence of its actions.<sup>24,25</sup> Besides, Zheng et al.<sup>12,14</sup> and Bahdanau et al.<sup>13</sup> learn parameterized intrinsic reward to help achieve learning goals.

## Background

In this work, we focus on the setting of Decentralized Partial Observation Markov Decision Process (Dec-POMDP).<sup>26</sup> It models the fully cooperative MARL task as a tuple  $G = \langle n, S, U, P, r, \mathcal{O}, \gamma \rangle$ , where  $n$  is the number of agents,  $s_t \in S$  is the true global state in the environment,  $\mu_t^i \in U$  is the individual action chosen by each agent,  $\mu_t \in U \equiv U^n$  is the joint action of all agents,  $P(s_{t+1}|s_t, \mu_t) : S \times U \times S \rightarrow [0, 1]$  is the transition function to determine a next global state  $s_{t+1}$  after that all agents perform the joint action,  $r(s_t, \mu_t) : S \times U \rightarrow \mathbb{R}$  is the global reward shared by all agents, and  $\gamma$  is the discount factor. Besides, Decentralized Partial Observation Markov Decision Process (Dec-POMDP) considers the partial local observation setting, in which each agent can only access the

local observation  $o_i \in \mathcal{O}$  according to the observation function  $O(s, i)$ .

To handle the partial observability, there is a technique<sup>27</sup> in MARL that applying the recurrent neural network<sup>28</sup> to estimate the IAV  $Q_i(\tau^i, u^i)$  and the JAV  $Q_{jt}(\tau, u)$  based on the local observation history and the local action history, where  $\tau^i = (o_1^i, o_2^i, \dots, o_T^i)$  is the local observation history,  $u^i = (\mu_1^i, \mu_2^i, \dots, \mu_T^i)$  is the local action history of an agent  $i$ ,  $\tau$  is the joint local observation history, and  $u$  is the joint local action history of all agents.

### Individual Q-learning

Individual Q-learning (IQL) views other agents as part of the environment, in which each agent uses the global reward to learn an IAV  $Q_i(o_t^i, \mu_t^i)$  based on its local experience (i.e. local observation and local action). Recent works<sup>6,29</sup> extend it to the deep reinforcement field, which applies the neural network with parameters  $\theta_i$  to estimate the IAV, and the parameters are optimized by the loss function

$$L(\theta_i) = \left( r_t + \gamma \max_{\mu_{t+1}^i} Q_i(o_{t+1}^i, \mu_{t+1}^i; \theta_i^-) - Q_i(o_t^i, \mu_t^i; \theta_i) \right)^2 \quad (1)$$

where  $\theta_i^-$  represent the parameters of the target networks and they are not optimized in Equation (1) but updated by the periodic copy from the parameters  $\theta_i$  of the *online networks*.

Note that the global reward may originate from its teammates' behavior, but each agent uses the global reward  $r_t$  to update its IAV in Equation (1), which may lead to the lazy agent problem. Besides, because the dynamics of the environment will change as its teammate changes its behavior strategy, the learning process of IQL may be a nonstationary problem.

### Decomposition method

Decomposition methods<sup>5,7,8</sup> apply the global reward to learn a JAV based on the joint experience (i.e. joint observations and joint actions), which is similar to the JAL.<sup>20</sup> Then, they decompose the JAV into per-agent IAV  $Q_i(o_t^i, \mu_t^i)$ , thus each agent can perform decentralized policies according to the IAV based on local experience. Most importantly, the IAV of each agent can be learned implicitly through end-to-end training from the JAV decomposition rather than from the global reward directly.

The key condition of decomposition methods is individual-global-max (IGM),<sup>8</sup> which can make true that the optimal joint actions based on the JAV are equivalent to the collection of individual optimal actions of each agent based on the IAVs by ensuring that the global argmax

operation on the JAV is the same with a collection of simple individual argmax operations of each IAV

$$\operatorname{argmax}_{\mu_t} Q_{jt}(\tau_t, \mu_t) = \begin{pmatrix} \operatorname{argmax}_{\mu_t^1} Q_1(o_t^1, \mu^1) \\ \operatorname{argmax}_{\mu_t^2} Q_2(o_t^2, \mu^2) \\ \vdots \\ \operatorname{argmax}_{\mu_t^N} Q_N(o_t^N, \mu^N) \end{pmatrix} \quad (2)$$

Thus, we can determine decentralized policies based on the optimal IAVs of each agent while the goal of the training is optimizing the JAV.

