Asynchronous Credit Assignment for Multi-Agent Reinforcement Learning

Yongheng Liang[1,2], **Hejun Wu**[1,2*], **Haitao Wang**[1,2] and **Hao Cai**[3]

[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, Guangdong, China
[3]College of Mathematics and Computer Science, Shantou University, Shantou, China
{liangyh38, wanght39}@mail2.sysu.edu.cn, wuhejun@mail.sysu.edu.cn, haocai@stu.edu.cn

Abstract

Credit assignment is a critical problem in multi-agent reinforcement learning (MARL), aiming to identify agents' marginal contributions for optimizing cooperative policies. Current credit assignment methods typically assume synchronous decision-making among agents. However, many real-world scenarios require agents to act asynchronously without waiting for others. This asynchrony introduces conditional dependencies between actions, which pose great challenges to current methods. To address this issue, we propose an asynchronous credit assignment framework, incorporating a Virtual Synchrony Proxy (VSP) mechanism and a Multiplicative Value Decomposition (MVD) algorithm. VSP enables physically asynchronous actions to be virtually synchronized during credit assignment. We theoretically prove that VSP preserves both task equilibrium and algorithm convergence. Furthermore, MVD leverages multiplicative interactions to effectively model dependencies among asynchronous actions, offering theoretical advantages in handling asynchronous tasks. Extensive experiments show that our framework consistently outperforms state-of-the-art MARL methods on challenging tasks while providing improved interpretability for asynchronous cooperation.

1 Introduction

Multi-agent reinforcement learning (MARL) is promising for many cooperative tasks, such as video games [Arulkumaran *et al.*, 2019] and collaborative control [Zhou *et al.*, 2023]. MARL typically assumes a synchronous decision-making setting, where all agents make decisions simultaneously and their joint actions have the same duration. This assumption simplifies the overall learning process [Oliehoek *et al.*, 2016].

Despite the success of MARL in synchronous settings, real-world tasks often exhibit asynchrony, i.e., agents cannot complete their atomic actions simultaneously, because of hardware constraints or the nature of agents' actions [Wang and Sun, 2021; Liang *et al.*, 2023], as shown in Figure 1a.

*Corresponding author

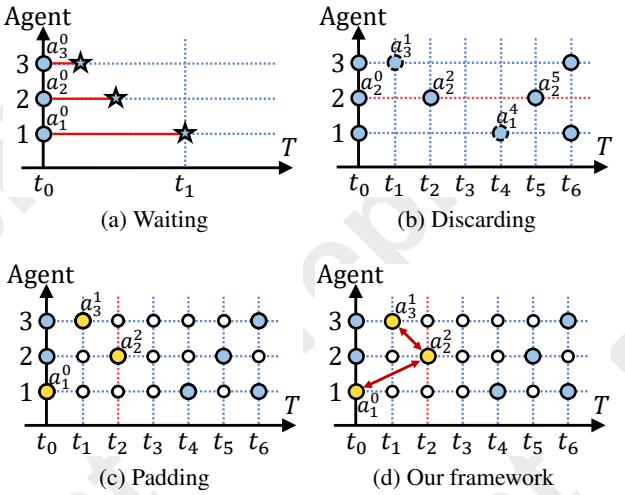


Figure 1: Illustration of various asynchronous MARL frameworks. Blue/Yellow circles denote agent $\#x$'s action a_x^y at time step t_y. Small white circles denote padding actions. Stars denote action completion. (a) Agents $\#2$ and $\#3$ must wait for agent $\#1$ to finish at t_1 before making next decisions. (b) Agent $\#2$ disregards action a_1^4 from t_2 to t_5. (c) Credit at t_2 is attributed to the padding actions. (d) Our proposed framework captures the interactions among asynchronous decisions being executed at t_2.

To address this issue, researchers have proposed two primary mechanisms as follows. *Discarding*: Agents with varying time step lengths collect data and update policies only when they make decisions [Xiao *et al.*, 2022], as shown in Figure 1b. *Padding*: Define a time step as the smallest indivisible time unit and use padding actions to transform asynchronous tasks into synchronous ones, so as to apply existing MARL [Chen *et al.*, 2021], as shown in Figure 1c.

Nevertheless, both discarding and padding struggle to address complex asynchronous cooperative tasks. Their failure is due to an inability to resolve the credit assignment problem, caused by the following two limitations. (1) **Bias introduced in evaluating the global impacts of asynchronous actions**. On one hand, the discarded information leads to biased evaluations of the impacts of decisions from other agents. As shown in Figure 1b, agent $\#2$ discards the information of action a_3^1 and a_1^4, neglecting their impacts on agent $\#2$'s tran-

sition from t_2 to t_5. On the other hand, the padding actions introduce bias in credit assignment. As shown in Figure 1c, the algorithm assigns credit at t_2 mainly to the two padding actions, rather than actions a_1^0 and a_3^1. (2) **Inability to model conditional dependencies between asynchronous actions**. In MARL, value decomposition (VD) and its variants [Sunehag *et al.*, 2018; Rashid *et al.*, 2020] are widely used synchronous credit assignment methods. They learn the marginal contribution of each agent and decompose the global Q-value Q_{tot} into individual agent-wise utilities Q_i to guide agents' behaviors. However, most VD algorithms have limitations in accounting for higher-order interactions [Liu *et al.*, 2023]. They fail to capture the conditional dependencies between decisions made by agents asynchronously, such as the dependency of the current decision a_2^2 on the actions being executed, a_1^0 and a_3^1, as shown in Figure 1c.

In this paper, we propose an asynchronous credit assignment framework that incorporates a Virtual Synchrony Proxy (VSP) mechanism and a Multiplicative Value Decomposition (MVD) algorithm. Inspired by virtual synchrony in distributed systems [Dolev *et al.*, 2018], our VSP introduces virtual proxies to migrate asynchronous actions to a unified time step. This allows actions a_1^0, a_2^2, and a_3^1 in Figure 1d to appear synchronized at t_2, facilitating better capturing their global impacts. We have proven that VSP preserves both the task equilibrium and the algorithm convergence. Based on VSP, we derive the multiplicative value decomposition formula as well as its higher-order forms and propose MVD. Our MVD leverages multiplicative interactions [Rumelhart and McClelland, 1987; Jayakumar *et al.*, 2020] to capture dependencies among asynchronous actions and we demonstrate its superior representational capacity compared to traditional VD algorithms. Moreover, we present three practical implementations of MVD. We evaluate MVD on a modified asynchronous variant of the classic MARL benchmark SMAC [Samvelyan *et al.*, 2019], along with two prominent asynchronous benchmarks: Overcooked [Wang *et al.*, 2020b] and POAC [Yao *et al.*, 2021]. Extensive experimental results show that MVD achieves considerable performance improvements in complex scenarios and provides easy-to-understand interaction processes among asynchronous decisions.

Our contributions are summarized as follows: (1) We propose an asynchronous credit assignment framework with VSP and MVD, capable of capturing high-order dependencies among asynchronous actions. (2) Theoretically, we prove the correctness of VSP and demonstrate the advantages of MVD. (3) Experimentally, we show MVD's effectiveness across three asynchronous tasks, achieving significant performance gains and interpretable interaction processes.

2 Related Works

Despite the significant progress in MARL, most existing works rely on the premise of synchronous decision-making which does not reflect reality in many practical applications. The easiest way to adapt MARL from synchronous to asynchronous decision-making is to split actions into sub-actions or wait for others to finish before making the next decisions. Evidently, these methods raise training costs and lower efficiency. Thus, several works have been conducted to exploit the strengths of MARL in asynchronous settings.

The discarding type methods recognize asynchronous actions with varying durations as a whole and focus solely on the decision information. ASM-PPO [Liang *et al.*, 2022] and ASM-HPPO [Liang *et al.*, 2023] propose that each agent collects its own decision information and utilize MAPPO [Yu *et al.*, 2022] for training. MAC IAICC [Xiao *et al.*, 2022] treats asynchronous actions as macro-actions [Theocharous and Kaelbling, 2003] and models the task as a MacDec-POMDP [Amato *et al.*, 2019]. CAAC [Wang and Sun, 2021] focuses on the bus holding control [Daganzo and Ouyang, 2019] and utilizes a graph attention network to capture the influence of agents' asynchronous decisions.

The padding type methods transform asynchronous problems into synchronous ones through padding action, thereby obtaining Dec-POMDP [Oliehoek *et al.*, 2016] and applying existing MARL methods. Since Dec-POMDP requires the collection of decision information from all agents at each time step, the padding action can be used as a substitute for the decision information of agents that are executing actions. VarLenMARL [Chen *et al.*, 2021] employs the most recent action for padding during the collection of joint transitions. EXP-Ms [Jia *et al.*, 2020] considers ongoing actions as idle, treating them as blank actions.

However, there still remains a lack of theoretical and visual analysis on asynchronous credit assignments, hindering the resolution of complex asynchronous cooperative tasks.

3 Preliminaries

3.1 Dec-POMDP

A fully cooperative multi-agent task with synchronous decision-making is typically formulated as a Dec-POMDP. Dec-POMDP is defined as a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{P}, r, O, \Omega, \gamma \rangle$, where $\mathcal{N}$ is a set of n agents and $s \in \mathcal{S}$ is a global state of the environment. At each time step, each agent $i \in \mathcal{N}$ obtains its own observation $o_i \in \Omega$ determined by the partial observation $O(s, i)$ and selects an action $a_i \in \mathcal{A}$ to form a joint action $\boldsymbol{a} = [a_i]_{i=1}^n \in \mathcal{A}^n$. Subsequently, all agents simultaneously complete their actions, leading to the next state s' through the transition function $\mathcal{P}(s'|s, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A}^n \to \mathcal{S}$ and to the global reward $r(s, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A}^n \to \mathbb{R}$. Each agent i has its own policy $\pi_i(a_i|\tau_i) : \mathcal{T} \times \mathcal{A} \to [0, 1]$ based on local action-observation history $\tau_i \in \mathcal{T}$. The objective of all agents is to find an optimal joint policy $\boldsymbol{\pi} = [\pi_i]_{i=1}^n$ and maximize the global value function $Q^{\boldsymbol{\pi}} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r^{t+1}]$ with a discount factor $\gamma \in [0, 1)$.

3.2 Credit Assignment and Value Decomposition

Credit assignment is a key challenge in designing reliable MARL methods [Oroojlooy and Hajinezhad, 2023]. It focuses on attributing team success to individual agents based on their respective marginal contributions, aiming at collective policy optimization. VD algorithms are the most popular branches in MARL. They leverage global information to learn agents' contributions and decompose the global Q-value function $Q_{tot}(s, \boldsymbol{a})$ into individual utility functions $Q_i(\tau_i, a_i)$. In the execution phase, agents cooperate via their

corresponding $Q_i(\tau_i, a_i)$, thereby realizing centralized training and decentralized execution (CTDE) [Oliehoek *et al.*, 2008]. Traditional VD algorithms, including VDN [Sunehag *et al.*, 2018], QMIX [Rashid *et al.*, 2020], and Qatten [Yang *et al.*, 2020b], can be represented by the following general additive interaction formulation [Li *et al.*, 2022]:

$$Q_{tot} = Q_{tot}(s, Q_1, Q_2, \cdots, Q_n) = k_0 + \sum_{i=1}^{n} k_i Q_i, \quad (1)$$

where k_0 is a constant and k_i denotes the credit that reflects the contributions of Q_i to Q_{tot}.

To capture high-order interactions that traditional VD algorithms ignored, NA^{2}Q [Liu *et al.*, 2023] employed a generalized additive model (GAM) [Hastie, 2017]:

$$\begin{aligned} Q_{tot} = f_0 + \sum_{i=1}^{n} k_i f_i^1(Q_i) + \cdots + \sum_{j \in \mathcal{D}_l} k_j f_j^l(Q_j) \\ + \cdots + k_{1\cdots n} f_{1\cdots n}^n(Q_1, \cdots, Q_n), \end{aligned} \quad (2)$$

where f_0 is a constant, f_j^l captures l-order interactions among agents j in $\mathcal{D}_l$. $\mathcal{D}_l$ is the set of all size-l subsets of $\{1, \cdots, n\}$, $1 \le l \le n$. The utility Q_j of agent j is the input of f_j^l.

In order to maintain the consistency between local and global optimal actions after decomposition, these VD algorithms should satisfy the following individual-global-max (IGM) principle [Son *et al.*, 2019]:

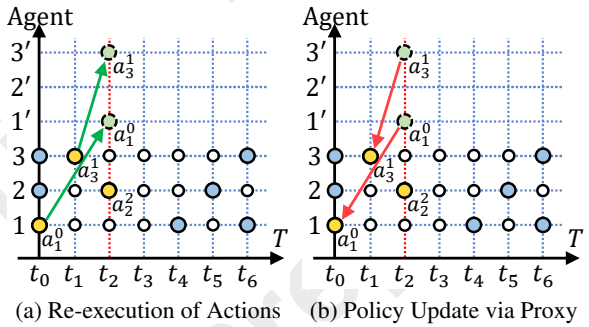
$$\arg\max_{\boldsymbol{a}} Q_{tot}(s, \boldsymbol{a}) = \begin{pmatrix} \arg\max_{a_1} Q_1(\tau_1, a_1) \\ \vdots \\ \arg\max_{a_n} Q_n(\tau_n, a_n) \end{pmatrix}. \quad (3)$$

For example, QMIX holds the monotonicity $\frac{\partial Q_{tot}}{\partial Q_i} \ge 0$ and achieves IGM between Q_{tot} and Q_i. More details of VD and other credit assignment methods are in Appendix A.

4 Virtual Synchrony Proxy

In asynchronous scenarios, system dynamics depend on actions taken at different times, yet existing methods fail to capture the global impacts of these asynchronous actions. The discarding mechanism overlooks the contributions of other agents' decisions. One padding approach uses the most recent action to synchronize decisions. For example, in Figure 1c, agent #1 selects a_1^0 as its padding actions at t_2. However, this introduces ambiguity since it cannot distinguish whether a_1^0 is being continued or restarted at t_2. Another padding approach uses blank actions to distinguish between decision-making and action execution, thereby avoiding ambiguity. Nevertheless, as shown in Figure 1c, the credit for agent #1 at t_2 is incorrectly assigned to the blank action rather than to a_1^0.

Our VSP is inspired by virtual synchrony, which offers an abstraction layer that ensures asynchronous messages are processed synchronously in distributed systems. Similarly, VSP employs virtual proxies to align asynchronous actions, allowing credit to be assigned at a unified time step. The general idea is as follows. To avoid ambiguity, VSP uses a **blank** action as padding when agent i is continuing action a_i. Meanwhile, to ensure semantic consistency, VSP introduces



(a) Re-execution of Actions (b) Policy Update via Proxy

Figure 2: At t_2, a_1^0 and a_3^1 are executing. (a) Virtual proxies $\#1'$ and $\#3'$ are introduced to re-execute a_1^0 and a_3^1. (b) The policies of agent $\#1$ and $\#3$ for a_1^0 and a_3^1 are updated through their proxies.

a virtual proxy i' at each time step when agent i is executing action a_i. The proxy i' shares the **same** policy as agent i and re-executes action a_i. Therefore, when the policies of the proxies $\#1'$ and $\#3'$ are updated with assigned credit at t_2, the policies of the corresponding agents $\#1$ and $\#3$ are updated as well, as shown in Figure 2.

We integrate the VSP mechanism into Dec-POMDP to model asynchronous decision-making problems. Before presenting the detailed formulation, the frequently used definitions are outlined below. Asynchronous action refers to an action that requires multiple time steps to complete, e.g., action a_1^0 spans over four time steps in Figure 2. The observation for making a decision is denoted as $\tilde{o}_i$, referring to the most recent observation upon which an agent i makes the decision to execute a new asynchronous action. $\tilde{a}_i$ denotes the most recent new decision made by the agent i.

Definition 1 (Dec-POMDP with VSP)**.** *Dec-POMDP with VSP is a tuple* $\langle \widehat{\mathcal{N}}, \mathcal{I}, \widehat{\mathcal{S}}, \mathcal{A}, \widehat{\mathcal{P}}, \widehat{r}, \widehat{O}, \mathcal{O}_{\mathcal{P}}, \Omega, \gamma \rangle$*, where* $\widehat{\mathcal{N}} = \{\mathcal{N}, \mathcal{N}'\}$*, with* $\mathcal{N}$ *as the real agent set and* $\mathcal{N}'$ *as the virtual proxy set. At each time step* t*, the agents are divided into two disjoint sets:* $i_d \in \mathcal{N}_d$*, the agent making the decision, and* $i_c \in \mathcal{N}_c$*, the agent executing the action. Time step index* t *is omitted for simplicity. Given an original state* $s \in \mathcal{S}$*, function* $\mathcal{I}(s)$ *returns a subset* $\mathcal{N}_c' \subseteq \mathcal{N}'$*, where proxy* $i_c' \in \mathcal{N}_c'$ *and agent* $i_c \in \mathcal{N}_c$ *form an real-virtual pair* $\langle i_c, i_c' \rangle$*. The agent* i_c *and the proxy* i_c' *in this pair have the same policy.* $\widehat{s} = [s; \tilde{\boldsymbol{o}}_c] \in \widehat{\mathcal{S}}$*, where* $[\cdot;\cdot]$ *is the concatenation operation.* s *is the original state and* $\tilde{\boldsymbol{o}}_c$ *is obtained according to* $\mathcal{O}_{\mathcal{P}}(\tilde{\boldsymbol{o}}_c|s)$*, which is the joint observation of all agents* i_c *for making decisions. Each proxy* i_c' *receives* $o_{i_c'} = \tilde{o}_{i_c} \in \Omega$ *according to* $\widehat{O}(\widehat{s}, i_c')$ *and selects action* $a_{i_c'} = \tilde{a}_{i_c}$ *to form* $\widehat{\boldsymbol{a}} = [\boldsymbol{a}; \tilde{\boldsymbol{a}}_c] \in \mathcal{A}^{n+n_c'}$*, where* $n_c' = |\mathcal{N}_c'|$*,* $\boldsymbol{a} \in \mathcal{A}^n$ *is the original joint action, and* $\tilde{\boldsymbol{a}}_c$ *is the most recent joint decision made by all agents* i_c*. Subsequently, they move to the next state* $\widehat{s}'$ *according to* $\widehat{\mathcal{P}}(\widehat{s}'|\widehat{s}, \widehat{\boldsymbol{a}}) = \mathcal{O}_{\mathcal{P}}(\tilde{\boldsymbol{o}}_c'|s')\mathcal{P}(s'|s, \boldsymbol{a})$ *and earn a joint reward* $\widehat{r}(\widehat{s}, \widehat{\boldsymbol{a}}) = r(s, \boldsymbol{a})$[1].

To ensure consistent dimensions of the extended state $\widehat{s}$ and action $\widehat{\boldsymbol{a}}$, virtual proxies are introduced for decision-making

[1]This setup allows us to investigate the effect of asynchronous actions $\tilde{\boldsymbol{a}}_c$ on the reward $r(s, \boldsymbol{a})$.

agents $i_d \in \mathcal{N}_d$ and action-executing agents $i_c \in \mathcal{N}_c$ at each time step, with proxies for agents i_d masked out.

Our VSP improves the training efficiency without increasing the complexity. Since a virtual proxy shares policy with the corresponding real agent, the introduction of a proxy does not expand the policy space. Furthermore, during the execution of action a_3^1 in Figure 2, virtual proxy $\#3'$ is repeatedly introduced, enabling multiple updates to the shared policy associated with action a_3^1. This significantly accelerates convergence, as demonstrated by the ablation studies in Section 6.2. Theoretically, we prove that VSP preserves the inherent characteristics of the task and the solution process.

Theorem 1. *Given an asynchronous decision-making task, define $\boldsymbol{\pi}^*_{Dec}$ and $\boldsymbol{\pi}^*_{VSP}$ as the Markov Perfect Equilibrium (MPE) respectively for modeling as a Dec-POMDP and a Dec-POMDP with VSP. $\mathcal{T}^*_{Dec}$ and $\mathcal{T}^*_{VSP}$ as the VD operator for the same. Assuming that $\mathcal{T}^*_{Dec}$ converges to the MPE of this task, i.e., $\boldsymbol{\pi}^*_{Dec}$, then:*
(1) $\boldsymbol{\pi}^*_{VSP} = \boldsymbol{\pi}^*_{Dec}$[2];
(2) Given the same initial joint policy $\boldsymbol{\pi}_0$, $\mathcal{T}^*_{Dec}$ and $\mathcal{T}^*_{VSP}$ converge to the same MPE.

Proof. Please see Appendix B. □

5 MVD

5.1 Multiplicative Interaction Form

Based on VSP, asynchronous credit assignment can be addressed in a synchronized manner within a single time step. According to the unified framework of general VD algorithms, the global Q-value Q_{tot} with VSP mechanism is formulated in terms of individual utilities Q_i as follows:

$$Q_{tot} = Q_{tot}(s, Q_{1_d}, \cdots, Q_{n_d}, Q_{1_c}, \cdots, Q_{n_c}, