Abstract

Recently, deep multiagent reinforcement learning (MARL) has become a highly active research area as many real-world problems can be inherently viewed as multiagent systems. A particularly interesting and widely applicable class of problems is the partially observable cooperative multiagent setting, in which a team of agents learns to coordinate their behaviors conditioning on their private observations and commonly shared global reward signals. One natural solution is to resort to the centralized training and decentralized execution paradigm and during centralized training, one key challenge is the multiagent credit assignment: how to allocate the global rewards for individual agent policies for better coordination towards maximizing system-level’s benefits. In this paper, we propose a new method called Q-value Path Decomposition (QPD) to decompose the system’s global Q-values into individual agents’ Q-values. Unlike previous works which restrict the representation relation of the individual Q-values and the global one, we leverage the integrated gradient attribution technique into deep MARL to directly decompose global Q-values along trajectory paths to assign credits for agents. We evaluate QPD on the challenging StarCraft II micromanagement tasks and show that QPD achieves the state-of-the-art performance in both homogeneous and heterogeneous multiagent scenarios compared with existing cooperative MARL algorithms.

1. Introduction

The cooperative multiagent reinforcement learning problem has attracted increasing research attention in the last decade (Busoniu et al., 2008; Gupta et al., 2017; Palmer et al., 2018), where a system of agents learn towards coordinated policies to optimize the accumulated global rewards. Cooperative multiagent systems (MAS) have been demonstrated beneficial in numerous applications, e.g., the coordination of autonomous vehicles (Cao et al., 2012) and optimizing the productivity of a factory in distributed logistics (Ying & Sang, 2005). One natural way to address the cooperative MARL problem is the centralized approach, which views the overall MAS as a whole and solves it as a single-agent learning task. In such settings, existing reinforcement learning (RL) techniques can be leveraged to learn joint optimal policies based on agents’ joint observations and common rewards (Tan, 1993). However, the centralized approach usually does not scale well, since the joint action space of agents grows exponentially as the agent number increases. Furthermore, centralized approaches may not be applicable in practical settings where only distributed policies can be deployed due to private observations and communication constraints (Foerster et al., 2018), i.e., each agent can only decide how to behave based on its local observations.

To address these above limitations, researchers resort to decentralized approaches, in which each agent learns its optimal policy independently based on its local observations and individual rewards. However, in cooperative multiagent environments, all agents receive the same global reward signal. Letting individual agents learn concurrently based on the global reward (aka. independent learners) has been well studied (Tan, 1993), but sometimes even shown to be difficult in even simple two-agent, single-state stochastic coordination problems. One main reason is that the global reward signal brings the nonstationarity that agents cannot distinguish between the stochasticity of the environment and the explorative behaviors of other co-learners (Lowe et al., 2017), thus may mistakenly update their policies. Therefore, the key of coordinating agents is to correctly allocate the reward signal for each agent, which is also known as the multiagent credit assignment problem (Chang et al., 2004).
sign agent-wise reward function based on domain knowledge. However, the heuristic design requires manual efforts and is not always applicable in complex cooperative multiagent tasks. It would be more desirable if there is any generalized principle to learn agent-wise reward functions. Foerster et al. (Foerster et al., 2018) propose a multiagent actor-critic method called counterfactual multiagent (COMA) policy gradients, which marginalizes out a single agent’s action while keeping the other agents’ actions fixed to calculate the advantage for agent policies. At the same time, Sunehag et al. (Sunehag et al., 2018) propose a value decomposition network (VDN) to decompose the global value function into agent-wise value functions in terms of local observations, which is not applicable for complex systems where agents have complicated relations and the decomposition is not accurate as the global information is not fully utilized. QMIX relaxes the limitation of the linear decomposition of global Q-value into individual ones in VDN while enforcing a monotonicity constraint among them. However, both VDN and QMIX restrict the relation representation between the individual Q-values and the global Q-value. Such a way restricts the accuracy of the individual Q-values and may impede the learning of coordinated policies in complex multiagent scenarios. Recently, QTRAN (Son et al., 2019) is proposed to guarantee optimal decentralization inheriting the linear constraints between the global Q-value and agent-wise ones, and avoids the representation limitations introduced by VDN and QMIX. But the constraints on the optimization problem are computationally intractable and practical relaxations may lead to the unsatisfactory performance in complex tasks (Mahajan et al., 2019).

In this paper, we propose a novel Q-value decomposition technique from the perspective of deep learning (DL). Similar to previous works, we set in a centralized learning and decentralized execution paradigm, where agents are trained centrally with shared information while executing in a decentralized manner. Our method employs integrated gradients (Sundararajan et al., 2017) method to analyze the contribution of each agent to the global Q-value $Q_{tot}$, and regards the contribution of each agent as its individual $Q_i$, which is used as the supervision signal to train each agent’s Q-value function. As we utilize trajectories of RL to implement attribution decomposition, we call this method Q-value Path Decomposition (QPD). Besides, we design a multi-channel attribution decomposition, we call this method Q-value Path Decomposition for Deep Multiagent Reinforcement Learning.

There have seen many related contributions to the setting of the decentralized partially observable Markov decision process (Dec-POMDPs). For the large-scale MAS setting, Duc Thien Nguyen et al. (Nguyen et al., 2017; 2018) study the Collective Dec-POMDPs where agent interactions are dependent on their collective influence on each other rather than their identities. At the same time, Yang et al., (Yang et al., 2018) assume that each agent is affected by its neighbors to reduce the nonstationary phenomenon and derive a mean-field approach. The above two methods are only investigated in the large-scale multiagent settings and satisfy the theoretical support under the large-scale assumption. Another notable direction is the multiagent exploration problem. Mahajan et al., (Mahajan et al., 2019) propose MAVEN to solve it, where value-based agents condition their behavior on the shared latent variable controlled by a hierarchical policy. The latent space which controls the exploration of joint behaviors mainly affects the agent’s individual utility network and is orthogonal to ours.

The remainder of this paper is organized as follows. We introduce the Dec-POMDPs and integrated gradients in Section 2. Then in Section 3, we explain our QPD framework for deep MARL in detail. Next, we validate our methods in the challenging StarCraft II platform in Section 4. Finally, conclusions and future work are provided in Section 5.

2. Background

2.1. Dec-POMDPs

Fully cooperative multiagent tasks can be modeled as Dec-POMDPs (Oliehoek & Amato, 2016). Formally, a Dec-POMDP $G$ is given by a tuple

$$G = \langle S, A, P, r, Z, O, n, \gamma \rangle$$

where $s \in S$ describes the true state of the environment. Dec-POMDPs consider partially observable scenarios in which an observation function $Z(s,i) : S \times N \rightarrow P(O)$, which defines the probability distribution of the observations $o^i \in O^i$ for each agent $i \in N \equiv \{1, \ldots, n\}$ draws individually. At each time step, each agent $i$ selects its action $a_i \in A_i$ based on its local observation $o_i$ according to its stochastic policy $\pi_i : O_i \times A_i \rightarrow [0,1]$. The joint action $\bar{a} \in \bar{A}$ produces the next state according to the state transition function $P : S \times A_1 \times \ldots \times A_n \rightarrow S$. All agents share the same reward function $r(s,\bar{a}) : S \times \bar{A} \rightarrow R$. All agents coordinate together to maximize the total expected return

$$J = \mathbb{E}_{\pi_1,\ldots,\pi_n} r \left( s_0, \bar{a}_0 \right) \sum_{t=0}^{T} \gamma^t r_t(s_t, \bar{a}_t)$$

where $\gamma$ is a discount factor and $T$ is the time horizon. Our problem setting follows the paradigm of centralized training and decentralized execution (Foerster et al., 2018). That is, each agent executes its policy in a distributed manner, since agents may only observe the partial environmental information due to physical limitations (e.g., scope or interfere) and
high communication cost in practice. Fortunately, each agent’s policy can be trained in a centralized manner (using a simulator with additional global information) to improve the learning efficiency. The global discounted return is \( R_t = \sum_{\tau=t}^{\infty} \gamma^l r_{\tau+t} \). The agents’ joint policy induces a value function, i.e., an approximation of expectation over \( R_t \), \( V^\pi(s_t) = E_{\tilde{\pi},s_{t+1} \sim \bar{p}[R_t|s_t]} \), and a global action-value \( Q^\pi(s_t, \tilde{a}_t) = E_{\tilde{\pi},s_{t+1} \sim \bar{p}[R_t|s_t, \tilde{a}_t]} \) remarked as \( Q_{\text{tot}} \).

2.2. Integrated Gradients

A lot of works intend to understand the input-output behaviors of deep networks and attribute the prediction of a deep network to its input features (Ancona et al., 2018). The goal of attribution methods is to determine how much influence each component of input features has in the network output value (Brasó Andilla, 2018).

**Definition 1.** Formally, suppose we have a function \( F : \mathbb{R}^d \rightarrow \mathbb{R} \) that represents a deep network, and an input \( \tilde{x} = (x_1, \ldots, x_j, \ldots, x_d) \in \mathbb{R}^d \). \( \mathbb{R} \) is the set of real numbers. \( F \) is the function with a d-dimension vector input. An attribution of the prediction at input \( \tilde{x} \) relative to a baseline input \( \tilde{b} \) is a vector \( A_F(\tilde{x}, \tilde{b}) = (c_1, \ldots, c_j, \ldots, c_d) \in \mathbb{R}^d \), where \( c_j \) is the contribution value of \( x_j \) to the difference between prediction \( F(\tilde{x}) \) and the baseline prediction \( F(\tilde{b}) \).

The attribution methods are widely studied (Baehrens et al., 2010; Binder et al., 2016; Montavon et al., 2018). As one of them, integrated gradients makes use of path integral to aggregate the gradients along the inputs that fall on the lines between the baseline and the input (Sundararajan et al., 2017), which is inspired by economic cost-sharing literature (Tarassenko et al., 2016) with theoretical supports (Hazewinkel, 1990). The integrated gradients explains how much one feature affects the deep network output while changing from \( F(\tilde{b}) \) to \( F(\tilde{x}) \) along a straight line between \( \tilde{x} \) and \( \tilde{b} \). Using integrated gradients along a path satisfies Sensitivity and Implementation Invariance that attribution methods ought to satisfy (Sundararajan et al., 2017). Although integrated gradients uses the straight line, there are many paths that monotonically interpolate between the two points, and each such path will yield a different attribution method depicting the feature changing process. The path integral focuses on the changing process of each variable to perform attribution and has shown impressive performance in various domains.

Formally, let \( \tau(\alpha) : [0, 1] \rightarrow \mathbb{R}^d \) be a smooth path function specifying a path in \( \mathbb{R}^d \) from the baseline \( \tilde{b} \) to the input \( \tilde{x} \), i.e., \( \tau(0) = \tilde{b} \) and \( \tau(1) = \tilde{x} \). Given a path function \( \tau \), path integrated gradients are obtained by integrating gradients along the path \( \tau(\alpha) \) for \( \alpha \in [0, 1] \). Mathematically, path integrated gradients along the \( j \)th dimension for input \( \tilde{x} \) (i.e., \( x_j \)) on the path \( \tau \) is defined as follows.

\[
c_{x_j} = \text{PathIG}_{x_j}^\tau(\tilde{x}) := \int_{\alpha=0}^{1} \frac{\partial F(\tau(\alpha))}{\partial \tau_j(\alpha)} \frac{\partial \tau_j(\alpha)}{\partial \alpha} d\alpha, \quad (2)
\]

where \( \frac{\partial F(\tau(\alpha))}{\partial \tau_j(\alpha)} \) is the gradient of \( F \) along the \( j \)th dimension.

Attribution methods based on path integrated gradients are collectively known as path methods. Sundararajan et al. first introduce path integrated gradients to perform attribution for deep networks. Due to the absence of the real feature varying path, they specify the straight line as the path for integration. Using the straight line path \( \tau(\alpha) = b + \alpha(\tilde{x} - \tilde{b}) \) for \( \alpha \in [0, 1] \), the integrated gradients (Sundararajan et al., 2017) to calculate the contribution value \( c_{x_j} \) along the \( j \)th dimension for input \( \tilde{x} \) is defined as follows.

\[
c_{x_j} = \text{IG}_{x_j}^\tau(\tilde{x}) := (\tilde{x}_j - \tilde{b}_j) \int_{\alpha=0}^{1} \frac{\partial F(\tau(\alpha))}{\partial \tau_j(\alpha)} d\alpha. \quad (3)
\]

In the computer vision and natural language processing domains, when applying integrated gradients, the zero embedding vector is usually used as the baseline \( \tilde{b} \). Besides, as mentioned above, the straight line is the choice for the path. It seems there are no better path choices for the image models or natural language models as the feature varying process is unknown. The zero-vector baseline and corresponding straight line are not suitable for many real problems as they do not really reflect how features change. For example, in an episode of RL, transitions of state and action features happens between every two adjacent steps from time \( t \) to \( T \). Such a feature varying process cannot be depicted by the straight line from the starting state to the all-zero vector.

