Q-value Path Decomposition for Deep Multiagent Reinforcement Learning

Yaodong Yang [1] **Jianye Hao** [1 2] **Guangyong Chen** [3] **Hongyao Tang** [1] **Yingfeng Chen** [4] **Yujing Hu** [4] **Changjie Fan** [4] **Zhongyu Wei** [5]

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]College of Intelligence and Computing, Tianjin University [2]Huawei Noah's Ark Lab [3]Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China [4]NetEase Fuxi AI Lab [5]Fudan University. Correspondence to: Jianye Hao <jianye.hao@tju.edu.cn>, Guangyong Chen <gy.chen@siat.ac.cn>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

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).

For simple problems, it might be possible to manually de-

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 term 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 unsatisfied 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 critic to generate Q_{tot} by following the individual, group and system concepts progressively based on agents' joint observations and actions. Lastly, we merge the integrated gradients into MARL to decompose Q_{tot} into approximative Q^i with respect to each agent's local observation and action for precise credit assignment. We evaluate QPD using the StarCraft II micromanagement tasks. Experiments show that QPD learns effective policies in both homogeneous and heterogeneous scenarios with state-of-the-art performance.

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

$$G =< S, A, P, r, Z, O, n, \gamma > \tag{1}$$

where $s \in S$ describes the true state of the environment. Dec-POMDPs consider partially observable scenarios in which an observation function $Z(s,i) : S \times N \to p(O)$, which defines the probability distribution of the observations $o^i \in O^i$ for each agent $i \in N \equiv \{1,...,n\}$ 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 $\pi_i : O_i \times A_i \to [0,1]$. The joint action $\vec{a} \in \vec{A}$ produces the next state according to the state transition function $P : S \times A_1 \times ... \times A_n \to S$. All agents share the same reward function $r(s,\vec{a}) : S \times \vec{A} \to R$. All agents coordinate together to maximize the total expected return $J = E_{a_1 \sim \pi_1, ..., a_n \sim \pi_n, s \sim P} \sum_{t=0}^{T} \gamma^t r_t(s,\vec{a})$ where γ 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 $R_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}$. The agents' joint policy induces a value function, i.e., an approximation of expectation over R_t, $V^{\vec{\pi}}(s_t) = E_{\vec{a}_t \sim \vec{\pi}, s_{t+1} \sim P}[R_t|s_t]$, and a global action-value $Q^{\vec{\pi}}(s_t, \vec{a}_t) = E_{\vec{a}_{t+1} \sim \vec{\pi}, s_{t+1} \sim P}[R_t|s_t, \vec{a}_t]$ remarked as Q_{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* $F : \mathbb{R}^d \to \mathbb{R}$ *that represents a deep network, and an input* $\vec{x} = (x_1, ..., x_j, ..., x_d) \in \mathbb{R}^d$. $\mathbb{R}$ *is the set of real numbers. F is the function with a d-dimension vector input. An attribution of the prediction at input* $\vec{x}$ *relative to a baseline input* $\vec{b}$ *is a vector* $A_F(\vec{x}, \vec{b}) = (c_{x_1}, ...c_{x_j}, ..., c_{x_d}) \in \mathbb{R}^d$*, where* c_{x_j} *is the contribution value of* x_j *to the difference between prediction* $F(\vec{x})$ *and the baseline prediction* $F(\vec{b})$.

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 (Tarashev et al., 2016) with theoretical supports (Hazewinkel, 1990). The integrated gradients explains how much one feature affects the deep network output while changing from $F(\vec{b})$ to $F(\vec{x})$ along a straight line between $\vec{x}$ and $\vec{b}$. 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 $\tau(\alpha) : [0,1] \to R^d$ be a smooth path function specifying a path in R^d from the baseline $\vec{b}$ to the input $\vec{x}$, i.e., $\tau(0) = \vec{b}$ and $\tau(1) = \vec{x}$. Given a path function τ, path integrated gradients are obtained by integrating gradients along the path $\tau(\alpha)$ for $\alpha \in [0,1]$. Mathematically, path integrated gradients along the jth dimension for input $\vec{x}$ (i.e., x_j) on the path τ is defined as follows.

