Inducing Cooperation through Reward Reshaping based on Peer Evaluations in Deep Multi-Agent Reinforcement Learning

David Earl Hostallero[*]
McGill University
Montreal, Canada
david.hostallero@mail.mcgill.ca

Daewoo Kim
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
dwkim@lanada.kaist.ac.kr

Sangwoo Moon
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
swmoon@lanada.kaist.ac.kr

Kyunghwan Son
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
khson@lanada.kaist.ac.kr

Wan Ju Kang
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
wjkang@lanada.kaist.ac.kr

Yung Yi
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
yiyung@kaist.edu

ABSTRACT

We propose a deep reinforcement learning algorithm for semi-cooperative multi-agent tasks, where agents are equipped with their separate reward functions, yet with some willingness to cooperate. It is intuitive that defining and directly maximizing a global reward function leads to cooperation because there is no concept of selfishness among agents. However, it may not be the best way of inducing such cooperation due to problems that arise from training multiple agents with a single reward (*e.g.*, credit assignment). In addition, agents may intentionally be given separate reward functions to induce task prioritization whereas a global reward function may be difficult to define without diluting the effect of different tasks and causing their reward factors to be disregarded. Our algorithm, called Peer Evaluation-based Dual DQN (PED-DQN), proposes to give *peer evaluation signals* to observed agents, which quantify how they strategically value a certain transition. This exchange of peer evaluation among agents over time turns out to render agents to gradually reshape their reward functions so that their action choices from the myopic best response tend to result in a more cooperative joint action.

CCS CONCEPTS

• **Computing methodologies → Multi-agent systems**; **Cooperation and coordination**; **Multi-agent reinforcement learning**;

ACM Reference Format:
David Earl Hostallero, Daewoo Kim, Sangwoo Moon, Kyunghwan Son, Wan Ju Kang, and Yung Yi. 2020. Inducing Cooperation through Reward Reshaping based on Peer Evaluations in Deep Multi-Agent Reinforcement Learning. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

[*]Work done in Korea Advanced Institute of Science and Technology

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand. © 2020 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

1 INTRODUCTION

Cooperation is a core challenge in many multi-agent scenarios. Many reinforcement learning methods have been proposed to solve sequential decision making for single agents, but they do not directly translate to multi-agent settings due to the simultaneously evolving policies. These simultaneous policy updates make the environment's transition probabilities non-stationary, which in turn makes learning difficult for the agents because a slight change in one's policy may cause another agent's policy to perform sub-optimally.

There exist many studies examining fully cooperative cases, corresponding to when agents agree to cooperate unconditionally, *e.g.*, being controlled by a single authority. In these cases, a set of agents usually learn to maximize a global[1] reward and thus the reward function itself can naturally induce a certain level of cooperation. Recent related work for fully cooperative tasks includes COMA [5], VDN [25], QMIX [20], CommNet [24], DIAL [2], PS-TRPO [7], Mordatch and Abbeel [16], Havrylov and Titov [8], LDQN [18], just to name a few.

However, other types of of tasks exist where agents have their own interests but may benefit from cooperation. An example is sequential social dilemmas [12]. It may also be the case that agents are intentionally given separate rewards to prioritize different tasks. In this paper, we consider such "semi-cooperative" tasks where agents may each have its own separate reward function (the joint reward scenario is a special case), but is willing to cooperate if an incentive for cooperation is appropriately provided.[2] In other words, semi-cooperative tasks are those with separate per-agent reward functions but whose agents may nonetheless benefit from cooperation. As such, zero-sum games are by nature not semi-cooperative. This may occur when agents are controlled by different administrative domains, in which some administrators may choose to prioritize some reward factors over others. The key challenge in semi-cooperative tasks is mainly due to the selfishness represented by separate reward functions, which often results in agents' choice

[1]Global reward means that each agent receive the same (scalar) reward value for each step. We use this interchangeably with "joint reward."
[2]In cooperative game theory, Nash bargaining solution [17] and Shapley value [21] are the axiomatic rules for a fair distribution of such cooperation incentive. The way of allocating the final cooperation gain to each agent is beyond the scope of this paper.

of non-cooperative actions (*e.g.*, in prisoner's dilemma, social optimum differs from Nash equilibrium). The goal is for the agents to find cooperative policies (which may not be an equilibrium *e.g.*, as in Prisoner's dilemma) that will maximize the *social welfare* (SW), which, for the purpose of this study, we define as the sum of the rewards of each agent across the entire episode. Since the goal is to maximize SW, one could artificially transform the semi-cooperative task into a fully cooperative one by defining the summation of individual rewards as the joint reward function, thereby summing up to the SW when accumulated across the entire episode. However, it is known that this vanilla conversion suffers from the signal-to-noise problem [30], which makes it difficult for the agents to associate their actions to the reward. In addition, Singh et al. [22] have shown that individual rewards lead to faster convergence. We argue that a simplistic summing of the rewards is impractical since (1) it disregards the reward assignment and (2) it essentially overlooks the misalignments in rewards, which potentially could have characterized the difference between agents' priorities.

1.1 Contribution

In this paper, we propose a value-based deep MARL method for semi-cooperative tasks, called PED-DQN (Peer-Evaluation based Dual-DQN). PED-DQN gradually reshapes the rewards in a distributed manner such that the agents' perception of the equilibrium gears towards the outcome that optimizes the SW to avoid the impracticalities that would arise in using the SW itself as the reward. Our method remains free from constraints on the number of reshaping since agents are reshaping rewards distributedly, with only peer evaluation signals and local observations as guide. The key idea is to address the problem from the mechanism design perspective in game theory. In semi-cooperative sequential tasks, agents will naturally choose the action that is the local best response to its current observation and local reward experience. In our approach, we gradually change the game that the agents play so that their local-view actions from best responses become closer to more cooperative joint actions. To this end, each agent receives simple peer evaluations in the form of feedback messages from other agents. These peer evaluations quantify how an agent "feels" about a certain transition, rendering each agent reshape its reward function, so as for agents' local best responses to result in global cooperation. We instantiate our idea by developing a learning mechanism consisting of two DQNs (Mission DQN and Action DQN) per agent, where Mission DQN calculates peer evaluations with respect to the agent's own reward, and Action DQN selects actions with consideration of the previously received peer evaluations. In execution, each agent is equipped with its own action DQN and executes in a fully distributed manner.