3. QPD for MARL

Here we describe our QPD MARL framework and Figure 1 shows the overall learning framework. First, we leverage integrated gradients techniques on the centralized critic to decompose \( Q_{\text{tot}} \) into individual Q-values \( Q \) approximately for each agent in Section 3.1. Such a decomposition process addresses the multiagent credit assignment via the co-variation analysis of each agent’s observations and actions along the trajectory path. The decomposed individual value which approximates \( Q \) is used as the supervision signal to train each agent’s recurrent Q-value network. Then, in Section 3.2, we design a multiagent multi-channel critic which consists of modular channels to extract hidden states for different groups of agents to learn the global Q-value \( Q_{\text{tot}} \) from agents’ joint observations and actions. Finally, we give the algorithm details and training losses in Section 3.3.
As mentioned above, in DL, it is usually unknown how features change from input to baseline. Thus, the straight line becomes the path choice for using integrated gradients in DL. However, in RL, a natural path luckily exists, which could be depicted by connecting the trajectory of state-action transitions in each episode to reflect how the state-action features change. As the trajectory path depicts the real feature varying process, we could achieve an accurate attribution. Using integrated gradients on the trajectory path, we perform the global Q-value decomposition by attributing the global Q-value prediction to its input features. Applying integrated gradients into RL was first studied in RUDDER (Arjona-Medina et al., 2018) to address the sparse delayed reward problem in single-agent RL and has shown excellent performance. However, one weakness of their approach is that they regard the zero vector as the baseline for all states, which ignores real state-action transitions. Another limitation is that they do not use the trajectory path but the straight line between current states and the zero vector as the path when applying integrated gradients, thus making the decomposition inaccurate. Different from RUDDER, we utilize the basic trajectory concept in RL to avoid the above issues and then use integrated gradients to naturally conduct multiagent credit assignment.

Now we introduce how to use the path integrated gradients on trajectories to decompose \( Q_{\text{tot}} \) into approximative individual \( Q^i \). The key to path integrated gradients is to find the correct changing path of each agent’s state-action features. As we analyzed previously, such a path could be depicted by the state-action transition trajectory in an RL environment, which captures the state-action feature transformation process from the start state to the termination state. Besides, with the trajectory as the path, we can naturally use the termination state \( s_T \) as baseline where \( Q(s_T, \emptyset) = 0 \). \( \emptyset \) means no action is further taken at the termination state. After specifying both the integration path and baseline, we employ integrated gradients on the trajectory path to decompose the critic’s prediction \( Q_{\text{tot}} \) to each agent’s local observations and actions to assign the credit. Formally, using joint observations \( \tilde{o} \) to represent the global state \( s \), we have Equation 4 and the proof is provided in Theorem 1.

\[
Q_{\text{tot}}(\tilde{o}_i, \tilde{a}_i) = \sum_{i=1}^{n} Q^i(\tilde{o}_i, \tilde{a}_i),
\]

(4)

where

\[
Q^i(\tilde{o}_i, \tilde{a}_i) \approx \sum_{s_j \in \mathcal{X}_j} \text{PathIG}^T_{\text{tot}}{\tilde{o}_i, \tilde{a}_i}. \tag{5}
\]

Here \( \tau^T \) is the trajectory path from time \( t \) to \( T \), and every two adjacent joint observations and actions are connected by straight lines. \( \mathcal{X}_j \) is the set of agent \( j \)'s observation-action features. By decomposing the global Q-value following the real trajectory path \( \tau \), we get each agent’s individual contribution to \( Q_{\text{tot}} \) based on its own observation and action. Because the attribution reveals how much each agent’s own observation and action contributes to \( Q_{\text{tot}} \) by following the real trajectory path, we regard the attribution value \( \sum_{s_j \in \mathcal{X}_j} \text{PathIG}^T_{\text{tot}}{\tilde{o}_i, \tilde{a}_i} \) of agent \( j \)'s observation-action features as its approximative individual Q-value \( Q^j(\tilde{o}_i, \tilde{a}_i) \).

