Agent-Temporal Attention for Reward Redistribution in Episodic Multi-Agent Reinforcement Learning

Baicen Xiao
University of Washington
Seattle, WA, USA
bcxiao@uw.edu

Bhaskar Ramasubramanian
Western Washington University
Bellingham, WA, USA
ramasub@wwu.edu

Radha Poovendran
University of Washington
Seattle, WA, USA
rp3@uw.edu

ABSTRACT

This paper considers multi-agent reinforcement learning (MARL) tasks where agents receive a shared global reward at the end of an episode. The delayed nature of this reward affects the ability of the agents to assess the quality of their actions at intermediate time-steps. This paper focuses on developing methods to learn a temporal redistribution of the episodic reward to obtain a dense reward signal. Solving such MARL problems requires addressing two challenges: identifying (1) relative importance of states along the length of an episode (along time), and (2) relative importance of individual agents' states at any single time-step (among agents). In this paper, we introduce **Agent-Temporal Attention for Reward Redistribution in Episodic Multi-Agent Reinforcement Learning (AREL)** to address these two challenges. *AREL* uses attention mechanisms to characterize the influence of actions on state transitions along trajectories (*temporal attention*), and how each agent is affected by other agents at each time-step (*agent attention*). The redistributed rewards predicted by *AREL* are dense, and can be integrated with any given MARL algorithm. We evaluate *AREL* on challenging tasks from the Particle World environment and the StarCraft Multi-Agent Challenge. *AREL* results in higher rewards in Particle World, and improved win rates in StarCraft compared to three state-of-the-art reward redistribution methods. Our code is available at **https://github.com/baicenxiao/AREL**.

KEYWORDS

Multi-agent reinforcement learning, credit assignment, episodic rewards, attention mechanism

ACM Reference Format:
Baicen Xiao, Bhaskar Ramasubramanian, and Radha Poovendran. 2022. Agent-Temporal Attention for Reward Redistribution in Episodic Multi-Agent Reinforcement Learning. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 14 pages.

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) involves multiple autonomous agents that learn to collaborate to complete tasks in a shared environment by maximizing a global reward [6]. Examples of systems where MARL has been used include autonomous vehicle coordination [37], and video games [38, 43].

One approach to enable better coordination is to use a single centralized controller that can access observations of all agents [21]. In this setting, algorithms designed for single-agent RL can be used for the multi-agent case. However, this may not be feasible when trained agents are deployed independently or when communication costs between agents and the controller is prohibitive. In such a situation, agents will need to be able to learn decentralized policies.

The centralized training with decentralized execution (CTDE) paradigm, introduced in [26, 34], enables agents to learn decentralized policies efficiently. Agents using CTDE can communicate with each other during training, but are required to make decisions independently at test-time. The absence of a centralized controller will require each agent to assess how its own actions can contribute to a shared global reward. This is called the *multi-agent credit assignment problem*, and has been the focus of recent work in MARL, such as COMA [12], QMIX [34] and QTRAN [40]. Solving the multi-agent credit assignment problem alone, however, is not adequate to efficiently learn agent policies when the (global) reward signal is delayed until the end of an episode.

In reinforcement learning, agents seek to solve a sequential decision problem guided by reward signals at intermediate time-steps. This is called the *temporal credit assignment problem* [42]. In many applications, rewards may be delayed. For example, in molecular design [33], *Go* [39], and computer games such as *Skiing* [4], a summarized score is revealed only at the end of an episode. The *episodic* reward implies absence of feedback on quality of actions at intermediate time steps, making it difficult to learn good policies. The long-term temporal credit assignment problem has been studied in single-agent RL by performing return decomposition via contribution analysis [2] and using sequence modeling [24]. These methods do not directly scale well to MARL since size of the joint observation space grows exponentially with number of agents [26].

Besides scalability, addressing temporal credit assignment in MARL with episodic rewards presents two challenges. It is critical to identify the relative importance of: i) each agent's state at any single time-step (*agent dimension*); ii) states along the length of an episode (*temporal dimension*). We introduce **Agent-Temporal Attention for Reward Redistribution in Episodic Multi-Agent Reinforcement Learning (AREL)** to address these challenges.

AREL uses attention mechanisms [44] to carry out *multi-agent temporal credit assignment* by concatenating: i) a *temporal attention module* to characterize the influence of actions on state transitions along trajectories, and; ii) an *agent attention module*, to determine how any single agent is affected by other agents at each time-step. The attention modules enable learning a redistribution of the episodic reward along the length of the episode, resulting in a *dense* reward signal. To overcome the challenge of scalability, instead of working with the concatenation of (joint) agents' observations,

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

AREL analyzes observations of each agent using a temporal attention module that is shared among agents. The outcome of the temporal attention module is passed to an agent attention module that characterizes the relative contribution of each agent to the shared global reward. The output of the agent attention module is then used to learn the redistributed rewards.

When rewards are delayed or episodic, it is important to identify 'critical' states that contribute to the reward. The authors of [14] recently demonstrated that rewards delayed by a long time-interval make it difficult for temporal-difference (TD) learning methods to carry out temporal credit assignment effectively. AREL overcomes this shortcoming by using attention mechanisms to effectively learn a redistribution of an episodic reward. This is accomplished by identifying critical states through capturing long-term dependencies between states and the episodic reward.

Agents that have identical action and observation spaces are said to be *homogeneous.* Consider a task where two homogeneous agents need to collaborate to open a door by locating two buttons and pressing them simultaneously. In this example, while locations of the two buttons (states) are important, the identities of the agent at each button are not. This property is termed *permutation invariance*, and can be utilized to make the credit assignment process sample efficient [14, 24]. Thus, a redistributed reward must identify whether an agent is in a 'good' state, and should also be invariant to the identity of the agent in that state. *AREL* enforces this property by designing the credit assignment network with permutation-invariant operations among homogeneous agents, and can be integrated with MARL algorithms to learn agent policies.

We evaluate *AREL* on three tasks from the Particle World environment [26], and three combat scenarios in the StarCraft Multi-Agent Challenge [38]. In each case, agents receive a summarized reward only at the end of an episode. We compare *AREL* with three state-of-the-art reward redistribution techniques, and observe that *AREL* results in accelerated learning of policies and higher rewards in Particle World, and improved win rates in StarCraft.

2 RELATED WORK

Several techniques have been proposed to address temporal credit assignment when prior knowledge of the problem domain is available. Potential-based reward shaping is one such method that provided theoretical guarantees in single [31] and multi-agent [8, 27] RL, and was shown to accelerate learning of policies in [9]. Credit assignment was also studied by incorporating human feedback through imitation learning [22, 35] and demonstrations [5, 18].