To satisfy IGM, VDN<sup>5</sup> decomposes the JAV based on the additivity

$$Q_{jt}(\tau, \mu) = \sum_{i=1}^N Q_i(\tau^i, u^i) \quad (3)$$

where  $Q_{jt}(\tau, \mu)$  is mixed as the sum of each  $Q_i(\tau^i, u^i)$ .

QMIX<sup>7</sup> decomposes the JAV based on the monotonicity

$$\frac{\partial Q_{jt}(\tau, u)}{\partial Q_i(\tau^i, u^i)} = g_i \geq 0, \quad \forall i \in \mathcal{N} \quad (4)$$

where  $Q_i(\tau^i, u^i)$  is mixed into the  $Q_{jt}(\tau, \mu)$  by a *mixing network* rather than directly summing together in Equation (3), and the parameters of *mixing network* are generated by separate *hypernetworks*<sup>18</sup> to ensure positive. Note that VDN can be regarded as a special case of QMIX, in which  $g_i = 1$ .

QTRAN<sup>8</sup> transforms the JAV into the sum of the IAVs and a state value, which can be free from the additivity/monotonicity structural constraints

$$\sum_{i=1}^N Q_i(\tau^i, u^i) - Q_{jt}(\tau, u) + V_{jt}(\tau) = \begin{cases} 0 & u = \bar{u} \\ \geq 0 & u \neq \bar{u} \end{cases} \quad (5)$$

with

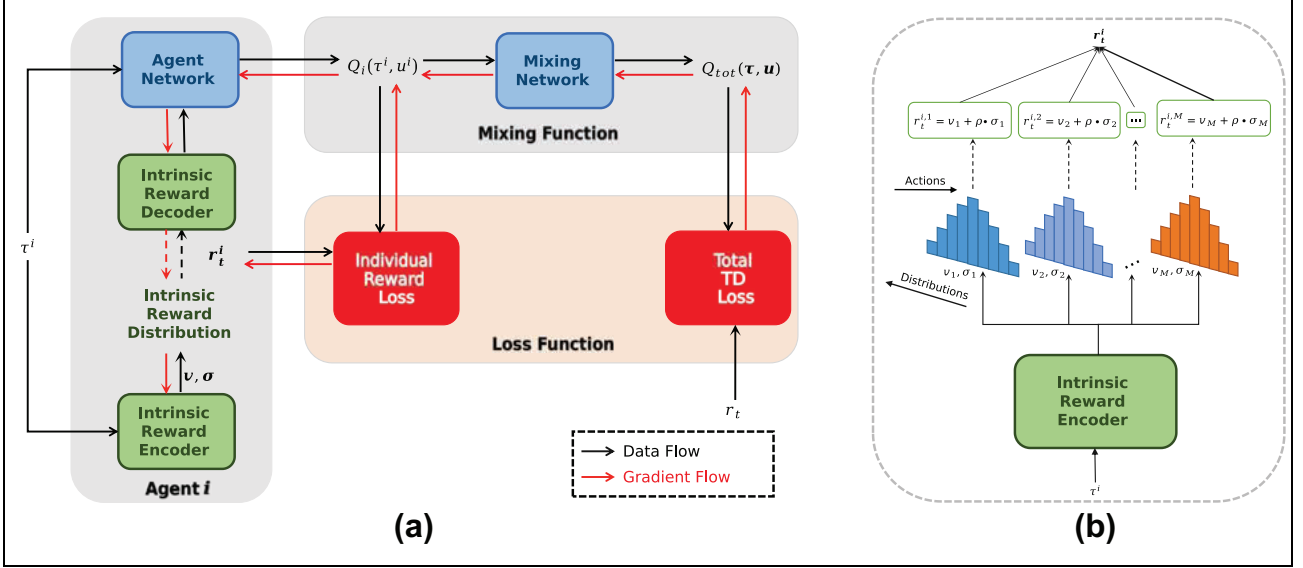
$$V_{jt}(\tau) = \max_u Q_{jt}(\tau, u) - \sum_{i=1}^N Q_i(\tau^i, \bar{u}^i)$$