This approach is expected to efficiently address many semi-cooperative tasks without explicitly forcing full cooperation, as in the methods of using the sum of individual rewards as a global, shared reward, thus allowing an easier per-agent action-to-reward association for sample-efficient training. Our method does not necessarily require a centralized module as in many recent works based on CTDE (Centralized Training with Distributed Execution), *e.g.*, COMA [5], VDN [25], QMIX [20], DIAL [2], and Multiagent Soft Q-Learning [29]. This allows easier distributed implementation for large-scale training and entails smaller exploration complexity unlike centralized modules in many CTDE methods that are affected by the permutation dependence of multi-agent actions and states. Finally, our approach is flexible since it does not require the knowledge of the number of agents beyond the agent's observation. The source code of our algorithm is downloadable in this link[3].

1.2 Related Work

One of the earliest approaches in decentralized deep MARL used independent DQN [26] to study the cooperation and competition among agents, depending on their reward functions. We concern ourselves with the cooperative end of the spectrum, where cooperative strategies can benefit the agents. Many studies considered fully cooperative multi-agent systems, where agents are expected to learn policies that directly maximize a global reward. Recent popular works that address joint rewards and some key challenges (*e.g.*, non-stationarity), utilize a centralized optimization module. COMA [5] proposed a counterfactual advantage function for better credit assignment and thus more efficient training. VDN [25], QMIX [20], and QTRAN [23] proposed value function decomposition among agents to induce cooperation. MADDPG [15] is an exception to the line of research handling a joint reward with a centralized module. However, MADDPG [15] still relies on a sort of pseudo-centralization, where each agent maintains its own centralized critic. Foerster et al. [4] adopts multi-agent importance sampling to modify the workings of the experience replay in a cooperative setting. Foerster et al. [6] approximates the joint belief of the hidden environment state as a product of individual beliefs, also in a fully cooperative setting. We note that this paper aligns with VDN [25], QMIX [20], and QTRAN [23] in that the four are all value-based methods, but the key distinction is that we consider the semi-cooperative tasks.

Although maximizing a joint reward is the goal, it is not always the case that a joint reward is given. Leibo et al. [12] studied the environment where each agent has a separate reward function. In particular, agents face a social dilemma where mutual cooperation may give them higher rewards, but they have no *a priori* guarantee of any other agent's cooperation. Jaques et al. [10] modified the agent's reward by incentivizing agents that have influential behavior. They claim that an agent's behavior itself is a form of communication, and that having influence on other agents' decision process allows agents to find coordinated policies. SOS [13] and LOLA [3] used opponent shaping, where each agent takes into consideration that other agents are also learning by anticipating the other agents' policy updates. These works focus on how an agent can modify its policy such that other agents who are reactively updating their policies can be shaped according to the agent's selfish reward. This can be seen as a "passive-aggressive" way of influencing others' policies whereas our approach can be seen as a direct confrontation since agents directly receive a form of incentive/penalty from their peers.

Hughes et al. [9] and Eccles et al. [1] attempt to solve these dilemmas by modifying the agents' reward to include factors that depend on other agents' rewards. These studies primarily target fairness; agents are penalized for having significantly different rewards

[3]https://github.com/ddhostallero/PED-DQN

compared to others'. Hughes et al. [9] incorporate a penalty term for unfair reward distribution among the agents. Eccles et al. [1] suggested a stochastic game extension of the tit-for-tat approach, where a niceness network is designed, calculating the expected change in the discounted rewards of the agents. The niceness network is highly relevant to our peer evaluation mechanism, but differs from our work in that we induce cooperation by maximizing the sum of rewards, rather than making the agents get a fair share of rewards.

Studies that share the common goal with ours, *i.e.*, maximizing the social welfare, include Peysakhovich and Lerer [19] and Wang et al. [27]. Peysakhovich and Lerer [19] considered a stag-hunt game and proved that modifying the reward by a weighted sum of one's and its opponent's rewards helps in choosing the cooperative actions. Wang et al. [27] introduced the genetic algorithm of evolving the reward network given to the agents, and the reward network is trained to maximize the overall sum of rewards. Our approach also takes into account the social welfare of the agents as in [19], but it is applicable to more general games. The difference with our approach is that ours uses local information in a distributed setting instead of a centralized arena of agents. Moreover, our method uses a more direct signal of peer evaluation, rather than using a genetically evolving reward network as in [27], thereby leading to better sample efficiency.

2 MODEL AND GOAL

Stochastic game. We consider a multi-agent task with each agent having its own individual reward by modeling it with a stochastic game [14]. We consider n agents, indexed by $a \in \mathcal{A} = \{1, 2, ..., n\}$ and denote $-a$ as the set of other agents $\mathcal{A} \setminus a$. Each agent a has a set of actions U_a. At each time step, each agent has a partial observation o_a according to observation function $O(s, a)$ where $s \in \mathcal{S}$ is the true state. It conditions its policy $\pi_a(u_a|o_a)$ on its observation to decide its action $u_a \in U_a$. Let $\boldsymbol{\pi} = [\pi_a]_{a\in\mathcal{A}}$. The set of observations from all agents at a time is denoted by O. The joint action of all the agents is denoted by $\boldsymbol{u} \in \mathcal{U}$, where $\mathcal{U} := U_1 \otimes \cdots \otimes U_n$ is the set of all joint actions. Once each agent takes its action, forming a joint action, say $\boldsymbol{u}$, a transition to the next state occurs, following a transition function $P(s'|s, \boldsymbol{u})$. Each agent then receives a reward $r_a(s, \boldsymbol{u})$ that may differ across agents and γ is the discount factor.

Semi-cooperative tasks. We define semi-cooperative tasks as the set of tasks where each agent may have a separate reward function but may benefit from cooperative strategies such as the prisoner's dilemma and the staghunt game.

Goal: Maximizing the social welfare. In semi-cooperative tasks, we aim to achieve maximum cooperation, which we quantify by obtaining the following socially optimal policy:

$$\boldsymbol{\pi}^* = \left(\pi_1^*(u_1|o_1), \pi_2^*(u_2|o_2), \ldots, \pi_n^*(u_n|o_n)\right)$$

across agents, *i.e.*, maximizing *social welfare* defined by the sum of the (discounted) rewards of all agents over the entire episode:

$$\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} \sum_{a\in\mathcal{A}} \mathbb{E}_{s\sim\rho^\pi, \boldsymbol{u}\sim\boldsymbol{\pi}}[r_a(s, \boldsymbol{u})]$$

where $\mathbb{E}_{s\sim\rho^\pi}[\cdot]$ denotes the expected reward with respect to discounted state distribution ρ^π by initial state distribution and state transition dynamics distribution.