When prior knowledge of the problem domain is not available, recent work has studied temporal credit assignment in single-agent RL with delayed rewards. An approach named RUDDER [2] used contribution analysis to decompose episodic rewards by computing the difference between predicted returns at successive time-steps. Parallelly, the authors of [24] proposed using natural language processing models for carrying out temporal credit assignment for episodic rewards. The scalability of the above methods to MARL, though, can be a challenge due to the exponential growth in the size of the joint observation space [26].

In the multi-agent setting, recent work has studied performing multi-agent credit assignment at each time-step. Difference rewards were used to assess the contribution of an agent to a global reward in [1, 10, 12] by computing a counterfactual term that marginalized out actions of that agent while keeping actions of other agents fixed. Value decomposition networks, proposed in [41], decomposed a centralized value into a sum of agent values to assess each one's contributions. A monotonicity assumption on value functions was imposed in QMIX [34] to assign credit to individual agents. A generalized approach to decompose a joint value into individual agent values was presented in QTRAN [40]. The Shapley Q-value was used in [46] to distribute a global reward to identify each agent's contribution. The authors of [47] decomposed global Q-values along trajectory paths, while [49] used an entropy-regularized method to encourage exploration to aid multi-agent credit assignment. The above techniques did not address long-term temporal credit assignment and hence will not be adequate for learning policies efficiently when rewards are delayed.

Attention mechanisms have been used for multi-agent credit assignment in recent work. The authors of [29] used an attention mechanism with a CTDE-based algorithm to enable each agent effectively model policies of other agents (from its own perspective). Hierarchical graph attention networks proposed in [36] modeled hierarchical relationships among agents and used two attention networks to effectively represent individual and group level interactions. The authors of [20, 25] combined attention networks with graph-based representations to indicate the presence and importance of interactions between any two agents. The above approaches used attention mechanisms primarily to identify relationships between agents at a specific time-step. They did not consider long-term temporal dependencies, and therefore may not be sufficient to learn policies effectively when rewards are delayed.

A method for temporal redistribution of episodic rewards in single and multi-agent RL was recently presented in [14]. A 'surrogate objective' was used to uniformly redistribute an episodic reward along a trajectory. However, this work did not use information from sample trajectories to characterize the relative contributions of agents at intermediate time-steps along an episode.

Our approach differs from the above-mentioned related work in that it uses attention mechanisms for multi-agent temporal credit assignment. *AREL* overcomes the challenge of scalability by analyzing observations of each agent using temporal and agent attention modules, which respectively characterize the effect of actions on state transitions along a trajectory and how each agent is influenced by other agents at each time-step. Together, these modules will enable an effective redistribution of an episodic reward. *AREL* does not require human intervention to guide agent behaviors, and can be integrated with MARL algorithms to learn decentralized agent policies in environments with episodic rewards.

3 BACKGROUND

A fully cooperative multi-agent task can be specified as a decentralized partially observable Markov decision process (Dec-POMDP) [32]. A Dec-POMDP is a tuple $G = (S, A, P, r, Z, O, n, \gamma)$, where $s \in S$ describes the environment state. Each agent $i \in \{1, 2, \dots, n\}$ receives an observation $o^i \in O^i$ according to an observation function $Z(s, i) : S \times \mathbb{N} \to O$. At each time step, agent i chooses action $a^i \in A^i$ according to its policy $\pi^i : O^i \times A^i \to [0, 1]$. $A^1 \times \cdots \times A^n := A$

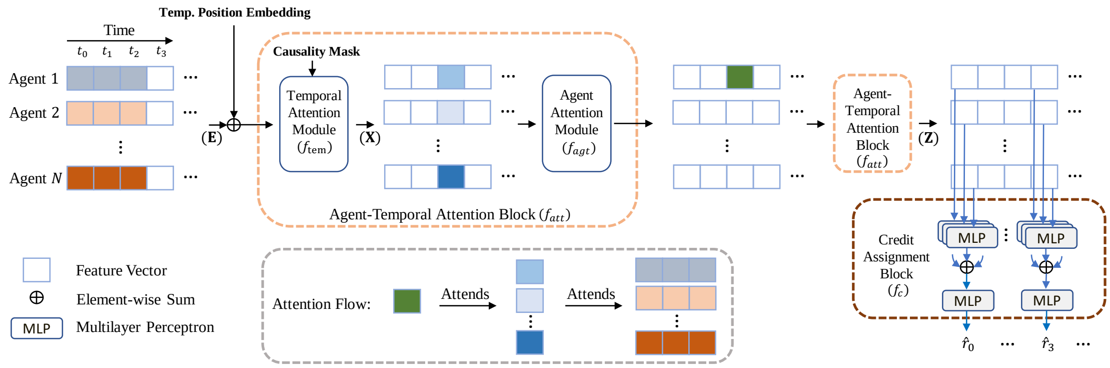


Figure 1: Schematic of AREL. The agent-temporal attention block concatenates temporal and agent attention modules, and summarizes input feature (e.g. observation) vectors. This is accomplished by establishing relationships between (*attending to*) information along time and among agents. The attention flow indicates that an output feature vector of the agent-temporal attention block for an agent at a time t (green square) can attend to input features from all other agents before and including time t. Multiple agent-temporal attention blocks can be concatenated to each other to improve expressivity. The output of the last such block is fed to the credit assignment block, which applies shared multi-layer perceptrons to each attention feature. The output is the redistributed reward, which is integrated with MARL algorithms (e.g. MADDPG, QMIX) to learn agent policies.

forms the joint action space, and the environment transitions to the next state according to the function $P : S \times A^1 \times \cdots \times A^n \longrightarrow S$. All agents share a global reward $r : S \times A \longrightarrow \mathbb{R}$. The goal of the agents is to determine their individual policies to maximize the *return*, $J := \mathbb{E}_{s\sim P, a^1\sim\pi^1,\ldots,a^n\sim\pi^n}[\sum_{t=0}^{T}\gamma^t r_t(s, a^1, \ldots, a^n)]$, where γ is a discount factor, and T is the length of the horizon. Let $a_t := (a_t^1, \ldots, a_t^n)$ and $R_t := \sum_{l=0}^{T-t}\gamma^l r_{t+1}$. A trajectory of length T is an alternating sequence of observations and actions, $\tau := (o_0, a_0, o_1, a_1, \ldots, o_T)$.

In a typical MARL task, agents receive reward $r(s, a)$ immediately following execution of action a at state s. The expected return can then be determined by accumulating rewards at each time step. In episodic RL, a reward is revealed only at the end of an episode at time T, and agents do not receive a reward at intermediate time-steps. As a consequence, the expected return for all $t < T$ will be the same (when $\gamma = 1$). Therefore, the quality of information available for learning policies will be poor at all intermediate time steps. Moreover, delayed rewards have been shown to introduce a large bias [2] or variance [31] in the performance of RL algorithms.

The CTDE paradigm [12, 26] can be adopted to learn decentralized policies effectively when dimensions of state and action spaces are large. During training, an agent can make use of information about other agents' states and actions to aid its own learning. At test-time, decentralized policies are executed. This paradigm has been used to successfully complete tasks in complex MARL environments [17, 19, 34].

4 APPROACH

This paper considers MARL tasks where agents share the same global reward, which is received only at the end of an episode. The objective is to redistribute this episodic reward for effective multi-agent temporal credit assignment. To accomplish this goal, it is critical to identify the relative importance of: i) individual agents' observations at each time-step, and; ii) observations along the length of a trajectory. We introduce *AREL* to address the above challenges. *AREL* uses an agent-temporal attention block to infer relationships among states at different times, and among agents. A schematic is shown in Fig. 1, and we describe its key components and overall workflow in the remainder of this section.

4.1 Agent-Temporal Attention

In order to redistribute an episodic reward in a meaningful way, we need to be able to extract useful information from trajectories. Each trajectory contains a sequence of observations involving all agents. At each time-step of an episode of length T, a feature of dimension D corresponds to the embedding of a single observation. When there are N agents, a trajectory is denoted by $\mathbf{E} \in \mathbb{R}^{T\times N\times D}$. The objective is to learn a mapping $f_{arel}(\mathbf{E}) : \mathbb{R}^{T\times N\times D} \longrightarrow \mathbb{R}^T$ to assign credit to the agents at each time-step. The information in a trajectory $\mathbf{E}$ comprises two parts: (1) *temporal information* between (embeddings of) observations at different time steps: this provides insight into the influence of actions on transitions between states; (2) *structural information*: this provides insight into how any single agent is affected by other agents.

These two parts are coupled, and hence studied together. The process of learning these relationships is termed *attention*. We propose an *agent-temporal attention structure*, inspired by the Transformer [44]. This structure selectively pays attention to different types of information- either from individual agents, or at different time-steps along a trajectory. This is accomplished by associating a weight to an observation based on its relative importance to other observations along the trajectory. The agent-temporal attention structure is formed by concatenating one agent attention module with one temporal attention module. The temporal attention modules determine how entries of $\mathbf{E}$ at different time-steps are related (along the 'first' dimension of $\mathbf{E}$). The agent attention module determines how agents influence one another (along the 'second' dimension of $\mathbf{E}$).

4.1.1 *Temporal-Attention Module.*

The input is a trajectory $\mathbf{E} \in \mathbb{R}^{T\times N\times D}$. To calculate the temporal attention feature, we obtain the transpose of $\mathbf{E}$ as $\bar{\mathbf{E}} \in \mathbb{R}^{N\times T\times D}$. Adopting notation from [44], each row $\mathbf{e}_i \in \mathbb{R}^{T\times D}$ of $\bar{\mathbf{E}}$ is transformed to a *query* $Q_i^{tem} := \mathbf{e}_i W_q^{tem}$, *key* $K_i^{tem} := \mathbf{e}_i W_k^{tem}$, and *value* $V_i^{tem} := \mathbf{e}_i W_v^{tem}$. $W_q^{tem}, W_k^{tem}, W_v^{tem} \in \mathbb{R}^{D\times D}$ are learnable parameters, and $i \in \{0, \dots, N-1\}$. The t^{th} row $x_{i,t}$ of the temporal attention feature $\mathbf{x}_i \in \mathbb{R}^{T\times D}$ is a weighted sum $x_{i,t} := \alpha_{i,t}^T V_i^{tem}$. The attention weight vector $\alpha_{i,t} \in \mathbb{R}^{T\times 1}$ is a normalization (*softmax*) of the inner-product between the t^{th} row of Q_i^{tem}, $q_{i,t}^{tem}$, and the key matrix K_i^{tem}:

$$\alpha_{i,t}^T = \text{softmax}\Big(\frac{q_{i,t}^{tem} K_i^{tem\,T}}{\sqrt{D}} \odot m_t^T\Big), \quad (1)$$

where $\odot$ is an element-wise product, and m_t is a *mask* with its first t entries equal to 1, and remaining entries 0. The mask preserves causality by ensuring that at any time t, information beyond t will not be used to assign credit. A temporal positional embedding [7] maintains information about relative positions of states in an episode. Position embeddings are learnable vectors associated to each temporal position of a trajectory. The sum of position and trajectory embeddings forms the input to the temporal attention module. The output of this module is $\mathbf{X} \in \mathbb{R}^{N\times T\times D}$, got by stacking $\mathbf{x}_i, i \in 0, \dots, N-1$. The temporal attention process can be described by a function $f_{tem}(\mathbf{E}) \to \mathbf{X}$.

The output of the temporal attention module results in an assessment of each agent's observation at any single time-step relative to observations at other time-steps of an episode. To obtain further insight into how an agent's observation is related to other agents' observations, an agent-attention module is concatenated to the temporal-attention module.

4.1.2 ***Agent-Attention Module.***

The agent-attention module uses the transpose of $\mathbf{X}$, denoted $\bar{\mathbf{X}} \in \mathbb{R}^{T\times N\times D}$. Each row of $\bar{\mathbf{X}}$, $\mathbf{x}_t \in \mathbb{R}^{N\times D}$ is transformed to a *query* $Q_t^{agt} = \mathbf{x}_t W_q^{agt}$, *key* $K_t^{agt} = \mathbf{x}_t W_k^{agt}$, and *value* $V_t^{agt} = \mathbf{x}_t W_v^{agt}$. Here, $W_q^{agt}, W_k^{agt}, W_v^{agt} \in \mathbb{R}^{D\times D}$ are learnable parameters. The i^{th} row $z_{t,i}$ of the agent attention feature $\mathbf{z}_t \in \mathbb{R}^{N\times D}$ is a weighted sum, $z_{t,i} = \beta_{t,i}^T V_t^{agt}$. Maintaining causality is not necessary when computing the agent attention weight vector $\beta_{t,i} \in \mathbb{R}^{N\times 1}$. These weights are determined similar to the temporal attention weight vector in Eqn. (1), except without a masking operation. Therefore,

$$\beta_{t,i}^T = \text{softmax}\Big(\frac{q_{t,i}^{agt} K_t^{agt\,T}}{\sqrt{D}}\Big). \quad (2)$$

The agent attention procedure can be described by a function $f_{agt}(\mathbf{X}) \to \mathbf{Z}$, where $\mathbf{Z} \in \mathbb{R}^{T\times N\times D}$.