where  $\bar{u}^i$  is the optimal local action  $\operatorname{argmax}_{u^i} Q_i(\tau^i, u^i)$ ,  $\bar{u} = [\bar{u}_i]_{i=1}^N$ , and  $V_{jt}(\tau)$  is the parameterized state value function.

Thereby, the IAVs  $Q_i(o_t^i, \mu_t^i)$  can be optimized end-to-end by optimizing the JAV  $Q_{jt}(\tau_t, \mu_t)$  in the above decomposition methods.

### Learning individual intrinsic reward

LIIR<sup>15</sup> learns each agent an IIR  $r_{i,t}^{\text{in}}$  and combines the IIR and the global reward  $r_t$  to obtain a proxy value function for each agent



**Figure 1.** (a) Architecture of GIIR. The *intrinsic reward encoder* generates  $M$  IIR distributions, and a  $M$ -dimensional IIR is sampled from the generated distributions. The *intrinsic reward decoder* (i.e. *hypernetworks*) generates the parameters for the agent network. Then the agent network will output IAV  $Q_i(\tau^i, u^i)$ , and the IAVs of all agents are mixed into a JAV  $Q_{tot}(\tau, u)$  by a mixing network. Besides, the total TD loss and the individual reward loss are used to optimize the parameters, and the framework can be trained in an end-to-end manner. (b) The generation of the IIR. The *intrinsic reward encoder* generates  $M$ -dimensional average and variance ( $v, \sigma^2$ ), i.e. there are  $M$  intrinsic reward distributions for  $M$  actions. GIIR: generating individual intrinsic reward; IIR: individual intrinsic reward; IAV: individual action value; JAV: joint action value; TD: Time Difference.

$$V_i^p(o_{i,t}) = \mathbb{E}_{\mu_{i,t}, o_{i,t+1}, \dots} \left[ \sum_{l=0}^{\infty} \gamma^l (r_{t+l} + \lambda r_{i,t+l}^{\text{in}}) \right] \quad (6)$$

where  $\mu_{i,t}$  is the local action of an agent  $i$  at timestep  $t$ ,  $o_{i,t}$  is the local observation of agent  $i$  at timestep  $t$ , and  $\lambda$  is a hyperparameter to balance the extrinsic global reward and the IIR.

Next, the proxy value function is used to optimize the policy of each actor (i.e. each agent)

$$L(\phi_i) = \log \pi_{\phi_i}(\mu_{i,t} | o_{i,t}) \left( r_t + \lambda r_{i,t}^{\text{in}} + V^p(o_{i,t+1}) - V^p(o_{i,t}) \right) \quad (7)$$

where  $\pi_{\phi_i}(\mu_{i,t} | o_{i,t})$  is the policy of agent  $i$ ,  $\phi_i$  represents the parameters of the actor-network, and  $r_t + \lambda r_{i,t}^{\text{in}} + V^{\text{proxy}}(o_{i,t+1}) - V^{\text{proxy}}(o_{i,t})$  is the advantage function.<sup>17</sup> Thereby, LIIR builds a connection between the learned IIR and the actor-critic algorithm to address the lazy agent problem.

## Method

In this section, we propose a GIIR algorithm, which constructs a connection between the IIR and the decomposition methods in the cooperative MARL field.

The main idea is that through optimizing by maximizing the expected global reward, the learned IIR can guide each agent to perform the action that can obtain a greater global reward by participating in the estimation of the IAV.