3 METHODS

3.1 Change of Games via Reward Reshaping

In the execution environment, where each agent only has access to its partial observation, it is natural that each agent chooses its action based on its local best response. In the semi-cooperative task with individual rewards, local best responses often lead to social dilemma. Our idea is to change the game by reshaping agents' reward functions over time, so that the agents' actions from the local best responses become socially optimal ones that maximize the cooperation effect. We call such reshaped rewards well-coordinated rewards.

Denote by G^t and $\hat{\boldsymbol{r}}^t = (\hat{r}_a^t : a \in \mathcal{A})$ the game and its reward function at time step $t = 0, 1, \cdots$. We initialize $t = 0$ and then run the following loop in the training:

(i) Compute the optimal policy π^t from G^t
(ii) Evaluate how well-coordinated G^t is by evaluating π^t
(iii) Update from G^t to G^{t+1} by updating from $\hat{\boldsymbol{r}}^t$ to $\hat{\boldsymbol{r}}^{t+1}$, using the 'evaluation feedback' from (*ii*)
(iv) Increment t and go to (*i*).

Through these updates, we hope that G^t gradually changes into a better coordinated one. We later elaborate on the details on the evaluation feedback and the implementation based on multiple DQNs.

Running policy computation from a given game G^t in (*i*) and reward update in (*iii*) is expected to require significantly long convergence time, because in large-scale tasks with many agents, just computing the optimal policy in (*i*) alone would be challenging in training. Motivated by this, we take an approach of running the following two updates in parallel.

$$\textbf{Policy Update:} \quad \boldsymbol{\pi}^{t+1} = F(\boldsymbol{\pi}^t, \hat{\boldsymbol{r}}^t) \tag{1}$$

$$\textbf{Reward Update:} \quad \hat{\boldsymbol{r}}^{t+1} = H(\hat{\boldsymbol{r}}^t, \boldsymbol{\pi}^t), \tag{2}$$

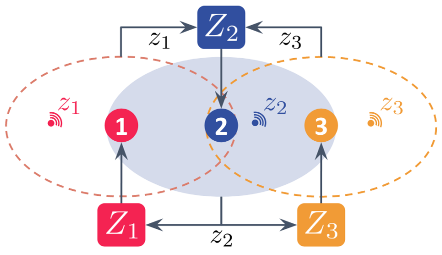
where we conceptually denote by F and H the mappings in policy and reward updates. Note that we may take different learning rates (thus different operation time-scale) in F and H for a more stable training, as done in many algorithms with multiple parallel updates, *e.g.*, actor and critic algorithms [11]. In the policy update F, each agent's policy $\boldsymbol{\pi}^t$ is updated to maximize the reshaped reward $\hat{\boldsymbol{r}}^t$ at that time that can be done by a reinforcement learning algorithm, *e.g.*, DQN. In the reward update H, agents estimate the value of current policies and update the reshaped reward appropriately.

3.2 Reward Update with Peer Evaluation

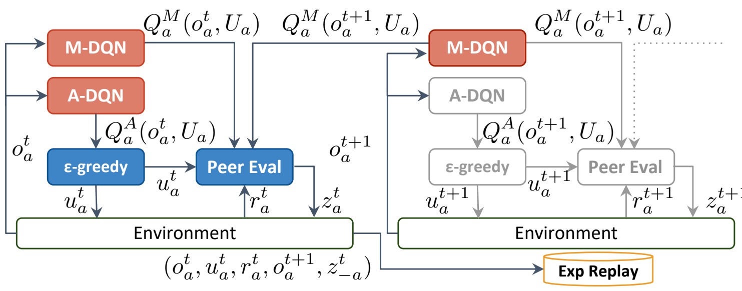
Peer evaluation signal. Given the current policy $\boldsymbol{\pi}^t$ and sampled trajectory $(\boldsymbol{o}^t, \boldsymbol{o}^{t+1}, \boldsymbol{u}^t, \boldsymbol{r}^t)$, each agent k generates a *counterfactual evaluation signal* (CES) z_k^t for the observed transition $o_k^t \to o_k^{t+1}$. For a given observed transition, action u_k^t, and base reward $r_k^t = r_k(s^t, \boldsymbol{u}^t)$, the peer evaluation signal generated from agent k is defined as:

$$z_k^t(o_k^t, o_k^{t+1}, u_k^t, r_k^t) := r_k^t + \gamma Q_k^{\boldsymbol{\pi}^t}(o_k^{t+1}, \pi_k^t(o_k^{t+1})) - Q_k^{\boldsymbol{\pi}^t}(o_k^t, u_k^t). \tag{3}$$

The peer evaluation signal in (3) can be interpreted as a quantification of the effect of the joint actions on agent k's expected reward,



(a) Example: Evaluation feedback exchange



(b) The architecture of PED-DQN

Figure 1: Evaluation Feedback and Architecture of PED-DQN. (a) Agent 2 broadcasts its evaluation of the transition z_2 and then calculates the aggregate peer evaluation Z_2 using the received feedback z_1 and z_3 from its peers, agents 1 and 3. (b) A-DQN is used for ϵ-greedy action selection, while M-DQN is used for the calcualtion of peer evaluation.

given a current joint policy $\boldsymbol{\pi}^t$. Positive z_k^t implies that the current joint action $\boldsymbol{u}^t$ results in a more favorable reward on average, whereas the opposite applies for negative values. Note that this has similar formulation as the *temporal difference* but it is used differently since these signals are intended to affect the peers instead of the agent itself. The peer evaluation signal z_k^t is broadcast to its "peers" K_k, which is a set of agents who impact agent k's observation. If K_k is empty, then no reward reshaping happens. We assume that K_k is given *a priori*; otherwise, $K_k = \mathcal{A} \setminus \{k\}$. We also use action-value function Q instead of value function V to reduce the noise from the agent's own action.