$$c_{x_j} = PathIG^{\tau}_{x_j}(\vec{x}) ::= \int_{\alpha=0}^{1} \frac{\partial F(\tau(\alpha))}{\partial \tau_{x_j}(\alpha)} \frac{\partial \tau_{x_j}(\alpha)}{\partial \alpha} \mathrm{d}\alpha, \quad (2)$$

where $\frac{\partial F(\tau(\alpha))}{\partial \tau_{x_j}(\alpha)}$ is the gradient of F along the jth 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 $\tau(\alpha) = \vec{b} + \alpha(\vec{x} - \vec{b})$ for $\alpha \in [0,1]$, the integrated gradients (Sundararajan et al., 2017) to calculate the contribution value c_{x_j} along the jth dimension for input $\vec{x}$ is defined as follows.

$$c_{x_j} = IG^{\tau}_{x_j}(\vec{x}) ::= (\vec{x}_j - \vec{b}_j) \int_{\alpha=0}^{1} \frac{\partial F(\tau(\alpha))}{\partial \tau_{x_j}(\alpha)} \mathrm{d}\alpha. \quad (3)$$

In the computer vision and natural language processing domains, when applying integrated gradients, the zero embedding vector is usually used as the baseline $\vec{b}$. 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_{tot} into individual Q-values Q^i approximately for each agent in Section 3.1. Such a decomposition process addresses the multiagent credit assignment via the covariation analysis of each agent's observations and actions along the trajectory path. The decomposed individual value which approximates Q^i 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_{tot} from agents' joint observations and actions. Finally, we give the algorithm details and training losses in Section 3.3.

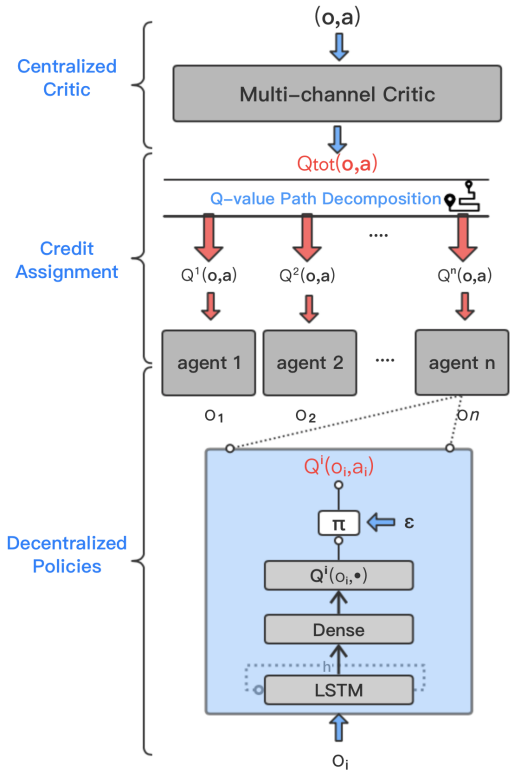


Figure 1. The overall QPD Framework. The top block is the centralized critic with a multi-channel modular design. The middle block is applying the Q-value path decomposition technique to achieve credit assignments on the agent level. The Q_{tot} is decomposed into the supervision signals for Q^i. The bottom block shows the network architecture of the agent policies, which are implemented by the recurrent deep Q-network.

3.1. Value Decomposition Through Integrated Gradients

In this section, we apply integrated gradients to assign credits for each agent on the multi-channel critic by performing attribution on each own states and actions with respect to the output Q_{tot}. As Deep Reinforcement Learning (DRL) employs deep neural networks to approximate the global Q_{tot}, we could utilize the attribution tools in DL combined with concepts in RL to extract the contribution of specified sets of features from different agents to the predicted Q-value. To this end, in this paper, we propose a new multiagent credit assignment approach that utilizes integrated gradients on the state-action trajectory.

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_{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 $Q(s_T, \varnothing) = 0$. $\varnothing$ 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_{tot} to each agent's local observations and actions to assign the credit. Formally, using joint observations $\vec{o}$ to represent the global state s, we have Equation 4 and the proof is provided in Theorem 1.

$$Q_{tot}(\vec{o}_t, \vec{a}_t) = \sum_{i=1}^{n} Q^i(\vec{o}_t, \vec{a}_t), \tag{4}$$

where

$$Q^i(\vec{o}_t, \vec{a}_t) \approx \sum_{x_j \in \mathbb{X}_i} PathIG^{\tau_t^T}_{x_j}(\vec{o}_t, \vec{a}_t). \tag{5}$$

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

Next, we show how to compute $\sum_{x_j \in \mathbb{X}_i} PathIG^{\tau_t^T}_{x_j}(\vec{o}_t, \vec{a}_t)$. As

paths between every two adjacent joint observations and actions are straight lines in the path τ_t^T, we can directly apply integrated gradients on the line between every two adjacent joint observations and actions from $(\vec{o}_{t+1}, \vec{a}_{t+1})$ to $(\vec{o}_t, \vec{a}_t)$ as shown in Equation 6.