We assume that there exists an IIR function  $R_i^i(s_t, \mu_t^i)$  for each agent. Hence, we introduce an *intrinsic reward encoder*  $E(r_i | \tau^i; \theta_r)$  with parameters  $\theta_r$  to generate the IIR. Note that the IIR function  $R_i^i(s_t, \mu_t^i)$  gives the reward  $r_t^i$  by performing action  $\mu_t^i$  in the state  $s_t$ , which should be based on the state and action. Because each agent can only access the partial local observation in the Decentralized Partial Observation Markov Decision Process (Dec-POMDP) setting, we apply a Gate Recurrent Unit Recurrent Neural Networks (GRU)<sup>30</sup> to process the local observation history  $\tau^i$  in the *intrinsic reward encoder*, which is a technique to handle the partial observability.<sup>27</sup> Hence, the  $s_t$  of the IIR function is replaced by the local observation history  $\tau^i$ . Besides, to perform decentralized policies during execution time, we only take the local observation history as the inputs of the *intrinsic reward encoder* but generate  $M$  IIR distributions for  $M$  actions (i.e.  $M$ -dimensional average  $v = \{v_1, v_2, \dots, v_M\}$  and variance  $\sigma^2 = \{\sigma_1^2, \sigma_2^2, \dots, \sigma_M^2\}$  of distribution), where  $M$  is the size of individual action space. In other words, the *intrinsic reward encoder* will generate a single intrinsic reward distribution for each action. Then, we apply the reparameterization trick<sup>31</sup> to sample  $M$ -dimensional IIR from  $M$  IIR distributions. Specifically, as shown in Figure 1(b), we sample a random

**Table 1.** SMAC scenarios.

Name	Ally units	Enemy units	Type
5m_vs_6m	5 Marines	6 Marines	Homogeneous
8m_vs_9m	8 Marines	9 Marines	Homogeneous
10m_vs_11m	10 Marines	11 Marines	Homogeneous
1c3s5z	1 Colossi, 3 Stalkers, 5 Zealots	1 Colossi, 3 Stalkers, 5 Zealots	Heterogeneous
MMM	1 Medivac, 2 Marauders, 7 Marines	1 Medivac, 2 Marauders, 7 Marines	Heterogeneous
MMM2	1 Medivac, 2 Marauders, 7 Marines	1 Medivac, 3 Marauders, 8 Marines	Heterogeneous

variable  $\rho$  from the norm distribution  $N(0, 1)$  and then compute the generated IIR for each action

$$r_t^{i,j} = v_j + \rho \cdot \sigma_j \quad (8)$$

where  $j \in \{1, 2, \dots, M\}$  represents the number of actions in action space and  $i \in \{1, 2, \dots, N\}$  is the agent ID. Finally, all generated one-dimensional  $r_t^{i,j}$  are stacked in a  $M$ -dimensional  $r_t^i$ . By the means of the reparameterization trick, the parameters  $\theta_r$  can be trained end-to-end, which is shown in Figure 1(a).

Similar to LIIR,<sup>15</sup> the generated IIR  $r_t^i$  does not have any physical meanings. To apply  $r_t^i$  to guide the estimation of the IAV, we introduce *hypernetworks*<sup>18</sup> with parameters  $\theta_h$  as the *intrinsic reward decoder*, which takes  $r_t^i$  as the inputs and outputs the parameters  $\theta_a$  of *agent networks*. Then, the *agent network* can estimate the IAV  $Q_i(\tau^i, u^i; \theta_i)$  based on the local observation history, where  $\theta_i = (\theta_a, \theta_h, \theta_r)$ .

Next, the IAVs of all agents will be mixed into a JAV  $Q_{jt}(\tau, u)$  by a *mixing network* with parameters  $\theta_m$

$$Q_{jt}(\tau, u; \theta_m, \theta_i) = \text{MIX}(Q_1(\tau^1, u^1; \theta_i), \dots, Q_N(\tau^N, u^N; \theta_i)) \quad (9)$$

which can be abbreviated as  $\text{MIX}(Q_i(\tau^i, u^i; \theta_i))$ . Note that to speed the training by sharing parameters, all agents use the same *agent network* but with different inputs.<sup>6</sup> Besides, we use the *mixing network* that is the same as that in QMIX<sup>7</sup> in this work. Specifically, the parameters of *mixing network* are outputted by *hypernetworks* based on the global state  $s_t$ , and the parameters are restricted to be positive to ensure monotonicity.

Then, we can use the global reward  $r_t$  to optimize the JAV

$$L_{tot}(\theta_m, \theta_i) = \left( r_t + \gamma Q_{jt}^{\max}(\tau, u; \theta_m^-, \theta_i^-) - Q_{jt}(\tau, u; \theta_m, \theta_i) \right)^2 \quad (10)$$

with

$$Q_{jt}^{\max}(\tau, u; \theta_m^-, \theta_i^-) = \text{MIX}_{\theta_m} \left( \max_{u^i} Q_i(\tau^i, u^i; \theta_i^-) \right)$$

$$Q_{jt}(\tau, u; \theta_m, \theta_i) = \text{MIX}_{\theta_m} (Q_i(\tau^i, u^i; \theta_i))$$

where  $\theta_m^-$  and  $\theta_i^-$  are the parameters of target networks that are similar to Equation (1).

Since the IAV of the decomposition methods can reflect each agent's contribution to the global reward to some extent,<sup>5</sup> we can apply the IAV to train each agent's IIR  $r_t^i$

$$L_r(\theta_r) = \sum_{i=1}^N \left( r_t^i + \gamma \max_{u^i} Q_i(\tau^i, u^i; \theta_i^-) - Q_i(\tau^i, u^i; \theta_i) \right)^2 \quad (11)$$

where  $r_t^i$  is parameterized by  $\theta_r$  that it is generated by the *intrinsic reward encoder*, and  $N$  is the number of agents. Note that only the parameters  $\theta_r$  of the generated IIR  $r_t^i$  are optimized in Equation (11). In other words,  $Q_i(\tau^i, u^i; \theta_i^-)$  and  $Q_i(\tau^i, u^i; \theta_i)$  are used as the constant and the parameters of them will not be optimized with the  $L_r(\theta_r)$ , although  $\theta_i = (\theta_a, \theta_h, \theta_r)$ .

Finally, the gradient of the overall loss function is as follows

$$\nabla_{\theta_m, \theta_i} L(\theta_m, \theta_i) = \nabla_{\theta_m, \theta_i} L_{tot}(\theta_m, \theta_i) + \nabla_{\theta_r} L_r(\theta_r) \quad (12)$$

## Experiment

In this section, we conduct the experiments on a benchmark named SMAC<sup>19</sup> to show the performance improvement of the proposed GIIR.

### Environmental setup

SMAC provides a set of fully cooperative multiagent scenarios, which focus on the decentralized micromanagement of the real-time strategy game StarCraft II. Namely, all units (ally units) are controlled by MARL agents to defeat another group of units (enemy units). The enemy units are controlled by the built-in game AI with difficulty from *very easy* to *cheat insane*, and we set the difficulty as *very difficult* in this work. Besides, SMAC considers the partial observability setting by introducing the *sight range*, in which each unit can only access the local observation with the field of the view.

To evaluate the policy performance of the proposed GIIR, we conduct the comparison experiments in three homogeneous scenarios (5m\_vs\_6m, 8m\_vs\_9m, and 10m\_vs\_11m) and three heterogeneous scenarios (1c3s5z,

MMM, and MMM2). The list of scenarios considered in our experiment is presented in Table 1, and the screenshots of the six scenarios are shown in the Online Appendix. We evaluate the proposed method and the comparison method across 10 independent runs with different seeds and run 20 independent test episodes every 20,000 timesteps training to calculate the percentage of winning episodes as the *win rates*, in which the winning episodes are those that all enemy