Reshaping reward from peer evaluation. Let a be a "peer" of the agent k, i.e., $k \in K_a$ and $a \in K_k$, who would receive the peer evaluation signal z_k^t from each $k \in K_a$. From the perspective of agent a, we average the received peer evaluation signals of its observation o_a^t and action u_a^t from its peer set K_a to the *reward reshaping term* $Z_a^t\left[o_a^t, u_a^t\right]$ as follows.[4]

$$Z_a^t\left[o_a^t, u_a^t\right] = \frac{1}{|K_a|} \sum_{k \in K_a} z_k^t. \tag{4}$$

Figure 1a shows an example of the reward reshaping term and the peer evaluation signal. The reshaping term $Z_a^t[o_a^t, u_a^t]$ is sampled several times to evaluate the averaged effect of (o_a^t, u_a^t) to its peers.

$$\hat{Z}_a^t\left[o_a^t, u_a^t\right] \approx \frac{1}{|K_a|} \sum_{k \in K_a} \times \\ \mathbb{E}_{(\boldsymbol{o}, \boldsymbol{o}', \boldsymbol{u}, \boldsymbol{r}) \sim \boldsymbol{\pi}^t : o_a = o_a^t, u_a = u_a^t}\left[z_k(o_k, o_k', u_k, r_k)\right] \tag{5}$$

In order to estimate the expectation, we maintain the moving average of $Z_a^t[o_a^t, u_a^t]$, as follows.

$$\hat{Z}_a^{t+1}[o_a^t, u_a^t] \leftarrow (1 - \alpha)\hat{Z}_a^t[o_a^t, u_a^t] + \alpha Z_a^t\left[o_a^t, u_a^t\right], \tag{6}$$

where α is the learning rate hyperparameter. Then, the reshaped reward $\hat{r}_a^t[o_a^t, u_a^t]$ is derived as the summation of the base reward r_a^t and the reshaping term $\hat{Z}_a^t[o_a^t, u_a^t]$:

$$\hat{r}_a^t[o_a^t, u_a^t] = r_a^t + \beta \hat{Z}_a^t[o_a^t, u_a^t], \tag{7}$$

where β is the weight hyperparameter quantifying how willing the agents are to reshape their own reward. Depending on the individual reward functions, setting $\beta > 0$ will sometimes prevent agents to achieve their individual optimal rewards. Since the goal is to maximize the social welfare, this sacrifice of individual optimum is a natural compromise. The case when $\beta = 0$ corresponds to totally selfish agents, which becomes equivalent to independent learning (*e.g.*, IQL [26]). Given this formulation, we can say that the reshaped reward vector $\hat{\boldsymbol{r}}^t$ is a mapping from the average evaluation vector $\hat{\boldsymbol{Z}}^t$. That said, the reward update $\hat{\boldsymbol{r}}^{t+1} = H(\hat{\boldsymbol{r}}^t, \boldsymbol{\pi}^t)$ can be understood as the evaluation update $\hat{\boldsymbol{Z}}^{t+1} = H'(\hat{\boldsymbol{Z}}^t, \boldsymbol{\pi}^t)$ for some other mapping H', where the relation between the reshaped rewards and the evaluation signals is given by (7).

We note that this reward reshaping plays a pivotal role in aiding the agents on how to "behave as is expected of them". With training, each agent is essentially incentivized to act in ways other agents think it would, with the collective welfare in mind. This makes the learning process easier and intuitively corresponds to a gradual removal of non-stationarity. This is because even though the updates may be concurrent, agents can somewhat tell how the other agents would update. CESs implicitly evaluate changes in other agents' policies. Typically, policies change due to exploration; however, in our mechanism, these exploration steps will automatically have consequences in the form of CESs, which will discourage agents from transitions that have been given highly negative CESs.

One interesting property of the reward reshaping scheme is that the resulting reshaped rewards is also dependent on the definition of peers. In particular, there is a trade-off between the locality of peers and the agent-interpretability of the reshaping term. If we consider all the other agents as the peers of a (i.e. $K_a = -a$), then the reshaping term is highly diluted and becomes less informative in terms of crediting the change of rewards to a because any of the other agents could have caused the Z_a to be non-zero. On the other hand, if the set of peers is too "localized", peer evaluations that should affect all the agents will propagate too slowly or in extreme cases, not propagate to the other agents at all. A reasonable definition of the peer set would be the set of observable agents as it is easier for the agents to interpret rewards that they can associate

[4]The square bracket in (4) means we update the entry $[o_a^t, u_a^t]$ in Z_a^t.

Algorithm 1 Peer Evaluation based Dual DQN Training Algorithm

```
1:  Initialize replay memory $D$
2:  Initialize $[Q_a^A]$, with random parameters $\boldsymbol{\theta}^A = \{\theta_a^A\}_{a=1}^N$
3:  Initialize $[Q_a^M]$, with random parameters $\boldsymbol{\theta}^M = \{\theta_a^M\}_{a=1}^N$
4:  Initialize target parameters $\boldsymbol{\theta}^{M'} = \boldsymbol{\theta}^M$ , $\boldsymbol{\theta}^{A'} = \boldsymbol{\theta}^A$
5:  for episode = 1 to max_episodes do
6:      Observe initial observation $\boldsymbol{o}^0 = \{o_1^0, o_2^0, ..., o_N^0\}$
7:      for $t = 1$ to max_steps do
8:          With probability $\epsilon$, select a random action $u_a^t$
9:          $u_a^t = \arg\max_{u_a^t} Q_a^A(o_a^t, u_a^t|\theta_a^A)$ for each agent $a$
10:         Take action $\boldsymbol{u}^t = \{u_a^t\}_{a=1}^N$
11:         Retrieve next observation $\boldsymbol{o}^{t+1}$ and reward $\boldsymbol{r}^t = \{r_a^t\}_{a=1}^N$
12:         Calculate the peer evaluation $z_a^t$ for each agent $a$
                $z_a^t = r_a^t + \gamma \max_u Q_a^M(o_a^t, u|\theta_a^{M'}) - Q_a^M(o_a^t, u_a^t|\theta_a^{M'})$
13:         Identify peers $K_a$ and aggregate peer evaluations $Z_a$ for
            each agent $a$ using Eq. (4).
14:         Store transition $(\boldsymbol{o}^t, \boldsymbol{u}^t, \boldsymbol{r}^t, \boldsymbol{o}^{t+1}, Z^t)$ in $D$
15:         Sample a random minibatch of transitions $(\boldsymbol{o}, \boldsymbol{u}, \boldsymbol{r}, \boldsymbol{o}', Z)$
            from $D$
16:         if $ET + t >$ warm_up_period then
17:             $\hat{r}_a = r_a + \beta Z_a$ for each sample of agent $a$
18:         else
19:             $\hat{r_a} = r_a$ for each sample of agent $a$
20:         end if
21:         Let $y_a^A(o_a', r_a|\theta_a^{A'}) = r + \gamma \max Q^A(o_a', u_a'|\theta_a^{A'})$
22:         Let $y_a^M(o_a', r_a|\theta_a^{M'}) = \hat{r} + \gamma \max Q^M(o_a', u_a'|\theta_a^{M'})$
23:         For each agent $a$ update $\theta^A$ by minimizing the loss
                $L_a^A(\boldsymbol{o}_a, \boldsymbol{u}_a, \boldsymbol{r}_a, \boldsymbol{o}_a'|\theta_a^A, \theta_a^{A'})$
                $= \sum_{\substack{(o,u,r,o')\in \\ (\boldsymbol{o}_a, \boldsymbol{u}_a, \boldsymbol{r}_a, \boldsymbol{o}_a')}} (Q^A(o, u|\theta_a^A) - y_a^A(o', r|\theta_a^{A'}))^2$
24:         For each agent $a$ update $\theta^M$ by minimizing the loss
                $L_a^M(\boldsymbol{o}_a, \boldsymbol{u}_a, \hat{\boldsymbol{r}}_a, \boldsymbol{o}_a'|\theta_a^M, \theta_a^{M'})$
                $= \sum_{\substack{(o,u,r,o')\in \\ (\boldsymbol{o}_a, \boldsymbol{u}_a, \boldsymbol{r}_a, \boldsymbol{o}_a')}} (Q^M(o, u|\theta_a^M) - y_a^M(o', r|\theta_a^{M'}))^2$
            Update target network parameters $\boldsymbol{\theta}^M = \boldsymbol{\theta}^{M'}$, $\boldsymbol{\theta}^{A'} = \boldsymbol{\theta}^A$
            every period $I$
25:     end for
26: end for
```

