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

Baichen 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

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 [5]. Examples of systems where MARL has been used include autonomous vehicle coordination [31], and video games [32, 37].

One approach to enable better collaboration is to use a single centralized controller that can access observations of all agents [16]. 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 are 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 [21, 28], 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 [10], QMIX [28] and QTRAN [34]. 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 [36]. In many applications, rewards may be delayed. For example, in molecular design [27], Go [33], and computer games such as Skiing [3], 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 [19]. These methods do not directly scale well to MARL since size of the joint observation space grows exponentially with number of agents [21].

Besides scalability, addressing temporal credit assignment in MARL with episodic rewards presents two challenges: identifying 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 [38] 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,
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 [11] 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 capturing long-term dependencies between states and the episodic reward to identify critical states.

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 pressing 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 [11, 19]. 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 [21], and three combat scenarios in the StarCraft Multi-Agent Challenge [32]. 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.

Additional analysis and experiments are presented in an Appendix available at https://arxiv.org/abs/2201.04612.

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 [25] and multi-agent [7, 22] RL, and was shown to accelerate learning of policies in [8]. Credit assignment was also studied by incorporating human feedback through imitation learning [17, 29] and demonstrations [4, 13].

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. In parallel, the authors of [19] 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 [21].

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, 9, 10] by computing a counterfactual term that marginalized out actions of that agent while keeping actions of other agents fixed. Value decomposition networks, proposed in [35], decomposed a centralized value into a sum of agent values to assess the contributions of each agent. A monotonicity assumption on value functions was imposed in QMIX [28] to assign credit to individual agents. A generalized approach to decompose a joint value into individual agent values was presented in QTRAN [34]. The Shapley Q-value was used in [39] to distribute a global reward to identify each agent’s contribution. The authors of [40] decomposed global Q-values along trajectory paths, while [42] 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 [24] used an attention mechanism with a CTDE-based algorithm to enable each agent to effectively model policies of other agents (from its own perspective). Hierarchical graph attention networks proposed in [30] modeled hierarchical relationships among agents and used two attention networks to effectively represent individual and group level interactions. The authors of [15, 20] 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 [11]. 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) [26]. A Dec-POMDP is a tuple $G = (S, A, P, r, Z, O, n, y)$, where $s \in S$ describes the environment state. Each agent $i \in \{1, 2, \ldots, n\}$ receives an observation $o^i \in O^i$ according to an observation function.
The output is the redistributed reward, which is integrated with MARL algorithms (e.g., MADDPG, QMIX) to learn agent policies. 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 $E \in \mathbb{R}^{T \times N \times D}$. The objective is to learn a mapping $\mathcal{E}_{arel}(E) : \mathbb{R}^{T \times N \times D} \rightarrow \mathbb{R}^{T \times N \times D}$ to assign credit to the agents at each time-step. The information in a trajectory $E$ 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 [38]. 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 $E$ at different time-steps are related (along the ‘first’ dimension of $E$). The agent attention module determines how agents influence one another (along the ‘second’ dimension of $E$).
obtain the transpose of \( E \) as \( \hat{E} \in \mathbb{R}^{N \times T \times D} \). Adopting notation from \cite{38}, each row \( \mathbf{e}_i \in \mathbb{R}^{T \times D} \) of \( \hat{E} \) is transformed to a query \( Q_{t,i}^{\text{tem}} := e_i W_q^{\text{tem}} \), key \( K_{t,i}^{\text{tem}} := e_i W_k^{\text{tem}} \), and value \( V_{t,i}^{\text{tem}} := e_i W_v^{\text{tem}} \), where \( W_q^{\text{tem}}, W_k^{\text{tem}}, W_v^{\text{tem}} \in \mathbb{R}^{D \times D} \) are learnable parameters, and \( i \in \{0, \ldots, N-1\} \). The \( t^{th} \) row \( x_{t,i} \) of the temporal attention feature \( X_t \in \mathbb{R}^{T \times D} \) is a weighted sum, \( \tilde{x}_{t,i} := d_{t,i}^{\text{tem}} \). The temporal attention weight vector \( a_{t,i} \in \mathbb{R}^{N \times 1} \) is a normalization of the inner-product between the \( t^{th} \) row of \( Q_{t,i}^{\text{tem}}, K_{t,i}^{\text{tem}} \), and the key \( K_{t,i}^{\text{tem}} \):