$$\sum_{x_j \in \mathbb{X}_i} PathIG_{x_j}^{\tau_t^T}(\vec{o}_t, \vec{a}_t) = \sum_{x_j \in \mathbb{X}_i} IG_{x_j}^{\tau_t^{t+1}}(\vec{o}, \vec{a}) + \sum_{x_j \in \mathbb{X}_i} IG_{x_j}^{\tau_{t+1}^{t+2}}(\vec{o}, \vec{a}) + \ldots + \sum_{x_j \in \mathbb{X}_i} IG_{x_j}^{\tau_{T-1}^{T}}(\vec{o}, \vec{a}). \tag{6}$$

Using integrated gradients to decompose Q_{tot} makes use of the available global information. Next, in Theorem 1, we prove that decomposing global Q_{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 $\vec{x}$ and baseline $\vec{b}$, which will be used in proving Theorem 1.

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

$$\sum_{x_j \in \vec{x}} IG_{x_j}^{\tau}(\vec{x}) = F(\vec{x}) - F(\vec{b}), \tag{7}$$

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

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

$$Q_{tot}(\vec{o}_t, \vec{a}_t) = \sum_{i=1}^{n} \sum_{x_j \in \mathbb{X}_i} PathIG_{x_j}^{\tau_t^T}(\vec{o}, \vec{a}). \tag{8}$$

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

$$\begin{aligned}
&Q_{tot}(\vec{o}_t, \vec{a}_t) = Q_{tot}(\vec{x}_t) = Q_{tot}(\vec{x}_t) - Q_{tot}(\vec{x}_T) = Q_{tot}(\vec{x}_t) - Q_{tot}(\vec{x}_{t+1}) \\
&+ Q_{tot}(\vec{x}_{t+1}) - Q_{tot}(\vec{x}_{t+2}) + \ldots + Q_{tot}(\vec{x}_{T-1}) - Q_{tot}(\vec{x}_T) \\
&= \sum_{x_j \in \vec{x}} IG_{x_j}^{\tau_t^{t+1}}(\vec{x}) + \sum_{x_j \in \vec{x}} IG_{x_j}^{\tau_{t+1}^{t+2}}(\vec{x}) + \ldots + \sum_{x_j \in \vec{x}} IG_{x_j}^{\tau_{T-1}^{T}}(\vec{x}) \\
&= PathIG_{x_1}^{\tau_t^T}(\vec{x}) + PathIG_{x_2}^{\tau_t^T}(\vec{x}) + \ldots + PathIG_{x_d}^{\tau_t^T}(\vec{x}) \\
&= \sum_{x_j \in \mathbb{X}_1} PathIG_{x_j}^{\tau_t^T}(\vec{x}) + \sum_{x_j \in \mathbb{X}_2} PathIG_{x_j}^{\tau_t^T}(\vec{x}) + \ldots + \sum_{x_j \in \mathbb{X}_n} PathIG_{x_j}^{\tau_t^T}(\vec{x}) \\
&= \sum_{i=1}^{n} \sum_{x_j \in \mathbb{X}_i} PathIG_{x_j}^{\tau_t^T}(\vec{x}) = \sum_{i=1}^{n} \sum_{x_j \in \mathbb{X}_i} PathIG_{x_j}^{\tau_t^T}(\vec{o}, \vec{a})
\end{aligned}$$

□

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_{tot} to individual Q-values on account of each agent's observation and action features.

3.2. Multi-channel Critic

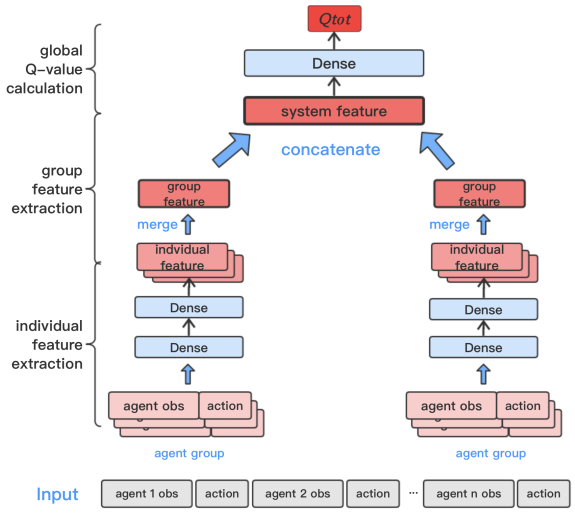


Figure 2. Multi-channel Critic.