to an observation. We leave the problem of selecting optimal peer sets for future studies.

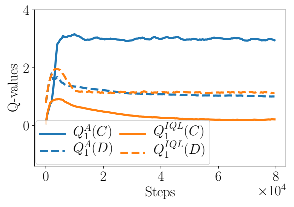
3.3 Peer-evaluation based Dual DQN (PED-DQN)

To instantiate our idea on the evaluation-based reward reshaping, we now describe the design of our training architecture called *Peer Evaluation based Dual DQN* (PED-DQN). PED-DQN consists of two DQNs running under different time-scales, where each learning agent is equipped with an *Action DQN* (A-DQN) and a *Mission DQN* (M-DQN), used for action selection and peer evaluation, respectively.

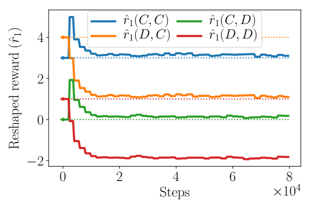
Figure 1b shows the flow of information during the training. A-DQN outputs the Q-values Q_a^A, which is trained with the reshaped reward $\hat{r}_a^t$. The agents use the output of A-DQN to select an action that has the maximum Q-value for a given observation. Updating the A-DQN corresponds to a mapping F in (1). M-DQN maintains an estimate of Q_a^M with respect to the current joint policy $\boldsymbol{\pi}^t$, which can be trained according to its own "mission" defined by the base reward r_a. The output of M-DQN is used to calculate the peer evaluation signal as in (3). As the joint policy of agents is updated, the peer evaluation signal also changes accordingly by updating the M-DQN, which corresponds to the mapping H in (2). As discussed with (1) and (2) in mind, we train the A-DQN and M-DQN in two different time-scales via employing different learning rates. As in the actor-critic method, where the critic usually has a faster learning rate, we use a faster learning rate for the M-DQN, which updates for quasi-stationary A-DQN.

We do note that, although agents can be trained simultaneously in a single computer, the mechanism is distributed by nature. Agents only transmit their peer evaluation to a subset of agents, which will allow for an easier distributed implementation. In addition, the method is flexible in terms of the number of agents. Increasing the number of agents would obviously affect the policies, but the algorithm and its implementation do not require any special consideration when introducing new agents.

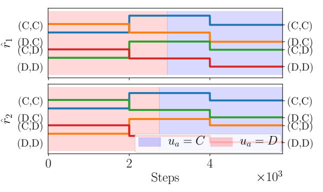
The training algorithm for PED-DQN is described in Algorithm 1. Our training algorithm is based on DQN algorithm. For practical stability, we use the target network parameters of the M-DQN to calculate the peer evaluation as shown in line 12. We also allow the agents to train the A-DQN using the base reward r_a for some period of time, called `warm_up_period`, before training with the reshaped reward as indicated in lines 16-20.



(a) IQL learned defection while ours learned cooperation



(b) Reshaped rewards $\hat{r}_1(D,C)$ and $\hat{r}_1(D,D)$ penalize defection



(c) Peer evaluations steered the agents' actions toward $u_a = C$

Figure 2: Prisoner's dilemma: Evolution of Q-values and reshaped rewards for agent 1.

3.4 Case Study: Prisoner's Dilemma

Table 1: Prisoner's dilemma

(a) original payoff

	C	D
C	3, 3	0, 4
D	4, 0	1, 1

(b) reshaped payoff

	C	D
C	3.11, 3.13	0.10, 1.13
D	1.11, 0.13	-1.90, -1.87

We illustrate how our evaluation-based reward reshaping works for a two-person Prisoner's Dilemma (PD) (see Table 1a). It is widely known that PD's Nash equilibrium (D,D) differs from the socially optimal joint action (C,C). At each step, both agents choose their actions and receive a reward according to their joint actions. In this case study, we simply use a single-state tabular Q-learning [28] with two actions for easier interpretability. For testing purposes, we keep track of the reshaped rewards by simulating all possible combinations of actions.

Figures 2a and 2b show the evolution of Q-values and reshaped rewards, respectively. Only Agent 1's values are shown since the game is symmetric. Independent Q-learning (IQL) can be seen as a best response strategy since it chooses the action that has the maximum Q-value. As expected, agents learn to choose defect (D), which is a sub-optimal point. When we use peer evaluation signals, the payoff tables are effectively reshaped, so that the each agent's best response leads to (C,C). At early stages of training, agents tend to prefer defection but eventually learn to cooperate as can be seen in Figure 2c, where the upper and lower shaded regions represent the actions of Agents 1 and 2, respectively. As seen in Table 1b, the game has been successfully reshaped and thus become well-coordinated.

4 EXPERIMENTS

To evaluate the performance of our algorithm, we consider two environments detailed in this section. For comparison, we use a feed-forward QMIX[5] to represent the assumption that giving a joint reward induces cooperation. We allow QMIX access to the global state, which can comprise all the agents' observations and actions, and the whole environment information (*e.g.*, coordinates in the PCP task). We use Independent DQN (IDQ) to be a baseline algorithm for agents with separate rewards. To illustrate the individualized importance of the different components of PED-DQN, we created ablated versions of our algorithm, which is detailed in the next subsection (Section 4.1).