4.1.3 ***Concatenating Attention Modules.***

The output of the temporal attention module is an entity $\mathbf{X}$ that attends to information at time-steps along the length of an episode for each agent. Passing $\mathbf{X}$ through the agent attention module results in an output $\mathbf{Z}$ that is attended to by embeddings at all time-steps and from all agents. The data-flow of this process can be written as a composition of functions: $f_{att} := f_{agt} \circ f_{tem}$. The temporal and agent attention modules can be repeatedly composed to improve expressivity. The position embedding is required only at the first temporal attention module when more than one is used.

4.2 Credit Assignment

The output of the attention modules is used to assign credit at each time-step along the length of the episode. Let $f_{arel} := f_c \circ (f_{att} \circ \cdots \circ f_{att})$, where $f_c : \mathbb{R}^{T\times N\times D} \to \mathbb{R}^T$. In order to carry out temporal credit assignment effectively, we leverage a property of permutation invariance.

4.2.1 ***Permutation Invariance.***

Agents sharing the same action and observation spaces are termed *homogeneous.* When homogeneous agents $ag1$ and $ag2$ cooperate to achieve a goal, the reward when $ag1$ observes $ob1$ and $ag2$ observes $ob2$ should be the same as the case when $ag1$ observes $ob2$ and $ag2$ observes $ob1$. This property is called *permutation invariance*, and has been shown to improve the sample-efficiency of multi-agent credit assignment as the number of agents increase [14, 23]. When this property is satisfied, the output of the function f_{arel} should be invariant to the order of the agents' observations. Formally, if the set of all permutations along the agent dimension (second dimension of $\mathbf{E}$) is denoted $\mathcal{H}$, then $f_{arel}(h_1(\mathbf{E})) = f_{arel}(h_2(\mathbf{E}))$ must be true for all $h_1, h_2 \in \mathcal{H}$.

The function f_{att} is permutation invariant along the agent dimension by design. A sufficient condition for f_{arel} to be permutation invariant is that the function f_c be permutation invariant. To ensure this, we apply a multi-layer perceptron (MLP), add the MLP outputs element-wise, and pass it through another MLP. When functions g_1 and g_2 associated to the MLPs are continuous and shared among agents, the evaluation at time t is the *predicted reward* $\hat{r}_t := g_2\big(\sum_{i=0}^{N-1} g_1(z_{t,i})\big)$. It was shown in [48] that any permutation invariant function can be represented by the above equation.

Remark 4.1. AREL *can be adapted to the* heterogeneous *case when cooperative agents are divided into homogeneous groups. Similar to a position embedding, we can apply an* agent-group embedding *such that agents within a group share an agent-group embedding. This will maintain permutation invariance of observations within a group, while enabling identification of agents from different groups.* AREL *will also work in the case when the multi-agent system is fully heterogeneous. This is equivalent to a scenario when there is only one agent in each homogeneous group. Therefore,* AREL *can handle agent types ranging from fully homogeneous to fully heterogeneous.*

4.2.2 ***Credit Assignment Learning.***

Given a reward R_T at the end of an episode of length T, the goal is to learn a temporal decomposition of R_T to assess contributions of agents at each time-step along the trajectory. Specifically, we want to learn $\{\hat{r}_t\}_{t=0}^T$ satisfying $\sum_{t=0}^T \hat{r}_t = R_T$. Since $f_{arel}^\theta(\mathbf{E})$ is a vector in $\mathbb{R}^T$, its t^{th} entry is denoted $f_{arel}^\theta(\mathbf{E}_t)$ $(= \hat{r}_t)$. The sequence $\{\hat{r}_t\}_{t=0}^T$ is learned by minimizing a *regression loss*, $l_r(\theta) := \mathbb{E}_{\mathbf{E},\mathbf{R_T}}\big[\frac{1}{T}\big(\sum_t (f_{arel}^\theta(\mathbf{E}_t)) - R_T\big)^2\big]$, where θ are neural network parameters.

The redistributed rewards will be provided as an input to a MARL algorithm. We want to discourage $\{\hat{r}_t\}_{t=0}^T$ from being sparse, since sparse rewards may impede learning policies [8]. We observe that more than one combination of $\{\hat{r}_t\}_{t=0}^T$ can minimize $l_r(\theta)$. We add a regularization loss $l_{reg}(\theta)$ to select among solutions that minimize $l_r(\theta)$. Specifically, we aim to choose a solution that also minimizes the variance $l_v(\theta)$ of the redistributed rewards, and set $l_{reg}(\theta) = l_v(\theta)$ (we examine other choices of $l_{reg}(\theta)$ in the *Appendix*). Using $l_v(\theta)$ as a regularization term in the loss function leads to learning less sparsely redistributed rewards. With $\omega \in \mathbb{R}_{\geq 0}$ denoting a hyperparameter, the combined loss function used to learn f^{θ}_{arel} is:

$$loss_{total}(\theta) = l_r(\theta) + \omega l_v(\theta), \tag{3}$$

where $l_v(\theta) := \mathbb{E}_{\mathbf{E}}\big[\frac{1}{T}\sum_t (f^{\theta}_{arel}(\mathbf{E}_t) - \bar{f}^{\theta}(\mathbf{E}))^2\big]$, and $\bar{f}^{\theta}(\mathbf{E}) := \big(\sum_t f^{\theta}_{arel}(\mathbf{E}_t)\big)/T$. The form of $loss_{total}(\theta)$ in Eqn. (3) incorporates the possibility that not all intermediate states will contribute equally to R_T, and additionally results in learning less sparsely redistributed rewards. Note that $\arg\min_{\theta}[loss_{total}(\theta)]$ will not typically yield $l_r(\theta) = l_v(\theta) = 0$ (which corresponds to a uniform redistribution of rewards). Since some states may be common to different episodes, the redistributed reward $\hat{r}_t$ at each time-step cannot be arbitrarily chosen. For e.g., consider N different episodes $\{E_i\}_{i=1}^{N}$, each of length L, with distinct cumulative episodic rewards R_i. If an intermediate state s is common to episodes E_j and E_k, under a uniform redistribution, distinct rewards R_j/L and R_k/L will be assigned to s, which is not possible. Thus, $l_r(\theta) = 0$ and $l_v(\theta) = 0$ will not both be true, implying that a uniform redistribution may not be viable.