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.

Algorithm 1 Q-value Path Decomposition algorithm

Initialize: Critic network θ^c, target critic $\widetilde{\theta}^c$ and agents' Q-value networks $\theta^\pi = (\theta^1, ..., \theta^n)$

1: **for** each training episode e **do**
2: s_0 = initial state, $t = 0$, $h_0^i = 0$ for each agent i.
3: **while** $s_t \neq$ terminal **and** $t < T$ **do**
4: $t = t + 1$.
5: **for** each agent i **do**
6: $Q^i(o_{t,i}, \cdot), h_t^i$ =DRQN$(o_{t,i}, h_{t-1}^i; \theta^i)$.
7: Sample $a_{t,i}$ from $\pi_i(Q^i(o_{t,i}, \cdot), \varepsilon(e))$.
8: **end for**
9: execute the joint actions $(a_{t,1}, a_{t,2}, ..., a_{t,n})$.
10: receive the reward r_t and next state s_{t+1}.
11: **end while**
12: Add episode to buffer and sample a batch of episodes.
13: **for** e in batch **do**
14: **for** $t = 1$ to T **do**
15: Calculate targets y_t using $\widetilde{\theta}^c$.
16: **end for**
17: **end for**
18: Update critic parameters θ^c with loss $\mathscr{L}(\theta^c)$.
19: Every C episodes reset $\widetilde{\theta}^c = \theta^c$.
20: **for** e in batch **do**
21: **for** $t = 1$ to T **do**
22: Unroll LSTM using states, actions and rewards.
23: Using the Integrated Gradients along with the trajectory e to decompose Q_{tot} at time t into $\widetilde{Q}_t^i = \sum_{x_j \in \mathbb{X}_i} PathIG_{x_j}^{\tau}(\vec{o}_t, \vec{a}_t)$ for each agent i.
24: **end for**
25: **end for**
26: Update θ^π with loss $\mathscr{L}(\theta^i)$ for each agent i.
27: **end for**

The critic structure includes three components: the individual feature extraction process, the group feature extraction process, and the system's global Q-value calculation process. We first use the individual feature extracting modules to extract embeddings for agents with one channel responding to one agent group. Next, the group feature merging operation combines the embeddings from the same group and then concatenates these group embeddings into the system features. The merging operation could be either concatenation or addition. Finally, the high-level system features are used to calculate the system's Q-values. Such a modular critic structure provides a succinct representation of the multiagent Q-value while the number of network parameters can be significantly reduced as well.

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_{tot} is trained to minimize the loss $\mathscr{L}(\theta^c)$ as defined in Equation 9.

$$\begin{aligned} \mathscr{L}(\theta^c) &= E_{\vec{o},\vec{a},r,\vec{o}'}[(Q_{tot}^{\theta^c}(o_1, ..., o_n, a_1, ..., a_n) - y)^2], \\ y &= r + \gamma(Q_{tot}^{\widetilde{\theta}^c}(o_1', ..., o_n', a_1', ..., a_n'), \end{aligned} \tag{9}$$

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

$$\begin{aligned} \mathscr{L}(\theta^i) &= E_{\vec{o},\vec{a},r,\vec{o}'}[(Q^{i,\theta^i}(o_i, a_i) - \widetilde{Q}^i)^2], \\ \widetilde{Q}^i &= \sum_{x_j \in \mathbb{X}_i} PathIG_{x_j}^{\tau}(\vec{o}, \vec{a}). \end{aligned} \tag{10}$$

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 ε-greedy with $\varepsilon(e)$ being the exploration rate as Equation 11.

$$\varepsilon(e) = \max(\varepsilon_{init} - e * \delta, 0), \tag{11}$$

where e is the episode number. ε_{init} is the start exploration rate and δ gives the decreasing amount of ε 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

with $|A|$ outputs. The input for agent networks is the sequential data which consists of the agent's local observations in latest 12 time steps for all scenarios. The architecture of the QPD critic is a feedforward neural network with the first two dense layers having 64 units for each channel, and then being concatenated or added in each group, and next being concatenated to the output layer of one unit. We set γ at 0.99. To speed up learning, we share the parameters across all individual Q-networks and a one-hot encoding of the agent type is concatenated onto each agent's observations to allow the learning of diverse behaviors. All agent networks are trained using RMSprop with a learning rate of 5×10^{-4} and the critic is trained with Adam with the same learning rate. Replay buffer contains the most recent 1000 trajectories and the batch size is 32. Target networks for the global critic are updated after every 200 training episodes.