4.1 Ablation Algorithms

Peer Evaluation based DQN. To illustrate the importance of A-DQN and M-DQN's separation, we created an ablated version of our algorithm, called PE-DQN. This ablated version of the PED-DQN is only composed of a single DQN, which is used for both action selection and peer evaluation. The agent follows the logic of training the Action-DQN. However, for peer evaluation, the following formula is used: $z_a = r_a + \gamma \max_u Q_a(o_a, u|\theta'_a) - Q_a(o'_a, u_a|\theta'_a)$.

Prosocial DQN. Prosocial DQN is a modified version of the PE-DQN. Instead of sending peer evaluations, agents use the immediate reward as the peer evaluation (*i.e.*, $z_a = r_a$). This can be thought of as the modified version of the prosocial agents in [19]. However, the difference is that only peers can share rewards, and that agents are using DQN instead of policy gradients. Given agent a's peers, K_a, the agent's perceived reward is now given by: $\hat{r}_a = r_a + \frac{\beta}{|K_a|}\sum_{k \in K_a} r_k$.

[5]Our experiments show better results and faster training when we removed recurrent layers in QMIX.

4.2 Environments

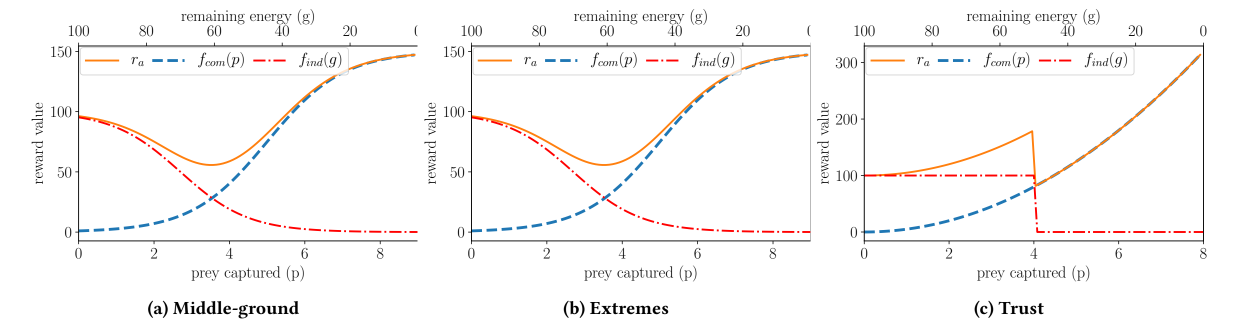
Resource sharing (RS). The scenario takes the structure of a graph, where each agent is a node. Agents are given g_a^1 gems at the start of the episode. Each agent's goal is to have g_a^* gems. At each time step t, agents choose whether to `open`, `close`, or `share` their gems. If the agent chooses `share`, it will share a gem to each of its one-hop neighbors who chose `open`. If there is not enough gems to share, the receivers are chosen randomly. Sharers and receivers will have to pay a *link usage cost*, which is worth -1 reward per gem. If the agent chooses `close`, it will not accept any gems from other agents. An agent's observation is the tuple $o_a^t = (g_a^*, g_a^t, u_a^{t-1}, u_{K_a}^{t-1})$, where $u_{K_a}^{t-1}$ is the previous action of agent a's one-hop neighbors.

In this scenario, at least two obvious reward functions can be formulated in addition to the link usage cost. We can simply use the negative individual absolute error (iAE)(*i.e.*, $-|g_a^* - g_a^t|$) as an individual reward or use the negative mean absolute error (MAE) as a common reward across the agents. We show that even though MAE is an intuitive reward to enforce cooperation, it might be easier to train using the iAE.

For all the algorithms, the individual DQNs are composed of two fully connected hidden layers with 32 units and ReLU as the activation function followed by 3 output units corresponding to the agent's possible actions. For QMIX, the global state is composed of the goal, target, and previous actions of all agents. The mixing network is composed of two hidden layers with 64 units.

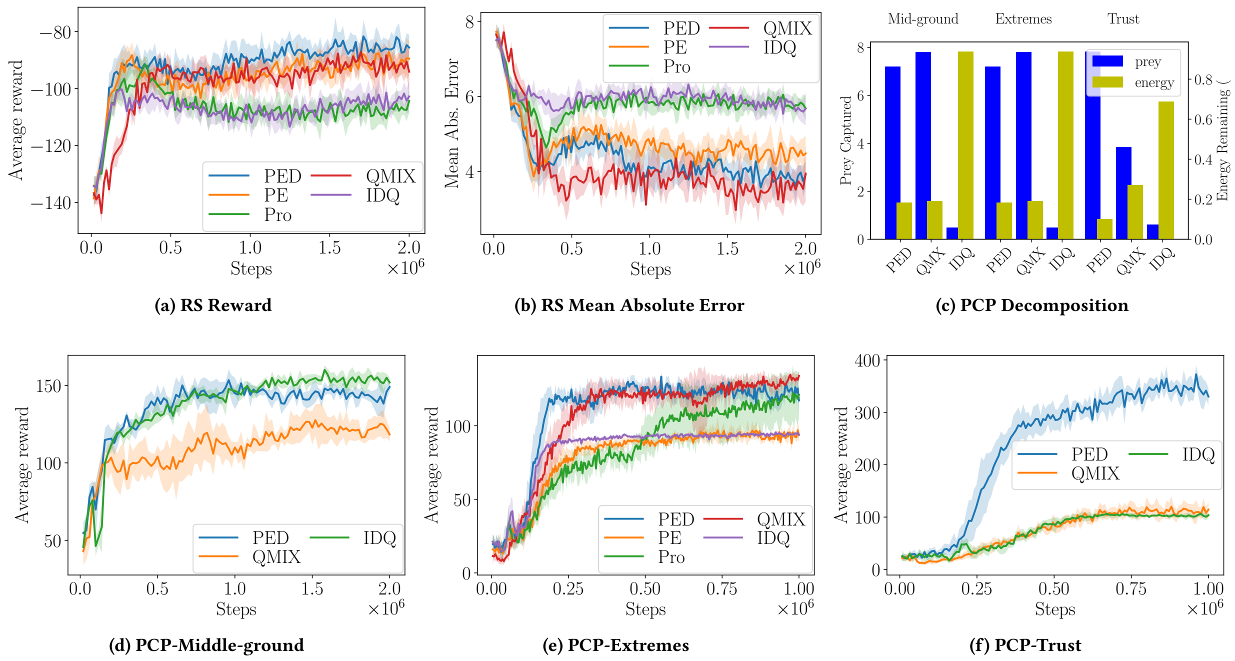
Partially cooperative pursuit (PCP). A popular performance evaluation environment in cooperative multi-agent systems is *pursuit* (a.k.a. *predator-prey*). The environment is a 10×10 grid world, where each cell is either empty, a wall, or occupied by a predator or a prey. We only trained the predators while the prey is a rule-based agent whose policy is to move to an empty adjacent cell. Each agent has five possible actions corresponding to `north`, `south`, `east`, `west`, and `stay`. The capture criterion is to have at least three predators surrounding the prey. Agents can only observe a 7×7 grid centered at their own coordinates. We modified the scenario by adding an individual factor to the rewards. We aim to show that, given this kind of structure, a naïve aggregation of rewards to induce cooperation (*e.g.*, mean or sum) may lead the agents to disregard their individual factors, while using the individual rewards may lead them to a suboptimal policy.

Each predator keeps track of their *energy*, which is initially set to 100 and is decreased by 1 for each action except `stay`. Each agent receives a reward at the end of the episode. An agent's reward is given by the sum of the common and individual rewards, $r_a = f_{\text{com}}(p) + f_{\text{ind}}(g_a)$, where p is the number of preys captured and g_a is the agent's energy remaining at the end of the episode. We modified the reward functions to show varying trade-offs, shown in Figure 3.



(a) Middle-ground (b) Extremes (c) Trust

Figure 3: Rewards formulations for PCP environments. Energy-to-prey mapping are approximations.



(a) RS Reward (b) RS Mean Absolute Error (c) PCP Decomposition

(d) PCP-Middle-ground (e) PCP-Extremes (f) PCP-Trust

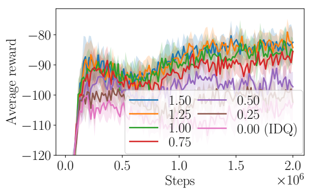
Figure 4: Performance evaluation on resource sharing and partially cooperative pursuit. Shaded areas denote 95% confidence intervals based on 5-6 different seeds. In (a), (b), and (e), 'PE' and 'Pro' correspond to the ablated versions of PED-DQN (see Section 4.1).

For all algorithms, the individual DQNs are composed for a convolutional layer with 32 filters with size 3×3. The output of the convolutional layer is flattened and concatenated with the extra information (*i.e.*, energy, prey captured, step). This is followed by two fully connected layers with 64 units, and an output layer with five actions. For QMIX, the global state is composed of the coordinates of all the agents (including the prey) concatenated with the extra information from all the predators. Although a recurrent network may improve the performance, we figured that using RNNs only adds complexity in training. Performance evaluations for each task show that agents can achieve the estimated optimal values, and thus the feed-forward network is deemed adequate for this scenario.

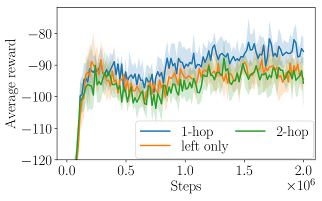
4.3 Results and Discussion

In this section, we discuss the results of our experiments. Due to space and visibility limitations only important ablation results are shown in graphs. Full results are included in the Supplementary.

Resource sharing. In our experiments, we used 8 agents in a circular graph. The total target gems and available gems are equal (i.e., $\sum g_a^* = \sum g_a^t, \forall t$) so that the maximum reward in one time step is zero. Figure 4a shows the total episodic base rewards of the agents averaged across multiple seeds. PED-DQN outperforms the other algorithms by a considerable margin. In this scenario, using the PE-DQN still has some cooperation-inducing effect, even though it did not achieve the same level as that of PED-DQN. Surprisingly, Pro-DQN's performance degraded after the first quarter of training. This



(a) Base rewards on varying β values



(b) Base rewards on different K's

Figure 5: Base rewards and mean absolute error across training for varying values of β and K.

is because sharing immediate rewards can amplify negative short-term rewards (link usage cost in this case). Peer evaluations consider possible future rewards, which dilute the negative effects of the immediate reward when sharing or receiving gems. QMIX achieved a similar performance as that of PE-DQN, though they learned different policies. As shown in Figure 4b, QMIX tries to minimize the mean absolute error, which is a large factor for the rewards. However, QMIX downplayed the link usage cost in contrast to PE-DQN. Once an IDQ agent achieves its goal, it does not have any incentive for letting gems pass through itself, which may hinder others from completing their own goals.

In addition to the algorithm comparisons, we also experimented with different β values (Figure 5a) and different definitions of peers K (Figure 5b). Smaller β values perform worse, while assigning $\beta > 1$ does not significantly improve the performance. Since the graph is circular, we added the two-hop neighbors and left-neighbor as variations of the K definitions. Our results show that the one-hop neighbors (default K) performs best because receiving feedback from only one agent has less representational power, while it does not make much sense to exchange evaluations beyond one's observation.

Partially cooperative pursuit. Figures 4d-4f show the rewards of each algorithm across time in different PCP configurations, and Figure 4c shows the outcomes of the policies learned with respect to the two factors of the agents' rewards. In all the configurations, PED-DQN consistently achieves a relatively good reward.

For the *middle-ground* scenario (Figure 3a), the best policy is to capture prey with some energy threshold. QMIX was fooled into learning to actively capture preys, which resulted in a suboptimal reward. I-DQN performs surprisingly well for this reward formulation. The reason is that agents can still passively capture the preys when they come close, even though the agents have learned to conserve energy.

In the *extremes* scenario (Figure 3b), agents are expected to either prioritize capturing preys or saving energy. PED-DQN, Pro-DQN, and QMIX learned to prioritize prey captures while IDQ and PE-DQN learned to act selfishly by trying to conserve their own energy. IDQ converged to this equilibrium because agents need a much higher level of coordination as opposed to just staying still, even though capturing preys produces higher rewards. As for PE-DQN, poor performance could also be attributed to incorrect peer evaluation. The peer evaluations given will be biased by the peer evaluation received, in which a poor evaluation in the earlier stages of the training could propagate until the end of the training. In contrast to the previous environment, Pro-DQN was able to find a good equilibrium, though slower than QMIX and PED-DQN. This is because the propagation of the pro-social reward is slower since the rewards are only given at the end of the episode. However, giving the rewards at the end of the episode actually encouraged the Pro-DQN agents to group together when the episode is about to end, allowing them to passively capture preys if possible.