Next, we show how to compute \( \sum_{s_j \in \mathcal{X}_j} \text{PathIG}^T_{\text{tot}}{\tilde{o}_i, \tilde{a}_i} \). As

---

**Figure 1.** The overall QPD Framework. The top block is the centralized critic with a multi-channel modular design. The middle block is applying the Q-value path decomposition technique to achieve credit assignments on the agent level. The \( Q_{\text{tot}} \) is decomposed into the supervision signals for \( Q^j \). The bottom block shows the network architecture of the agent policies, which are implemented by the recurrent deep Q-network.
paths between every two adjacent joint observations and actions are straight lines in the path $\tau_i^T$, we can directly apply integrated gradients on the line between every two adjacent joint observations and actions from $(\tilde{a}_t, \tilde{a}_{t+1})$ to $(\tilde{a}_t, \tilde{a}_{t+1})$ as shown in Equation 6.

$$
\sum_{x_j \in \mathcal{X}_j} \text{PathIG}_{x_j}^T(\tilde{a}, \tilde{a}) = \sum_{x_j \in \mathcal{X}_j} \text{IG}_{x_j}^{\tau_{t+1}}(\tilde{a}, \tilde{a}) + \sum_{x_j \in \mathcal{X}_j} \text{IG}_{x_j}^{\tau_t}(\tilde{a}, \tilde{a}) + \ldots + \sum_{x_j \in \mathcal{X}_j} \text{IG}_{x_j}^{\tau_1}(\tilde{a}, \tilde{a}).
$$

(6)

Using integrated gradients to decompose $Q_{\text{tot}}$ makes use of the available global information. Next, in Theorem 1, we prove that decomposing global $Q_{\text{tot}}$ through the trajectory satisfies the additive property across agents, which realizes an intact decomposition. Before proof, we introduce one important property of integrated gradients in Proposition 1 (Sundararajan et al., 2017) that the attributions add up to the difference between function $F$’s outputs at the input $\tilde{x}$ and baseline $\tilde{b}$, which will be used in proving Theorem 1.

**Proposition 1.** If $F : \mathbb{R}^d \to \mathbb{R}$ is differentiable almost everywhere, then

$$
\sum_{x_j \in \mathcal{X}_j} \text{IG}_{x_j}^T(\tilde{x}) = F(\tilde{x}) - F(\tilde{b}),
$$

(7)

where $j$ is the feature index and $\tilde{x}$ gives all the features. $\tau$ represents the straight path between $\tilde{x}$ and $\tilde{b}$. Deep networks built out of Sigmoids, Relus, and pooling operators satisfy the differentiable condition. Using Equation 7 and the definition of $\text{PathIG}$ and $\text{IG}$ in Equation 2 and 3, we could decompose the $Q_{\text{tot}}$ completely to individual contributions through the trajectory path.

**Theorem 1.** Let $\tau_i^T$ represent the joint observation and action trajectory from step $1$ to the termination step $T$, then

$$
Q_{\text{tot}}(\tilde{a}_t, \tilde{a}_{t+1}) = \sum_{i=1}^{n} \sum_{x_j \in \mathcal{X}_j} \text{PathIG}_{x_j}^{\tau_i^T}(\tilde{a}, \tilde{a}).
$$

(8)

**Proof.** Let $\tilde{x}_i \in \mathbb{R}^d$ represent the feature vector $(\tilde{a}_t, \tilde{a}_{t+1})$ concisely. $\tau_i^T$ is composed of $(\tau_i^{t+1}, \tau_i^{t+2}, \ldots, \tau_i^{-1})$, where $\tau_i^{t+1}$ is the straight line path from $(\tilde{a}_t, \tilde{a}_{t+1})$ to $(\tilde{a}_{t+1}, \tilde{a}_{t+1})$.