\[
a_{t,i}^T = \text{softmax} \left( \frac{q_{t,i}^{\text{tem}} k_{t,i}^{\text{tem}}}{|D|} \right),
\]

where \( \circ \) 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 \cite{6} 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 \( X \in \mathbb{R}^{N \times T \times D} \), obtained by stacking \( x_t, i \in \{0, \ldots, N-1\} \). The temporal attention process can be described by a function \( f_{\text{tem}}(E) \to X \).

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 \( X \), denoted \( \hat{X} \in \mathbb{R}^{T \times N \times D} \). Each row of \( \hat{X} \), \( x_t \in \mathbb{R}^{T \times N \times D} \) is transformed to a query \( Q_t^{\text{agt}} := x_t W_q^{\text{agt}} \), key \( K_t^{\text{agt}} := x_t W_k^{\text{agt}} \), and value \( V_t^{\text{agt}} := x_t W_v^{\text{agt}} \). Here, \( W_q^{\text{agt}}, W_k^{\text{agt}}, W_v^{\text{agt}} \in \mathbb{R}^{D \times D} \) are learnable parameters. The \( i^{th} \) row \( z_{t,i} \) of the agent attention feature \( Z_t \in \mathbb{R}^{T \times N} \) is a weighted sum, \( z_{t,i} = \beta_{t,i} V_t^{\text{agt}} \). The agent attention weight vector \( \beta_{t,i} \in \mathbb{R}^{N \times 1} \) is similarly defined to the temporal attention weight vector in Eqn. (1) but without the masking operation since causality need not be maintained. Therefore, \( \beta_{t,i}^T = \text{softmax} \left( \frac{q_{t,i}^{\text{agt}} k_{t,i}^{\text{agt}}}{|D|} \right) \).

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

4.1.3 Concatenating Attention Modules. The output of the temporal attention module is an entity \( X \) that attends to information at time-steps along the length of an episode for each agent. Passing \( X \) through the agent attention module results in an output \( Z \) 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: \( f_{\text{agt}} := f_{\text{tem}} \circ f_{\text{agt}} \). 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 \( f_{\text{arel}} = f_{\text{agt}} \circ (f_{\text{att}} \circ \cdots \circ f_{\text{att}}) \), where \( f_{\text{att}} : \mathbb{R}^T \to \mathbb{R}^T \). 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 \( a_1 \) and \( a_2 \) cooperate to achieve a goal, the reward when \( a_1 \) observes \( o_{b1} \) and \( a_2 \) observes \( o_{b2} \) should be the same as the case when \( a_1 \) observes \( o_{b2} \) and \( a_2 \) observes \( o_{b1} \). 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 \([11, 18]\). When this property is satisfied, the output of the function \( f_{\text{arel}} \) should be invariant to the order of the agents’ observations. Formally, if the set of all permutations along the agent dimension (second dimension of \( E \)) is denoted \( \mathcal{H} \), then \( f_{\text{arel}}(h_1(E)) = f_{\text{arel}}(h_2(E)) \) must be true for all \( h_1, h_2 \in \mathcal{H} \).