4.3 Algorithm

Algorithm 1 *AREL*

Input: Number of agents N. Reward weight $\alpha \in [0, 1]$.
Initialize parameters θ, ϕ for credit assignment and RL (policy/critic) modules respectively.
Experience buffer for storing trajectories $B_e \leftarrow \emptyset$. Prediction function update frequency M.

```
1:  for Episode k = 0, ... do
2:      Reset episode return R_T ← 0; Reset trajectory for current episode τ ← ∅
3:      for step t = 0, ..., T − 1 do
4:          Sample action a_t^i ∼ π_φ^i(o_t^i), for i = 0, ..., N − 1
5:          Take action a_t; Observe o_{t+1} = (o_{t+1}^0, ..., o_{t+1}^{N−1})
6:          Store transition τ ← τ ∪ {(o_t, a_t, o_{t+1})}
7:      end for
8:      Update episode reward R_T; Store trajectory B_e ← B_e ∪ (τ, R_T)
9:      Sample a batch of trajectories B_τ ∼ B_e
10:     Predict reward r̂_t using f_arel^θ(τ) for each τ ∈ B_τ
11:     Update φ using {(o_t, a_t, o_{o_{t+1}})} ∈ τ and weighted reward αr̂_t + (1 − α)1_{t=T} R_T
12:     if k mod M is 0 then
13:         for each gradient update do
14:             Sample a batch from B_e, and compute estimate of total loss, loss^_total(θ)
15:             θ ← θ − ∇_θ loss^_total(θ)
16:         end for
17:     end if
18: end for
```

AREL is summarized in Algorithm 1. Parameters ϕ of RL modules and θ of the credit assignment function are randomly initialized. Observations and actions of agents are collected in each episode (Lines 2-6). Trajectories and episodic rewards are stored in an experience buffer B_e (Line 8). The reward $\hat{r}_t$ at each time step for every trajectory in a batch B_τ (sampled from B_e) is predicted (Lines 9-10). The predicted $\hat{r}_t$ changes as θ is updated, but the episode reward R_T remains the same. A weighted sum $\alpha\hat{r}_t + (1-\alpha)\mathbf{1}_{t=T}R_T$ ($\mathbf{1}_{t=T}$ is an indicator function) is used to update ϕ in a stable manner by using a MARL algorithm (Line 11). The credit assignment function f^{θ}_{arel} is updated when M new trajectories are available (Lines 12-17).

4.4 Analysis

In order to establish a connection between redistributed rewards from *Line 10* of Algorithm 1 and the episodic reward, we define *return equivalence* of *decentralized partially observable sequence-Markov decision processes (Dec-POSDP).* This generalizes the notion of return equivalence introduced in [2] in the fully observable setting for the single agent case. A Dec-POSDP is a decision process with Markov transition probabilities but has a reward distribution that need not be Markov. We present a result that establishes that return equivalent Dec-POSDPs will have the same optimal policies.

Definition 4.2. *Dec-POSDPs $\tilde{\mathcal{P}}$ and $\mathcal{P}$ are **return-equivalent** if they differ only in their reward functions but have the same return for any trajectory τ.*

Theorem 4.3. *Given an initial state s_0, return-equivalent Dec-POSDPs will have the same optimal policies.*

According to Definition 4.2, any two return equivalent Dec-POSDPs will have the same expected return for any trajectory τ. That is, $\tilde{R}_0(\tau) = R_0(\tau), \forall\tau$. This is used to prove Theorem 4.3.

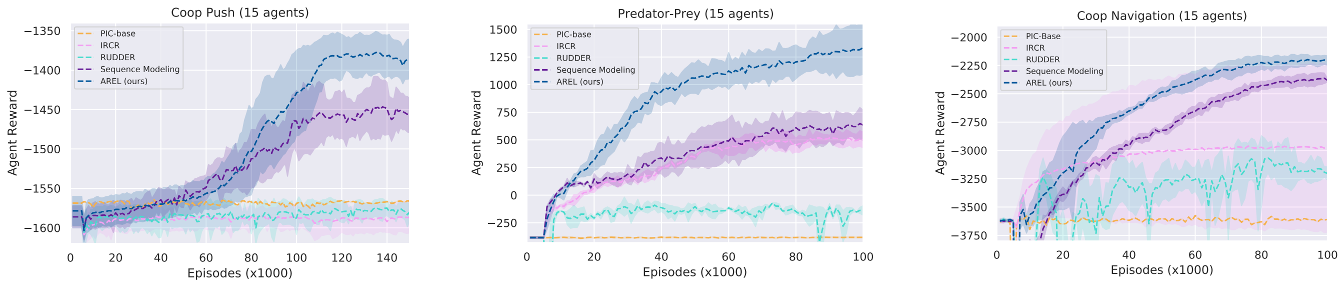
Proof. Consider two return-equivalent Dec-POSDPs $\tilde{\mathcal{P}}$ and $\mathcal{P}$. Since $\tilde{\mathcal{P}}$ and $\mathcal{P}$ have the same transition probability and observation functions, the probabilities that a trajectory τ is realized will be the same if both Dec-POSDPs are provided with the same policy. For any joint agent policy $\pi := (\pi^1, \ldots, \pi^n)$ and sequence of states $s := (s_0, \ldots, s_T)$, we have:

$$\begin{aligned}
&\mathbb{E}_{\tau\sim(\pi,\tilde{Z},\tilde{P})}\big[\tilde{R}_0(\tau)\big] \\
&= \sum_{\tau}\tilde{R}_0(\tau)\underbrace{\sum_{s}\tilde{p}(s_0)\prod_{t=0}^{T-1}\pi(a_t|o_t)\tilde{Z}(o_t|s_t)\tilde{p}(s_{t+1}|s_t,a_t)}_{\tilde{p}^{\pi}(\tau)} \\
&= \sum_{\tau}\tilde{R}_0(\tau)\underbrace{\sum_{s}p(s_0)\prod_{t=0}^{T-1}\pi(a_t|o_t)Z(o_t|s_t)p(s_{t+1}|s_t,a_t)}_{p^{\pi}(\tau)} \\
&= \sum_{\tau}R_0(\tau)\underbrace{\sum_{s}p(s_0)\prod_{t=0}^{T-1}\pi(a_t|o_t)Z(o_t|s_t)p(s_{t+1}|s_t,a_t)}_{p^{\pi}(\tau)} \\
&= \mathbb{E}_{\tau\sim(\pi,Z,P)}\big[R_0(\tau)\big].
\end{aligned}$$

These equations follow from Definition 4.2. Let π^* denote an optimal policy for $\tilde{\mathcal{P}}$. Then, we have:

$$\begin{aligned}
\mathbb{E}_{\tau\sim(\pi^*,\tilde{Z},\tilde{P})}\big[\tilde{R}_0(\tau)\big] &= \mathbb{E}_{\tau\sim(\pi^*,Z,P)}\big[R_0(\tau)\big] \\
\geq \mathbb{E}_{\tau\sim(\pi,\tilde{Z},\tilde{P})}\big[\tilde{R}_0(\tau)\big] &= \mathbb{E}_{\tau\sim(\pi,Z,P)}\big[R_0(\tau)\big].
\end{aligned}$$

Therefore, π^* will also be an optimal policy for $\mathcal{P}$. □



(a) Cooperative Push **(b) Predator and Prey** **(c) Cooperative Navigation**

Figure 2: Average agent rewards and standard deviation for tasks in Particle World with episodic rewards and $N = 15$. *AREL* (dark blue) results in the highest average rewards in all tasks.