$$
Q_{\text{tot}}(\tilde{a}_t, \tilde{a}_{t+1}) = Q_{\text{tot}}(\tilde{x}_i) = Q_{\text{tot}}(\tilde{x}_i) - Q_{\text{tot}}(\tilde{x}_i - \tilde{x}_1) + Q_{\text{tot}}(\tilde{x}_i + \tilde{x}_1),
$$

$$
= \sum_{x_j \in \mathcal{X}_j} \text{IG}_{x_j}^{\tau_{t+1}}(\tilde{x}) + \sum_{x_j \in \mathcal{X}_j} \text{IG}_{x_j}^{\tau_{t+2}}(\tilde{x}) + \ldots + \sum_{x_j \in \mathcal{X}_j} \text{IG}_{x_j}^{\tau_1}(\tilde{x})
$$

$$
= \text{PathIG}_{x_j}^{\tau_i^T}(\tilde{x}) + \text{PathIG}_{x_j}^{\tau_i^T}(\tilde{x}) + \ldots + \text{PathIG}_{x_j}^{\tau_i^T}(\tilde{x})
$$

$$
= \sum_{x_j \in \mathcal{X}_j} \text{PathIG}_{x_j}^{\tau_i^T}(\tilde{x}) + \sum_{x_j \in \mathcal{X}_j} \text{PathIG}_{x_j}^{\tau_i^T}(\tilde{x}) + \ldots + \sum_{x_j \in \mathcal{X}_j} \text{PathIG}_{x_j}^{\tau_i^T}(\tilde{x})
$$

$$
= \sum_{i=1}^{n} \sum_{x_j \in \mathcal{X}_j} \text{PathIG}_{x_j}^{\tau_i^T}(\tilde{a}, \tilde{a}).
$$

Line 4 to line 6 in the proof shows that, as we apply integrated gradients at every two adjacent joint observation-action pair along the trajectory, we aggregate each agent’s features’ attribution into the contribution of each agent for the global Q-values. Finally, we conclude that integrated gradients on the trajectory path attributes the global Q-value to each agent’s feature changes and the decomposition is intact. From the angle of the path integrated gradients, we here find the right feature varying process and then follow this trajectory path to decompose $Q_{\text{tot}}$ to individual Q-values on account of each agent’s observation and action features.

### 3.2. Multi-channel Critic

In realistic MAS, there may exist heterogeneous agents of different kinds. The space of agents’ joint states and actions is very large in such systems, causing the learning of the global Q-value to be extremely hard. Although agents in MAS are unique, they can also be categorized into different groups according to their feature attributions and personal profile. This fact enlightens us on using sub-network channels to extract information with one channel for one agent group. From bottom to top, agents can be first classified as several kinds of groups and then summarized as a unified system. Based on such a MAS abstraction, we design the multi-channel network structure as illustrated in Figure 2 to collect the hidden states from each agent’s decentralized observations and actions instead of simply using full-connected layers. At the same time, as there may exist homogeneous agents of the same kind group, we use parameter sharing for homogeneous agents. This technique is adopted widely in many complicated environments and challenging tasks (Yang et al., 2018; Iqbal & Sha, 2018) and could effectively reduce the network parameters and accelerate learning.
3.3. Algorithm and Training Process

The algorithm details are shown in Algorithm 1. Lines 2-10 show that the decentralized agents interact with the environment. Next, Lines 13-19 update the critic and target critic networks. The centralized critic $Q_{\text{init}}$ is trained to minimize the loss $\mathcal{L}(\theta^c)$ as defined in Equation 9.

\[
\mathcal{L}(\theta^c) = E_{\tilde{e},\tilde{d},r,\tilde{o}}[(Q^c(t_1, ..., o_n, a_1, ..., a_n) - y)^2],
\]

where $\theta^c$ is the critic parameters and $\tilde{\theta}^c$ is the target critic parameters, which are reset every $C$ episode. Agent $i$'s network parameters are remarked as $\theta^i$. At last, Lines 20-26 update each agent’s individual Q-value network using the decomposed $\tilde{Q}^i$ as the target label for each agent $i$. The loss of agent $i$’s Q-value network is defined as Equation 10.

\[
\mathcal{L}(\theta^i) = E_{\tilde{e},\tilde{d},r,\tilde{o}}[(Q^i(\theta^i(a_i,a_i) - \tilde{Q}^i)^2],
\]

Notably, for each training, we sample a batch of complete trajectories in the replay buffer for updating. The agent network in the realistic implement is a Recurrent Deep Q-Network (RDQN), which is the basic DQN augmented with the LSTM units. Besides, the exploration policy is $\epsilon$-greedy with $\epsilon(e)$ being the exploration rate as Equation 11.

\[
\epsilon(e) = \max(\epsilon_{\text{init}} - e \cdot \delta, 0),
\]

where $e$ is the episode number. $\epsilon_{\text{init}}$ is the start exploration rate and $\delta$ gives the decreasing amount of $\epsilon$ at each episode.