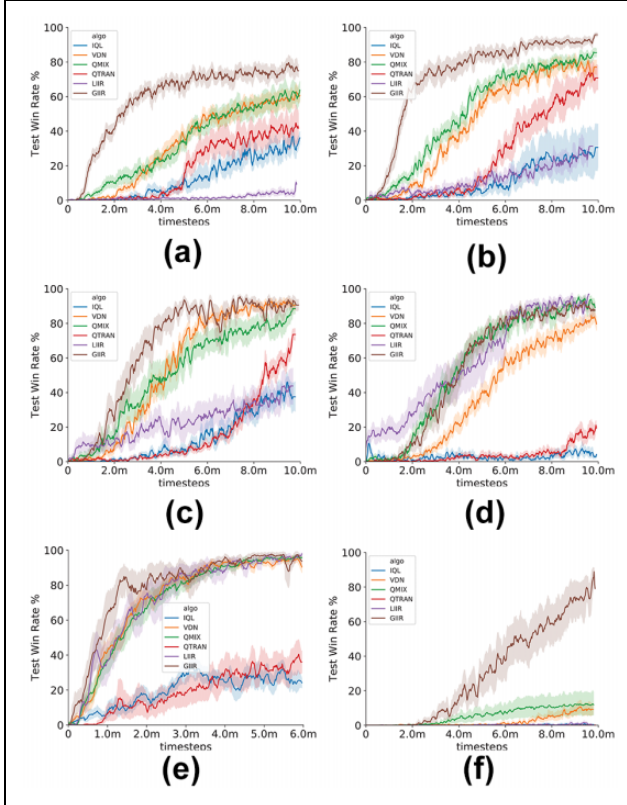
units are defeated within a time limit. All methods are evaluated after 10 million training timesteps, and the exception is the MMM, which is an easy scenario and only be evaluated after 6 million training timesteps. More experimental details are presented in the Online Appendix.

### SMAC: StarCraft multiagent challenge.

#### Comparison results

We compare the proposed GIIR with decomposition methods (i.e. VDN, QMIX, and QTRAN) and LIIR. Besides, we also compare it with the IQL, which learns the IAV based on the global reward directly.

The learning curves are shown in Figure 2 and the experimental statistic results are shown in Table 2. Because ignoring the coordination among agents and employing the global reward directly as the individual reward to learn per-agent IAV, the performance of IQL is almost the worst in all scenarios. Decomposition methods learn IAV implicitly through end-to-end training from the JAV decomposition, the performance has gain a great improvement, in which the performance of QMIX is slightly better than that of VDN, and QTRAN indeed performs poorly in most SMAC scenarios according to Wen et al.<sup>32</sup> The experiments of Du et al.<sup>15</sup> have shown that LIIR can outperform decomposition methods in many easy scenarios, which are similar to the Figure 2(d) and (e). This proves that learning each agent an IIR can help to improve performance. However, when the scenarios become more difficult, the performance of LIIR is poor, and even is worse than that of the decomposition methods, which are shown in Figure 2(a) to (c). Compared with that, GIIR combines the advantages of the decomposition methods and LIIR, which introduces the IIR into decomposition methods. The experimental results show that it can almost obtain better performance than all comparison methods, in which the performance of GIIR ( $89 \pm 8\%$ ) is only slightly worse than that of QMIX ( $88 \pm 9\%$ ) in scenario 1c3s5z. Most importantly, GIIR can



**Figure 2.** Test win rates of IQL, VDN, QMIX, QTRAN, and GIIR on six scenarios. (a) 5m\_vs\_6 m, (b) 8m\_vs\_9 m, (c) 10m\_vs\_11 m, (d) 1c3s5z, (e) MMM, and (f) MMM2. IQL: individual Q-learning; VDN: value-decomposition network.