The function \( f_{\text{arel}} \) is permutation invariant along the agent dimension by design. A sufficient condition for \( f_{\text{arel}} \) to be permutation invariant is that the function \( f_e \) 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 \( \hat{r}_t := g_2(\sum_{i=0}^{T-1} g_1(z_{t,i})) \). It was shown in \([41]\) 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 \( f_{\text{arel}}(E) \) is a vector in \( \mathbb{R}^T \), its \( t^{th} \) entry is denoted \( f_{\text{arel}}(E)_t \). The sequence \( \{ \hat{r}_t \}_{t=0}^T \) is learned by minimizing a regression loss, \( L_r(\theta) := \mathbb{E}_{E,R}[\frac{1}{2} \left( \sum_{t=0}^T (f_{\text{arel}}(E)_t - R_t)^2 \right) \] , where \( \theta \) 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 \([7]\). We observe that more than one combination of \( \{ \hat{r}_t \}_{t=0}^T \) can minimize \( L_r(\theta) \). We add a regularization loss \( L_{\text{reg}}(\theta) \) to select among solutions that minimize \( L_r(\theta) \). Specifically, we aim to choose a solution that also minimizes the variance \( L_\sigma(\theta) \) of the redistributed rewards, and set \( L_{\text{reg}}(\theta) = L_\sigma(\theta) \) (we examine other choices of \( L_{\text{reg}}(\theta) \) in the Appendix). Using \( L_\sigma(\theta) \) 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 \( f^\theta_{\text{rel}} \) is:

\[
\text{loss}_{\text{total}}(\theta) = l_\mu(\theta) + \omega l_\rho(\theta),
\]

where \( l_\mu(\theta) := \mathbb{E}_k \left[ \frac{1}{T} \sum_t (f^\theta_{\text{rel}}(E_t) - \bar{f}^\theta(E_t))^2 \right] \), and \( \bar{f}^\theta(E_t) := \left( \frac{\sum_t f^\theta_{\text{rel}}(E_t)}{T} \right) / T \). 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\( \theta \{ \text{loss}_{\text{total}}(\theta) \} \) will not typically yield \( l_\mu(\theta) = l_\rho(\theta) = 0 \) (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, \( \hat{l}_\mu(\theta) = 0 \) and \( \hat{l}_\rho(\theta) = 0 \) 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 \( \theta, \phi \) 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 \( k = 0, \ldots \) do
2. Reset episode return \( R_T \leftarrow 0 \); Reset trajectory for current episode \( \tau \leftarrow \emptyset \)
3. for step \( t = 0, \ldots, T - 1 \) do
4. Sample action \( a_i^t \sim \pi^t_{\phi}(\theta_i^t) \), for \( i = 0, \ldots, N - 1 \)
5. Take action \( a_t \); Observe \( o_{t+1} = (o^0_{t+1}, \ldots, o^N_{t+1}) \)
6. Store transition \( \tau \leftarrow \tau \cup \{(o_t, a_t, o_{t+1})\} \)
7. end for
8. Update episode reward \( R_T \); Store trajectory \( B_e \leftarrow B_e \cup (\tau, R_T) \)
9. Sample a batch of trajectories \( B_e \sim B_e \)
10. Predict reward \( \hat{r}_t \) using \( f^\theta_{\text{rel}}(\tau) \) for each \( \tau \in B_T \)
11. Update \( \phi \) using \( \{(o_t, a_t, o_{t+1})\} \in \tau \) and weighted reward \( \alpha \hat{r}_t + (1 - \alpha)\forall \tau \in \hat{R}_T \)
12. if \( k \text{ mod } M = 0 \) then
13. for each gradient update do
14. Sample a batch from \( B_e \), and compute estimate of total loss, \( \text{loss}_{\text{total}}(\theta) \)
15. \( \theta \leftarrow \theta - \nabla \text{loss}_{\text{total}}(\theta) \)
16. end for
17. end if
18. end for

AREL is summarized in Algorithm 1. Parameters \( \phi \) of RL modules and \( \theta \) 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 \( B_T \) (sampled from \( B_e \)) is predicted (Lines 9-10). The predicted \( \hat{r}_t \) changes as \( \theta \) is updated, but the episode reward \( R_T \) remains the same. A weighted sum \( \alpha \hat{r}_t + (1 - \alpha)\forall \tau \in \hat{R}_T \) (1=\( \forall \tau \) is an indicator function) is used to update \( \phi \) in a stable manner by using a MARL algorithm (Line 11). The credit assignment function \( f^\theta_{\text{rel}} \) 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 identical optimal policies.