4. Experiment and Analysis

4.1. Experimental Setup

In this section, we describe the StarCraft II decentralized micromanagement problems, in which each of the learning agents controls an individual allied army unit. The enemy units are controlled by a built-in StarCraft II AI, which makes use of handcrafted heuristics. The difficulty of the game AI is set to the “very difficult” level. At the beginning of each episode, the enemy units are going to attack the allies. Proper micromanagement of units during battles are needed to maximize the damage to enemy units while minimizing damage received, hence requires a range of skills such as focusing fire and avoiding overkill. Learning these diverse cooperative behaviors under partial observation is a challenging task, which has become a common benchmark for evaluating state-of-the-art MARL approaches such as COMA (Foerster et al., 2018), QMIX (Rashid et al., 2018), and QTRAN (Son et al., 2019). We use StarCraft Multi-Agent Challenge (SMAC) environment (Samvelyan et al., 2019) as our testbed. More setup details are in the Appendix.

4.1.1. Network and Training Configurations

The architecture of agent Q-networks is a DRQN with an LSTM layer with a 64-dimensional hidden state, with a fully-connected layer after, and finally a fully-connected layer.
Q-value Path Decomposition for Deep Multiagent Reinforcement Learning

4.1.2. Decomposition Path Settings

For the Q-value decomposition process, integrated gradients can be efficiently approximated via a summation at points occurring at sufficiently small intervals along the trajectory path over each pair of consecutive state-action transitions \((\bar{a}_t, \bar{d}_t)\) and \((\bar{a}_{t+1}, \bar{d}_{t+1})\). Then the gradient integral path is obtained by repeatedly interpolating between every two adjacent states from the current state to the terminated state. With \(m\) being the number of steps in the Riemann approximation and \(\bar{x}_t\) being \((\bar{d}_t, \bar{a}_t)\) for simplification, we calculate the integrated gradients for every two adjacent states as:

\[
I\bar{G}^{(\bar{d}, \bar{a})}_{t_k} (\bar{t}_k) = I\bar{G}^{(\bar{d}, \bar{a})}_{t_k} (\bar{x}_{t_k}) := (\bar{x}_{t_k} - \bar{x}_{t_k+1}) \sum_{k=1}^{m} \frac{\partial F(\bar{x}_{t_k+1} + \frac{1}{m} \times (\bar{x}_{t_k} - \bar{x}_{t_k+1}))}{\partial (\bar{x}_{t_k+1} + \frac{1}{m} \times (\bar{x}_{t_k} - \bar{x}_{t_k+1}))} \times \frac{1}{m},
\]

(12)

Although larger \(m\) could obtain more accurate decomposition, due to the trade-off of high qualified performance and limited computation time and resources, we set \(m\) at 5 after the experimental studies. It achieves impressive performance and could be referred to in Section 4.3.2.

4.2. Results

To validate QPD, we evaluate it on both homogeneous and heterogeneous scenarios. To encourage exploration, we use \(\epsilon\)-greedy which anneals from 1 to 0 at the first 2000 episodes. We test our method at every 100 training episodes on 100 testing episodes with exploratory behaviors disabled.

The main evaluation metric is the win percentage of evaluation episodes over the course of training (Samvelyan et al., 2019). The results include the median performance as well as the 25-75% percentiles recommended in (Samvelyan et al., 2019) to avoid the effect of any outliers. Another metric, the mean win rate over all runs, is also reported. All experiments are conducted across 12 independent runs and QPD’s learning curves on all maps are shown in Figure 3.

All maps are of the different agent number or different types. Both sides in Map 3m have 3 Marines while in Map 8m have 8 Marines. In Map 2s3z, both sides have 2 Stalkers and 3 Zealots. For Map 3s5z, both sides have 3 Stalkers and 5 Zealots. Map 1c3s5z, both sides have an extra Colossus compared with Map 3s5z. In map 3s5z_vs_3s6z, ally has 3 Stalkers and 5 Zealots while enemy has 3 Stalkers and 6 Zealots. To compare QPD with existing MARL methods, we use results from SMAC (Samvelyan et al., 2019) because methods in their report show higher performance than the original works (Rashid et al., 2018; Foerster et al., 2018) and our implementation. We also compare with QTRAN.