**Table 2.** Mean, SD, and median of test win rates in six SMAC scenarios.

Scenarios		IQL	VDN	QMIX	QTRAN	LIIR	GIIR
5m_vs_6 m	Mean $\pm$ SD	36 $\pm$ 8	61 $\pm$ 12	64 $\pm$ 16	42 $\pm$ 19	6 $\pm$ 5	<b>75 <math>\pm</math> 9</b>
	Median	42	67	69	46	9	<b>75</b>
8m_vs_9 m	Mean $\pm$ SD	31 $\pm$ 29	77 $\pm$ 13	85 $\pm$ 8	71 $\pm$ 16	27 $\pm$ 6	<b>96 <math>\pm</math> 4</b>
	Median	13	75	83	67	31	<b>96</b>
10m_vs_11 m	Mean $\pm$ SD	38 $\pm$ 21	90 $\pm$ 9	88 $\pm$ 7	66 $\pm$ 15	43 $\pm$ 22	<b>91 <math>\pm</math> 8</b>
	Median	35	92	90	69	56	<b>96</b>
1c3s5z	Mean $\pm$ SD	3 $\pm$ 2	80 $\pm$ 11	<b>89 <math>\pm</math> 8</b>	17 $\pm$ 9	86 $\pm$ 21	88 $\pm$ 9
	Median	4	83	87	21	87	<b>90</b>
MMM	Mean $\pm$ SD	22 $\pm$ 10	90 $\pm$ 13	96 $\pm$ 4	36 $\pm$ 13	<b>97 <math>\pm</math> 3</b>	<b>97 <math>\pm</math> 7</b>
	Median	25	96	96	29	96	<b>97</b>
MMM2	Mean $\pm$ SD	0 $\pm$ 0	9 $\pm$ 11	12 $\pm$ 28	0 $\pm$ 0	0 $\pm$ 0	<b>84 <math>\pm</math> 15</b>
	Median	0	4	0	0	0	<b>92</b>

SD: standard deviation; SMAC: StarCraft multiagent challenge; IQL: individual Q-learning; VDN: value-decomposition network; LIIR: learning individual intrinsic reward; GIIR: generating individual intrinsic reward.



obtain better performance in the superhard scenario MMM2, in which neither decomposition methods nor LIIR can handle it.

## Conclusion

In this article, we propose GIIR algorithm, which constructs a connection between the decomposition and the learned IIR to address the lazy agent problem. The proposed GIIR generates an IIR for each agent to help to confirm per-agent contribution to the global success. Besides, the generated IIR is optimized by maximizing the expected global reward, thus it can help to obtain greater teamwork success. Our experimental results in SMAC prove that GIIR improves the final performance over both the decomposition methods and LIIR in cooperative tasks.

In future work, we will focus on studying other methods to better utilize the generated intrinsic reward to estimate the IAVs. Furthermore, we will conduct additional experiments on other SMAC scenarios with a larger number and greater diversity of units. Moreover, we will apply the proposed method to actual multiagent system scenarios, such as the logistics robot task, and conflict resolution in the air traffic control task.


## Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

## ORCID iD

Hui Li  <https://orcid.org/0000-0002-7333-6942>

## Supplemental material

Supplemental material for this article is available online.