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

**Theorem 4.3.** Given an initial state \( s_0 \), return-equivalent Dec-POSDPs will have identical optimal policies.

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

**Proof.** Since return equivalent Dec-POSDPs \( \hat{\mathcal{P}} \) and \( \mathcal{P} \) have the same transition probability and observation functions, the probabilities that a trajectory \( \tau \) is realized will be the same if both Dec-POSDPs are given the same policy. For any joint agent policy \( \pi := (\pi^1, \ldots, \pi^N) \) and sequence of states \( s := (s_0, s_T) \),

\[
\mathbb{E}_{\tau \sim (\pi, \mathcal{Z}, \mathcal{P})}[\hat{R}_0(\tau)]
\]

\[
= \sum_{\tau} \hat{R}_0(\tau) \sum_s p(s_0) \prod_{t=0}^{T-1} \pi(a_t|o_t)Z(o_{t+1}|s_t, a_t)\forall \tau \in \hat{R}_T
\]

\[
= \sum_{\tau} \hat{R}_0(\tau) \sum_s p(s_0) \prod_{t=0}^{T-1} \pi(a_t|o_t)Z(o_{t+1}|s_t, a_t)\forall \tau \in \hat{R}_T
\]

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

\[
\mathbb{E}_{\tau \sim (\pi^*, \mathcal{Z}, \mathcal{P})}[\hat{R}_0(\tau)] = \mathbb{E}_{\tau \sim (\pi, \mathcal{Z}, \mathcal{P})}[\hat{R}_0(\tau)]
\]

\[
\geq \mathbb{E}_{\tau \sim (\pi, \mathcal{Z}, \mathcal{P})}[\hat{R}_0(\tau)] = \mathbb{E}_{\tau \sim (\pi, \mathcal{Z}, \mathcal{P})}[R_0(\tau)]
\]

Therefore, \( \pi^* \) will also be an optimal policy for \( \mathcal{P} \).
We study tasks from Agent Challenge [32]. These have been identified as challenging (dark blue) results in the highest average rewards in all tasks.

StarCraft (4) The maximum reward is obtained when there is exactly one agent (3) Cooperative Navigation (2) Predator-Prey (1c3s5z); 3 Stalkers vs. 5 Zealots (3s_vs_5z).

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 [21] and the StarCraft Multi-Agent Challenge [32]. These have been identified as challenging multi-agent environments in [21, 32]. 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 [38] for both agent and temporal attention. The permutation invariant critic (PIC) [11]: ‘Guidance estimators' 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.

5.3 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** [19]: An attention mechanism is used for temporal decomposition of rewards along an episode.
3. **Iterative Relative Credit Refinement (IRCR)** [11]: ‘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 characterizing the contributions of individual agents.

The PIC baseline [18] 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, while RUDDER carries out a temporal redistribution of rewards, it does not consider the effect of agents contributing differently to a reward.

Sequence Modeling [19] 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 [19] 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 [11] results in obtaining rewards comparable to AREL in some runs in the Cooperative Navigation task. However, the reward when using IRCR has a much
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 [34] 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 \( \omega \) 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 \( \omega (\omega = 0, \omega = 1, \omega = 10000) \). Third, we evaluate the effect of mixing the original episodic reward and redistributed reward by changing the reward weight \( \alpha \) 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 4: Ablations: Effects of the agent attention module (Fig. 4a) and regularization parameter \( \omega \) in Eqn (3) (Fig. 4b) in Cooperative Push, and reward weight \( \alpha \) (Fig. 4c) in the 2s3z StarCraft map.

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 \( R_T = \sum r_t \) takes continuous values and provides

![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.](image)

![Figure 5](image)
The objective of this task is for each agent to cover a distinct 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.

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 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.

5.5 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 [21] and QMIX [28] 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.

ACKNOWLEDGMENTS

This work was supported by the Office of Naval Research via Grant N00014-17-S-B001. We thank Dr. Sreeram Kannan and Dr. Sukarno Mertoguno for insightful discussions.
REFERENCES