When $l_r(\theta) = 0$ in Eqn. (3), a Dec-POSDP with the redistributed reward will be return-equivalent to a Dec-POSDP with the original episodic reward. Theorem 4.3 indicates that in this scenario, the two Dec-POSDPs will have the same optimal policies. An additional result in *Appendix A* gives a bound on $loss_{total}(\theta)$ when the estimators $f^{\theta}_{arel}(E_t)$ are unbiased at each time-step.

5 EXPERIMENTS

In this section, we describe the tasks that we evaluate *AREL* on, and present results of our experiments. Additional results and our code is available at **https://github.com/baicenxiao/AREL**.

5.1 Environments and Tasks

We study tasks from *Particle World* [26] and the *StarCraft Multi-Agent Challenge* [38]. These have been identified as challenging multi-agent environments in [26, 38]. In each task, a reward is received by agents only at the end of an episode. No reward is provided at other time steps. We briefly summarize the tasks below and defer detailed task descriptions to *Appendix B*.

(1) **Cooperative Push**: N agents work together to move a large ball to a landmark.
(2) **Predator-Prey**: N predators seek to capture M preys. L landmarks impede movement of agents.
(3) **Cooperative Navigation**: N agents seek to reach N landmarks. The maximum reward is obtained when there is exactly one agent at each landmark.
(4) **StarCraft**: Units from one group (controlled by RL agents) collaborate to attack units from another (controlled by heuristics). We report results for three maps: 2 Stalkers, 3 Zealots (2s3z); 1 Colossus, 3 Stalkers, 5 Zealots (1c3s5z); 3 Stalkers vs. 5 Zealots (3s_vs_5z).

5.2 Architecture and Training

In order to make the agent-temporal attention module more expressive, we use a transformer architecture with multi-head attention [44] for both agent and temporal attention. The permutation invariant critic (*PIC*) based on the multi-agent deep deterministic policy gradient (MADDPG) from [23] is used as the base RL algorithm in Particle World. In StarCraft, we use QMIX [34] as the base RL algorithm. The value of α is set to 1 in Particle World and 0.8 in StarCraft. Additional details are presented in *Appendix C*.

5.3 Evaluation

We compare *AREL* with three state-of-the-art methods:
(1) ***RUDDER*** [2]: A long short-term memory (LSTM) network is used for reward decomposition along the length of an episode.
(2) ***Sequence Modeling*** [24]: An attention mechanism is used for temporal decomposition of rewards along an episode.
(3) ***Iterative Relative Credit Refinement (IRCR)*** [14]: 'Guidance rewards' for temporal credit assignment are learned using a surrogate objective.

RUDDER and Sequence Modeling were originally developed for the single agent case. We adapted these methods to MARL by concatenating observations from all agents. We added the variance-based regularization loss in our experiments for Sequence Modeling, and observed that incorporating the regularization term resulted in an improved performance compared to without regularization.

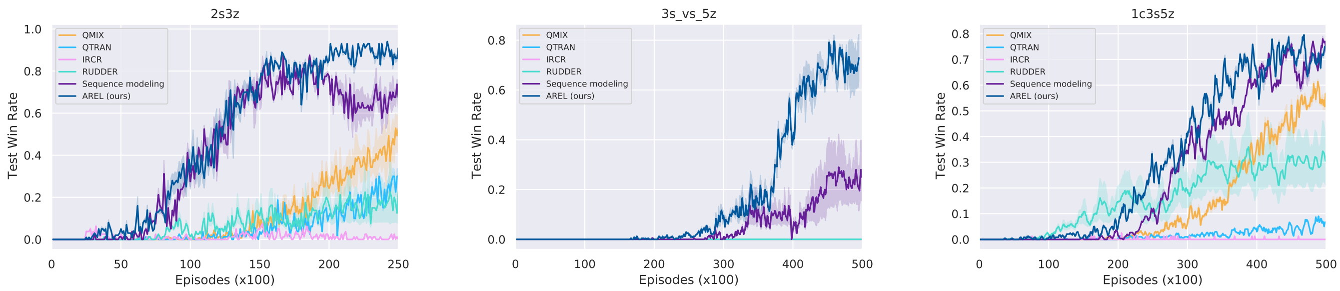
5.4 Results

5.4.1 ***AREL enables improved performance.*** Figure 2 shows results of our experiments for tasks in Particle World for $N = 15$. In each case, *AREL* is able to guide agents to learn policies that result in higher average rewards compared to other methods. This is a consequence of using an attention mechanism to redistribute an episodic reward along the length of an episode, and also characterizing the contributions of individual agents.

The *PIC* baseline [23] fails to learn policies to complete tasks with episodic rewards. A similar result of failure to complete tasks was observed when using *RUDDER* [2]. An explanation for this could be that RUDDER only carries out a temporal redistribution of rewards, but does not consider the effect of agents contributing differently to a reward.

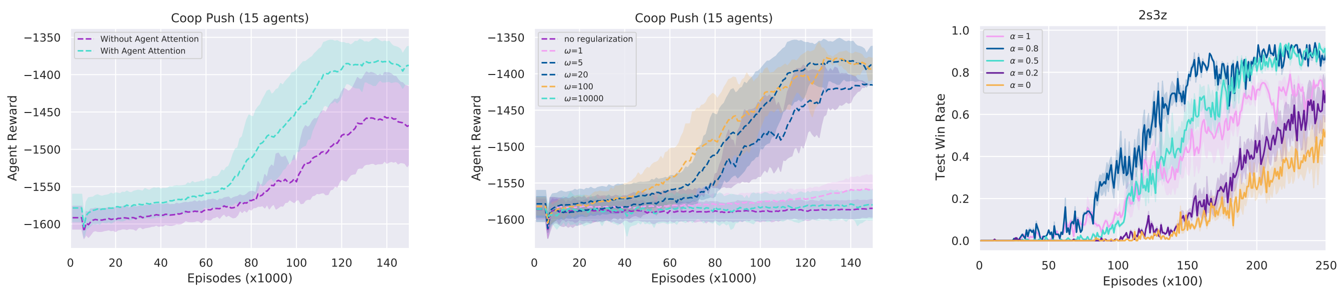
Sequence Modeling [24] performs better than *RUDDER* and the *PIC* baseline, possibly because it uses attention to redistribute episodic rewards. This was shown to outperform LSTM-based models, including *RUDDER*, in [24] in single-agent episodic RL, due to the relative ease of training the attention mechanism. We believe that absence of an explicit characterization of agent-attention resulted in a lower reward for this method compared to *AREL*.

Using a surrogate objective in *IRCR* [14] results in obtaining rewards comparable to *AREL* in some runs in the Cooperative Navigation task. However, the reward when using *IRCR* has a much



(a) 2s3z **(b) 3s vs 5z** **(c) 1c3s5z**

Figure 3: Average test win rate and variance in StarCraft. *AREL* (dark blue) results in the highest win rates in 2s3z and 3s_vs_5z, and obtains a comparable win rate to Sequence Modeling in 1c3s5z.



(a) Agent Attention **(b) Regularization** **(c) Reward Weight**

Figure 4: Ablations: Effects of the agent attention module (Fig. 4a) and regularization parameter ω in Eqn (3) (Fig. 4b) in Cooperative Push, and reward weight α (Fig. 4c) in the 2s3z StarCraft map.

higher variance compared to that obtained when using *AREL*. A possible reason for this is that *IRCR* does not characterize the relative contributions of agents at intermediate time-steps.

Figure 3 shows the results of our experiments for the three maps in StarCraft. *AREL* achieves the highest average win rate in the 2s3z and 3s_vs_5z maps, and obtains a comparable win rate to *Sequence Modeling* in 1c3s5z. *Sequence Modeling* does not explicitly model agent-attention, which could explain the lower average win rates in 2s3z and 3s_vs_5z. *RUDDER* achieves a nonzero, albeit much lower win rate than *AREL* in two maps, possibly because the increased episode length might affect the redistribution of the episode reward for this method. *IRCR* and *QTRAN* [40] obtain the lowest win rates. Additional experimental results are provided in *Appendix D*.

5.4.2 ***Ablations***. We carry out several ablations to evaluate components of *AREL*. Figure 4a demonstrates the impact of the agent-attention module. In the absence of agent-attention (while retaining permutation invariance among agents through the shared temporal attention module), rewards are significantly lower. We study the effect of the value of ω in Eqn. (3) on rewards in Figure 4b. This term is critical in ensuring that agents learn good policies. This is underscored by observations that rewards are significantly lower for very small or very large ω ($\omega = 0, \omega = 1, \omega = 10000$). Third, we evaluate the effect of mixing the original episodic reward and redistributed reward by changing the reward weight α in Figure 4c. The reward mixture influences win rates; $\alpha = 0.5$ or 0.8 yields the highest win rate. The win rate is $\sim 10\%$ lower when using redistributed reward alone ($\alpha = 1$). Additional ablations and evaluating the choice of regularization loss are shown in *Appendices E and F*.

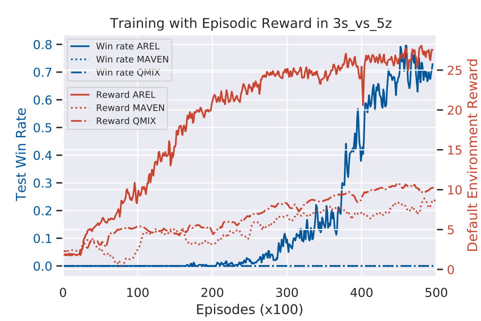


Figure 5: Comparison of AREL with QMIX and a strategic exploration technique, MAVEN in the $3s\_vs\_5z$ **StarCraft map (avg. over 5 runs). AREL yields highest rewards and win rates.**

5.4.3 ***Credit Assignment vs. exploration***. This section demonstrates the importance of effective redistribution of an episodic reward *vis-a-vis* strategic exploration of the environment. The episodic reward $R_T (= \sum_t r_t)$ takes continuous values and provides

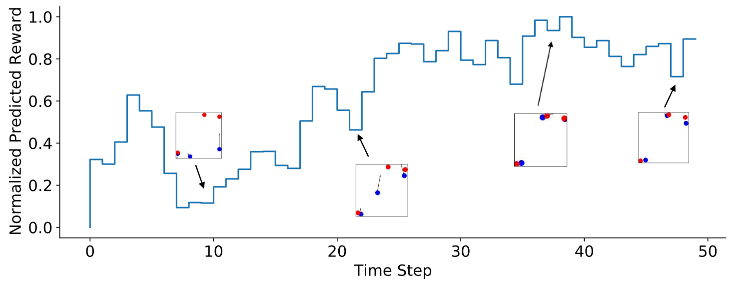


Figure 6: An instantiation of the Cooperative Navigation task with $N = 3$ where rewards are provided only at the end of an episode. Blue and red dots respectively denote agents and landmarks. Arrows on agents represent their directions of movement. The objective of this task is for each agent to cover a distinct landmark. The $y-$axis of the graph shows the $0 - 1$ normalized predicted rewards for a sample trajectory. The positions of agents relative to landmarks are shown at several points along this trajectory. The figure shows a scenario where two agents are close to a single landmark. In this case, one of them must remain close to this landmark, while the other moves towards a different landmark. The predicted redistributed reward encourages such an action, since it has a higher magnitude when agents navigate towards distinct landmarks. The predicted redistributed reward by *AREL* is not uniform along the length of the episode.

fine-grained information on performance (beyond only win/ loss). AREL learns a redistribution of R_T by identifying critical states in an episode, and does not provide exploration abilities beyond that of the base RL algorithm. The redistributed rewards of AREL can be given as input to any RL algorithm to learn policies (in our experiments, we demonstrate using QMIX for StarCraft; MADDPG for Particle World). Figure 5 illustrates a comparison of *AREL* with a state-of-the-art exploration strategy, MAVEN [28] and with QMIX [34] in the $3s\_vs\_5z$ StarCraft map. We observe that when rewards are delayed to the end of an episode, effectively redistributing the reward can be more beneficial than strategically exploring the environment to improve win-rates or total rewards.

5.4.4 *Interpretability of Learned Rewards.*

Figure 6 presents an interpretation of the decomposed predicted rewards *vis-a-vis* the relative positions of agents to landmarks in the Cooperative Navigation task with $N = 3$. When the reward is provided only at the end of an episode, *AREL* is used to learn a temporal redistribution of this episodic reward. The predicted rewards are normalized to a $0 - 1$ scale for ease of representation. The positions of the agents relative to the landmarks are shown at several points along a sample trajectory. Successfully trained agents must learn policies that enable each agent to cover a distinct landmark. For example, in a scenario where two agents are close to a single landmark, one of them must remain close to this landmark, while the other moves towards a different landmark. We observe that the magnitude of the predicted rewards is consistent with this insight in that it is higher when agents navigate away and towards different landmarks.

This visualization in Figure 6 reveals that the attention mechanism in *AREL* is able to learn to redistribute an episodic reward effectively in order to successfully train agents to accomplish task objectives in cooperative multi-agent reinforcement learning. Moreover, it reveals that the redistributed reward predicted by *AREL* is not uniform along the length of the episode.

5.5 Discussion

This paper focused on developing techniques to effectively learn policies in MARL environments when rewards were delayed or episodic. Our experiments demonstrate that *AREL* can be used as a module that enables more effective credit assignment by identifying critical states through capturing long-term temporal dependencies between states and an episodic reward. Redistributed rewards predicted by *AREL* are dense, which can then be provided as an input to MARL algorithms that learn value functions for credit assignment (we used MADDPG [26] and QMIX [34] in our experiments).

By including a variance-based regularization term, the total loss in Eqn. (3) enabled incorporating the possibility that not all intermediate states would contribute equally to an episodic reward, while also learning less sparse redistributed rewards. Moreover, any exploration ability available to the agents was provided solely by the MARL algorithm, and not by *AREL*. We further demonstrated that effective credit assignment was more beneficial than strategic exploration of the environment when rewards are episodic.

6 CONCLUSION

This paper studied the multi-agent temporal credit assignment problem in MARL tasks with episodic rewards. Solving this problem required addressing the twin challenges of identifying the relative importance of states along the length of an episode and individual agent's state at any single time-step. We presented an attention-based method called *AREL* to deal with the above challenges. The temporally redistributed reward predicted by *AREL* was dense, and could be integrated with MARL algorithms. *AREL* was evaluated on tasks from Particle World and StarCraft, and was successful in obtaining higher rewards and better win rates than three state-of-the-art reward redistribution techniques.

ACKNOWLEDGMENT

This work was supported by the Office of Naval Research via Grant N00014-17-S-B001.

REFERENCES

[1] Adrian K Agogino and Kagan Tumer. 2006. QUICR-learning for multi-agent coordination. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 21. 1438.

[2] Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. 2019. RUDDER: Return decomposition for delayed rewards. In *Neural Information Processing Systems*. 13566–13577.

[3] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. 2019. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences* 116, 32 (2019), 15849–15854.

[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The Arcade Learning Environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.

[5] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. 2019. Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations. In *International Conference on Machine Learning*.

[6] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] Sam Devlin and Daniel Kudenko. 2011. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 225–232.

[9] Sam Devlin, Daniel Kudenko, and Marek Grześ. 2011. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems* 14, 02 (2011), 251–278.

[10] Sam Devlin, Logan Yliniemi, Daniel Kudenko, and Kagan Tumer. 2014. Potential-based difference rewards for multiagent reinforcement learning. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 165–172.

[11] Pedro Domingos. 2000. A unified bias-variance decomposition for zero-one and squared loss. *AAAI/IAAI* 2000 (2000), 564–569.

[12] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*. 2974–2982.

[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The Elements of Statistical Learning*. Springer Series in Statistics.

[14] Tanmay Gangwani, Yuan Zhou, and Jian Peng. 2020. Learning Guidance Rewards with Trajectory-space Smoothing. In *Neural Information Processing Systems*.

[15] Stuart Geman, Elie Bienenstock, and René Doursat. 1992. Neural networks and the bias/variance dilemma. *Neural computation* 4, 1 (1992), 1–58.

[16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep Learning*. MIT Press.

[17] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*. 66–83.

[18] Sili Huang, Bo Yang, Hechang Chen, Haiyin Piao, Zhixiao Sun, and Yi Chang. 2020. MA-TREX: Multi-agent Trajectory-Ranked Reward Extrapolation via Inverse Reinforcement Learning. In *International Conference on Knowledge Science, Engineering and Management*. Springer, 3–14.

[19] Shariq Iqbal and Fei Sha. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*. 2961–2970.

[20] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. 2019. Graph Convolutional Reinforcement Learning. In *International Conference on Learning Representations*.

[21] Jiechuan Jiang and Zongqing Lu. 2018. Learning attentional communication for multi-agent cooperation. In *Neural Information Processing Systems*.

[22] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. 2019. HG-DAgger: Interactive Imitation Learning with Human Experts. In *International Conference on Robotics and Automation*. IEEE, 8077–8083.

[23] Iou-Jen Liu, Raymond A Yeh, and Alexander G Schwing. 2020. PIC: Permutation invariant critic for multi-agent deep reinforcement learning. In *Conference on Robot Learning*. 590–602.

[24] Yang Liu, Yunan Luo, Yuanyi Zhong, Xi Chen, Qiang Liu, and Jian Peng. 2019. Sequence Modeling of Temporal Credit Assignment for Episodic Reinforcement Learning. *arXiv preprint arXiv:1905.13420* (2019).

[25] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. 2020. Multi-agent game abstraction via graph attention neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7211–7218.

[26] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Neural Information Processing Systems*. 6379–6390.

[27] Xiaosong Lu, Howard M Schwartz, and Sidney Nascimento Givigi. 2011. Policy invariance under reward transformations for general-sum stochastic games. *Journal of Artificial Intelligence Research* 41 (2011), 397–406.

[28] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. MAVEN: Multi-agent variational exploration. *arXiv preprint arXiv:1910.07483* (2019).

[29] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, and Zhibo Gong. 2019. Modelling the Dynamic Joint Policy of Teammates with Attention Multi-agent DDPG. In *Proceedings of the 18th International Conference on Autonomous Agents and Multi-Agent Systems*. 1108–1116.

[30] Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. 2018. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591* (2018).

[31] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*. 278–287.

[32] Frans A Oliehoek, Christopher Amato, et al. 2016. *A concise introduction to decentralized POMDPs*. Vol. 1. Springer.

[33] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. 2017. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics* 9, 1 (2017), 48.

[34] 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 *International Conference on Machine Learning*. 4295–4304.

[35] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*. 627–635.

[36] Heechang Ryu, Hayong Shin, and Jinkyoo Park. 2020. Multi-agent actor-critic with hierarchical graph attention network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7236–7243.

[37] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2017. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging* 2017, 19 (2017), 70–76.

[38] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. *CoRR* abs/1902.04043 (2019).

[39] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016).

[40] 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 *International Conference on Machine Learning*. 5887–5896.

[41] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward.. In *International Conference on Autonomous Agents and Multiagent Systems*. 2085–2087.

[42] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press.

[43] 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 (2017), e0172395.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*. 5998–6008.

[45] Martin J Wainwright. 2019. *High-dimensional statistics: A non-asymptotic viewpoint*. Cambridge University Press.

[46] Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. 2020. Shapley Q-value: A local reward approach to solve global reward games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7285–7292.

[47] Yaodong Yang, Jianye Hao, Guangyong Chen, Hongyao Tang, Yingfeng Chen, Yujing Hu, Changjie Fan, and Zhongyu Wei. 2020. Q-value Path Decomposition for Deep Multiagent Reinforcement Learning. In *International Conference on Machine Learning*.

[48] Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J Smola. 2017. Deep Sets. In *Neural Information Processing Systems*.

[49] Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. 2020. Learning Implicit Credit Assignment for Cooperative Multi-Agent Reinforcement Learning. In *Neural Information Processing Systems*.