Table 1 shows the evaluation metric results, where \(m\) is the median win percentage and \(\bar{m}\) is the mean win percentage.

We could see that QPD’s performance is competitive with QMIX in four simple scenarios, 3m, 8m, 2s3z, and 1c3s5z. More importantly, in the more difficult 3s5z where all ex-
isting methods perform poorly, QPD achieves superior performance much better than others. Furthermore, in a super hard scenario 3s5z vs 3s6z, QPD also beats other methods, where all other methods fail completely. To understand the rationale behind the results, we analyze the learned behaviors of agents. In 3m, agents learn to focus fire for beating enemies. Furthermore, in 8m, agents learn to stand into a line to shoot the enemy while avoiding overkill. In the heterogeneous 2s3z and 1c3s5z, both QMIX and QPD could solve it. Our method successfully learned to intercept the enemy Zealots with allied Zealots to protect the allied Stalkers from severe damage. However, in 3s5z, the learned policy of QPD is quite different from 2s3z: allied Zealots go around the enemy Zealots to attack the enemy Stalkers first and then attack the enemy Zealots with the allied Stalkers on both sides. Other methods fail to learn policies of a high win rate in this scenario. In 3s5z vs 3s6z, Zealots need to hold enemy’s Zealots to protect ally’s Stalkers and attack enemy’s Stalkers at the same time. Such a behavior is learned only by QPD which starts to win. Overall, QPD learns excellent decentralized policies comparable to the state-of-the-art MARL methods in both homogeneous and heterogeneous scenarios and outperforms QMIX and QTRAN in more complicated settings.

4.3. Ablation

4.3.1. Multi-channel Critic Evaluation

Using a modular network structure in the centralized critic is common in MARL algorithms and could effectively improve the performance (Iqbal & Sha, 2018; Liu et al., 2019). We also test the naive critic with several fully-connected dense layers, but we found this structure is with a high variance and its performance is lower than the modular ones. The reason is that the number of features fed into the critic is up to hundreds and increases quadratically with the number of agents, which causes a huge challenge for the naive network to learn effective hidden states from these features. Thus, we omit the naive critic’s results. One main difference with previous modular critic methods is that we explicitly consider the heterogeneous multiagent setting. We use different channels for different kinds of agents. Furthermore, we choose the concatenation operation as the way of the integration of the hidden features from each channel. We show this design could slightly improve the performance of QPD. The reason for this phenomenon is clear. The multi-channel and concatenation operation own the greater representation ability to keep track of the feature influence of each agent of each kind in the multiagent Q-value prediction process.

4.3.2. Decomposition Step

As the integrated gradients is the core of QPD, it is critical and interesting to study the decomposition step’s impact on the performance. Between each adjacent joint state-action pairs, we set the decomposition step of 1, 2, 5, 10, and 25 for studying. Results are presented in Figure 5. As we can see, the decomposition step affects the performance a lot. When the decomposition step number is low, the decomposition is not accurate enough to assign credits for agents, thus making the training unstable and win rate low. But when the decomposition step increases, the more accurate decomposed individual Q-values could update the policies more accurately. Especially, QPD is capable of the setting of moderate decomposition step number, where step of 5 could reach a comparable performance level of step 10 and 25. It means that QPD does not require lots of computation resources for decomposing to reach high performance.

5. Conclusion and Future Work

In this paper, we propose QPD to solve the multiagent credit assignment problem in Dec-POMDP settings. Different from previous methods, we propose the trajectory-based integrated gradients attribution method to achieve effective Q-value decomposition at the agent level. Experiments on the challenging StarCraft II micromanagement tasks show that QPD learns well-coordinated policies on various scenarios and reaches the state-of-the-art performance.

For the future work, better configurations of the path integrated gradients should be investigated to help attribution such as alternative choices of interpolation methods. Also, policy gradient methods combined with the path integrated gradients are expected to leverage better coordination.
ACKNOWLEDGMENTS

The work is supported by the National Natural Science Foundation of China (Grant Nos.: 61702362, U1836214, U1813204), Special Program of Artificial Intelligence and Special Program of Artificial Intelligence of Tianjin Municipal Science and Technology Commission (No.: 569 17ZXRGGX00150).

References