An interesting outcome was observed in the *trust* scenario (Figure 3c). The best policy is to prioritize capturing preys, although a high level of cooperation is required due to the form of the reward function. PED-DQN agents learned to maximize the prey capture counts; I-DQN agents conserved their energy; and QMIX agents learned to balance the two factors. One of the possible reasons why QMIX failed to find the best policy is the fact that the reward function discourages the agents to explore the "risky" behavior. Although the agents have established that capturing preys leads to better rewards, they did not exploit this due to the risk of getting lower rewards in the attempt. However, PED-DQN agents were incentivized for capturing the prey, which mitigated the effect of the trough in the reward function.

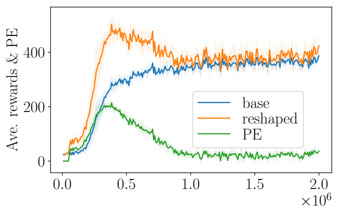


Figure 6: Base reward, reshaped rewards, and peer evaluation signals for PCP-Trust.

Convergence. Although we do not have theoretical convergence guarantees, Figure 6 presents an empirical evidence for convergence. We have that the PEs tending towards zero so that the reshaped rewards are only reshaped "enough" to maintain good social welfare. Reshaping beyond this extent may lead to policies that significantly diverge from the original intent of the reward designs (*i.e.*, prioritization). Similar observation can be found for other environments. It is also worth noting that similar to actor-critic methods, the difference in learning rates is not an absolute necessity.

5 CONCLUSION

In this paper, we presented Peer Evaluation based Dual DQN (PED-DQN) in order to induce cooperation among semi-cooperative agents. Through PED-DQN, agents exchange peer evaluation signals, which gradually change the game so that an agent's myopic best response eventually leads to a higher global reward. We performed experiments in environments where semi-cooperation can be beneficial and shown competitive results against other algorithms. Our results show that the appropriate use of peer evaluation reshapes the individual rewards, such that it encourages cooperation among the agents. Future work includes smarter aggregation of peer evaluation, as well as run-time peer evaluations, where agents could change policies during execution.

ACKNOWLEDGMENTS

This work has been supported by the Unmanned Swarm CPS Research Laboratory program of Defense Acquisition Program Administration and Agency for Defense Development.(UD190029ED)

REFERENCES

[1] Tom Eccles, Edward Hughes, János Kramár, Steven Wheelwright, and Joel Z. Leibo. 2019. Learning Reciprocity in Complex Sequential Social Dilemmas. *CoRR* abs/1903.08082 (2019). arXiv:1903.08082 http://arxiv.org/abs/1903.08082

[2] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Proceedings of the Advances in NeurIPS*. 2137–2145.

[3] Jakob Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. 2018. Learning with Opponent-Learning Awareness. In *Proceedings of AAMAS*.

[4] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Philip Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of ICML*.

[5] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of AAAI*.

[6] Jakob N. Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. 2019. Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning. In *Proceedings of ICML*.

[7] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *Proceedings of AAMAS*. 66–83.

[8] Serhii Havrylov and Ivan Titov. 2017. Emergence of language with multi-agent games: learning to communicate with sequences of symbols. In *Proceedings of the Advances in NeurIPS*. 2146–2156.

[9] Edward Hughes, Joel Z. Leibo, Matthew G. Philips, Karl Tuyls, Edgar A. Duéñez-Guzmán, Antonio García Castañeda, Iain Dunning, Tina Zhu, Kevin R. McKee, Raphael Koster, Heather Roff, and Thore Graepel. 2018. Inequity aversion resolves intertemporal social dilemmas. In *Proceedings of NIPS*. 3326–3336.

[10] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Çaglar Gülçehre, Pedro A. Ortega, DJ Strouse, Joel Z. Leibo, and Nando de Freitas. 2018. Intrinsic Social Motivation via Causal Influence in Multi-Agent RL. *CoRR* abs/1810.08647 (2018). arXiv:1810.08647 http://arxiv.org/abs/1810.08647

[11] Vijaymohan R. Konda and Vivek S. Borkar. 1999. Actor-Critic–Type Learning Algorithms for Markov Decision Processes. *SIAM J. Control Optim.* 38, 1 (Nov. 1999), 94–123.

[12] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of AAMAS*. 464–473.

[13] Alistair Letcher, Jakob Foerster, David Balduzzi, Tim Rocktäschel, and Shimon Whiteson. 2019. Stable Opponent Shaping in Differentiable Games. In *Proceedings of ICLR*.

[14] Michael L. Littman. 1994. Markov Games As a Framework for Multi-agent Reinforcement Learning. In *Proceedings of ICML*. 157–163.

[15] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Proceedings of NIPS*. 6379–6390.

[16] Igor Mordatch and Pieter Abbeel. 2018. Emergence of Grounded Compositional Language in Multi-Agent Populations. In *Proceedings of AAAI*.

[17] John Nash. 1950. The Bargaining Problem. *Econometrica* 18, 2 (1950), 155–162.

[18] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. 2018. Lenient Multi-Agent Deep Reinforcement Learning. In *Proceedings of AAMAS*.

[19] Alexander Peysakhovich and Adam Lerer. 2017. Prosocial learning agents solve generalized Stag Hunts better than selfish ones. *CoRR* abs/1709.02865 (2017). arXiv:1709.02865 http://arxiv.org/abs/1709.02865

[20] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of ICML*.

[21] L. Shapley. 1953. *A Value for n-Person Games*. In H. W. Kuhn and A. W. Tucker, editors, Contribution to the Theory of Games II, vol. 28 of Annals of Mathematics Studies, Princeton University Press. 307–317 pages.

[22] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. 2019. Individualized Controlled Continuous Communication Model for Multiagent Cooperative and Competitive Tasks. In *Proceedings of ICLR*.

[23] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of ICML*.

[24] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Proceedings of the Advances in NeurIPS*. 2244–2252.

[25] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of AAMAS*. 2085–2087.

[26] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE* 12, 4 (04 2017), 1–15.

[27] Jane X. Wang, Edward Hughes, Chrisantha Fernando, Wojciech M. Czarnecki, Edgar A. Duéñez-Guzmán, and Joel Z. Leibo. 2019. Evolving intrinsic motivations for altruistic behavior. In *Proceedings of AAMAS*.

[28] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards*. Ph.D. Dissertation. King's College, Cambridge.

[29] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. 2018. Multiagent Soft Q-Learning. *arXiv preprint arXiv:1804.09817* (2018).

[30] David Wolpert and Kagan Tumer. 2002. Optimal Payoff Functions for Members of Collectives. *Advances in Complex Systems* (2002), 355–369.