4.1.2. Decomposition Path Settings

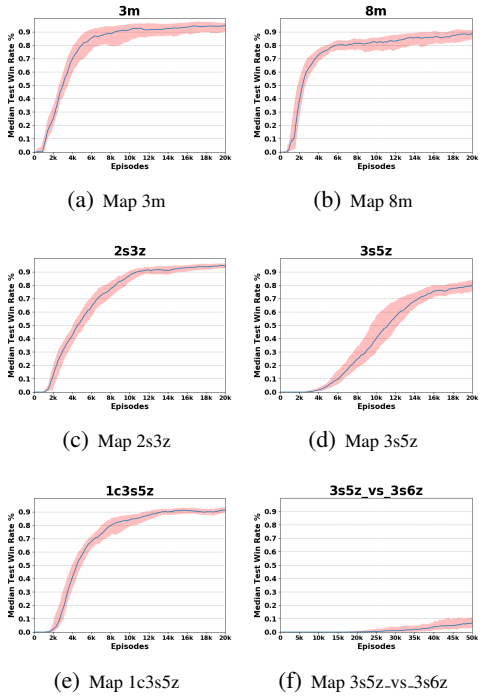
For the Q-value decomposition process, integrated gradients can be efficiently approximated via a summation at points occurring at sufficiently small intervals along the trajectory path over each pair of consecutive state-action transitions $(\vec{o}_t, \vec{a}_t)$ and $(\vec{o}_{t+1}, \vec{a}_{t+1})$. 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 Riemman approximation and $\vec{x}_t$ being $(\vec{o}_t, \vec{a}_t)$ for simplification, we calculate the integrated gradients for every two adjacent states as:

$$\widetilde{IG}_{x_j}^{\tau_l^{t+1}}(\vec{o}_t, \vec{a}_t) = \widetilde{IG}_{x_j}^{\tau_l^{t+1}}(\vec{x}_t) ::= (\vec{x}_{t,j} - \vec{x}_{t+1,j}) \times \sum_{k=1}^{m} \frac{\partial F(\vec{x}_{t+1} + \frac{k}{m} \times (\vec{x}_t - \vec{x}_{t+1}))}{\partial (\vec{x}_{t+1} + \frac{k}{m} \times (\vec{x}_t - \vec{x}_{t+1}))} \times \frac{1}{m}. \tag{12}$$

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 ε-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.



(a) Map 3m (b) Map 8m

(c) Map 2s3z (d) Map 3s5z

(e) Map 1c3s5z (f) Map 3s5z_vs_3s6z

Figure 3. QPD's median win percentage of different map scenarios. 25%-75% percentile is shaded.

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 $\widetilde{m}$ is the median win percentage and $\overline{m}$ is the mean win percentage.

Table 1. Median and mean performance of the test win percentage.

Map	IQL		COMA		QMIX		QTRAN		QPD	
	$\widetilde{m}$	$\overline{m}$	$\widetilde{m}$	$\overline{m}$	$\widetilde{m}$	$\overline{m}$	$\widetilde{m}$	$\overline{m}$	$\widetilde{m}$	$\overline{m}$
3m	**100**	97	91	92	**100**	99	**100**	**100**	95	92
8m	91	90	95	94	**100**	96	**100**	**97**	94	93
2s3z	39	42	66	64	**100**	**97**	77	80	95	94
3s5z	0	3	0	0	16	25	0	4	**85**	**81**
1c3s5z	7	8	30	30	89	89	31	33	**92**	**92**
3s5z_vs_3s6z	0	0	0	0	0	0	0	0	**8**	**10**

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 2s3s: 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.

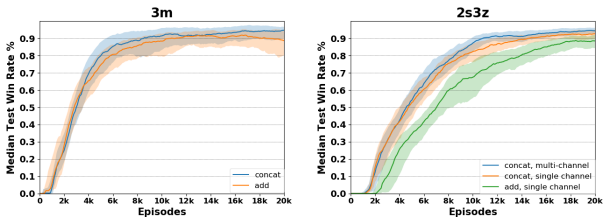
4.3. Ablation

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.

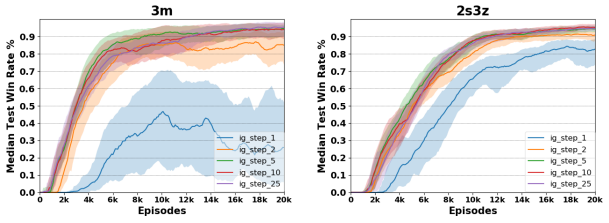
4.3.2. Decomposition Step

As the integrated gradients is the core of QPD, it is critical and interesting to study the decomposition step's impact on



(a) Map 3m (b) Map 2s3z

Figure 4. Median win percentage of 12 runs for critic ablation.



(a) Map 3m (b) Map 2s3z

Figure 5. Median win percentage of 12 runs for decomposing steps.

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

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. Towards better understanding of gradient-based attribution methods for deep neural networks. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., and Hochreiter, S. RUDDER: Return decomposition for delayed rewards. *arXiv preprint*, abs/1806.07857, 2018. URL http://arxiv.org/abs/1806.07857.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., and Müller, K.-R. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11:1803–1831, 2010.

Binder, A., Montavon, G., Lapuschkin, S., Müller, K.-R., and Samek, W. Layer-wise relevance propagation for neural networks with local renormalization layers. In *Proceedings of the 25th International Conference on Artificial Neural Networks*, pp. 63–71, 2016.

Brasó Andilla, G. Attribution methods for deep convolutional networks. 2018.

Busoniu, L., Babuska, R., and De Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(2): 156–172, 2008.

Cao, Y., Yu, W., Ren, W., and Chen, G. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, 9(1):427–438, 2012.

Chang, Y.-H., Ho, T., and Kaelbling, L. P. All learning is local: Multi-agent learning in global reward games. In *Proceedings of the 17th Advances in Neural Information Processing Systems*, pp. 807–814, 2004.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

Gupta, J. K., Egorov, M., and Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In *Proceedings of the 16th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 66–83, 2017.

Hazewinkel, M. (ed.). *Encyclopaedia of Mathematics*, volume Integral over trajectories. Springer Netherlands, 1990.

Iqbal, S. and Sha, F. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *arXiv preprint*, volume abs/1810.02912, 2018. URL http://arxiv.org/abs/1810.02912.

Liu, I.-J., Yeh, R. A., and Schwing, A. G. PIC: Permutation Invariant Critic for Multi-Agent Deep Reinforcement Learning. In *Proceedings of the 3rd Conference on Robot Learning*, 2019.

Lowe, R., WU, Y., Tamar, A., Harb, J., Pieter Abbeel, O., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31th Advances in Neural Information Processing Systems*, pp. 6379–6390, 2017.

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. MAVEN: Multi-Agent Variational Exploration. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d., Fox, E., and Garnett, R. (eds.), *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 7611–7622. Curran Associates, Inc., 2019.

Montavon, G., Samek, W., and Müller, K.-R. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

Nguyen, D. T., Kumar, A., and Lau, H. C. Policy Gradient With Value Function Approximation For Collective Multiagent Planning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Proceedings of the 30th Advances in Neural Information Processing Systems*, pp. 4319–4329. Curran Associates, Inc., 2017.

Nguyen, D. T., Kumar, A., and Lau, H. C. Credit Assignment For Collective Multiagent RL With Global Rewards. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Proceedings of the 31th Advances in Neural Information Processing Systems*, pp. 8113–8124. Curran Associates, Inc., 2018.

Oliehoek, F. A. and Amato, C. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer International Publishing, 2016.

Palmer, G., Tuyls, K., Bloembergen, D., and Savani, R. Lenient multi-agent deep reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 443–451, 2018.

Rashid, T., Samvelyan, M., Witt, C. S. d., Farquhar, G., Foerster, J. N., and Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4292–4301, 2018.

Samvelyan, M., Rashid, T., Schroeder de Witt, C., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C.-M., Torr, P. H. S., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2186–2188, Richland, SC, 2019.

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5887–5896, Long Beach, California, USA, 2019. PMLR.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 3319–3328, 2017.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2085–2087, 2018.

Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *In Proceedings of the 10th International Conference on Machine Learning*, pp. 330–337. Morgan Kaufmann, 1993.

Tarashev, N., Tsatsaronis, K., and Borio, C. Risk attribution using the shapley value: Methodology and policy applications. *Review of Finance*, 20(3):1189–1213, 2016.

Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and Wang, J. Mean field multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 5567–5576, 2018.

Ying, W. and Sang, D. Multi-agent framework for third party logistics in e-commerce. *Expert Systems with Applications*, 29(2):431–436, 2005.