## References

1. Ying W and Dayong S. Multi-agent framework for third party logistics in e-commerce. *Expert Syst Appl* 2005; 29(2): 431–436.
2. Xu Z, Lyu Y, Pan Q, et al. Multi-vehicle flocking control with deep deterministic policy gradient method. In *2018 IEEE 14th international conference on control and automation (ICCA)*, Anchorage, Alaska, USA, June 12–15, 2018. IEEE, pp. 306–311.
3. Cao Y, Yu W, Ren W, et al. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Trans Industr Inform* 2012; 9(1): 427–438.
4. Ye D, Zhang M, and Yang Y. A multi-agent framework for packet routing in wireless sensor networks. *Sensors* 2015; 15(5): 10026–10047.
5. Sunehag P, Lever G, Gruslys A, et al. Valuedecomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th international conference on autonomous agents and multiagent systems*. AAMAS 18, 2018, pp. 2085–2087. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.
6. Foerster J, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients. In *Proceedings of the thirty-second AAAI conference on artificial intelligence*, New Orleans, Louisiana, USA, February 2–7, 2018. 2018.
7. Rashid T, Samvelyan M, de Witt CS, et al. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J Mach Learn Res* 2020; 21(178): 1–51.
8. Son K, Kim D, Kang WJ, et al. QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *The 36th international conference on machine learning*, Long Beach, California, USA, 9–15 June 2019. pp. 5887–5896.
9. Ng A, Harada D, and Russell S. Policy invariance under reward transformations: theory and application to reward shaping. *Proceedings of the sixteenth international conference on machine learning*, San Francisco, CA, United States, June 27–30, 1999.
10. Devlin S, Yliniemi L, Kudenko D, et al. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the 2014 international conference on autonomous agents and multi-agent systems*, Paris, France, 05, 2014–May 09, 2014, pp. 165–172.
11. Eck A, Soh LK, Devlin S, et al. Potential-based reward shaping for finite horizon online pomdp planning. *Auton Agents Multi-Agent Syst* 2016; 30(3): 403–445.
12. Zheng Z, Oh J, and Singh S. On learning intrinsic rewards for policy gradient methods. *arXiv preprint arXiv:180406459*. 2018.
13. Bahdanau D, Hill F, Leike J, et al. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:180601946*. 2018.
14. Zheng Z, Oh J, Hessel M, et al. What can learned intrinsic rewards capture? In *Proceedings of the 37th international conference on machine learning, proceedings of machine learning research*, PMLR, 2020, 119: 11436–11446.
15. Du Y, Han L, Fang M, et al. LIIR: learning individual intrinsic reward in multi-agent reinforcement learning. *Adv Neural Inf Process Syst* 2019; 32: 4403–4414.
16. Mnih V, Badia AP, Mirza M, et al. Asynchronous methods for deep reinforcement learning. In Balcan MF and Weinberger KQ (eds) *Proceedings of the 33rd international conference on machine learning*, PMLR, 2016; 48: 1928–1937. New York, USA.
17. Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization. In Bach F and Blei D (eds) *Proceedings of the 32nd international conference on machine learning*, PMLR, 2015; 37: 1889–1897. Lille, France.
18. Ha D, Dai AM, and Le QV. Hypernetworks. In *5th international conference on learning representations*, ICLR, Toulon, France, April 24–April 26 2017.
19. Samvelyan M, Rashid T, de Witt CS, et al. The starcraft multi-agent challenge. *CoRR* 2019; abs/1902.04043.

20. Claus C and Boutilier C. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI* 1998; 1998(746–752): 2.
21. Jiang J and Lu Z. The emergence of individuality in multi-agent reinforcement learning. *arXiv preprint arXiv:200605842*. 2020.
22. Bellemare MG, Srinivasan S, Ostrovski G, et al. Unifying count-based exploration and intrinsic motivation. *arXiv preprint arXiv:160601868*. 2016.
23. Savinov N, Raichuk A, Marinier R, et al. Episodic curiosity through reachability. *arXiv preprint arXiv:181002274*. 2018.
24. Houthoofd R, Chen X, Duan Y, et al. VIME: variational information maximizing exploration. *arXiv preprint arXiv:160509674*. 2016.
25. Pathak D, Agrawal P, Efros AA, et al. Curiositydriven exploration by self-supervised prediction. In *Proceedings of the 34th international conference on machine learning, ICML*, 2017; 70: 2778–2787.
26. Oliehoek FA and Amato C. *A concise introduction to decentralized POMDPs*. Berlin: Springer, 2016.
27. Hausknecht M and Stone P. Deep recurrent q-learning for partially observable mdps. *arXiv: Learning*. 2015.
28. Hochreiter S and Schmidhuber J. Long short-term memory. *Neural Comput* 1997; 9(8): 1735–1780.
29. Tampuu A, Matiisen T, Kodelja D, et al. Multiagent cooperation and competition with deep reinforcement learning. *PLoS one* 2017; 12(4): e0172395.
30. Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:14061078*. 2014.
31. Kingma DP and Welling M. Auto-encoding variational bayes. *arXiv preprint arXiv:13126114*. 2013.
32. Wen C, Yao X, Wang Y, et al. Smix( $\lambda$ ): Enhancing centralized value functions for cooperative multi-agent reinforcement learning. *Proc Conf AAAI Artif Intell* 2020; 34: 7301–7308. DOI:10.1609/aaai.v34i05.6223.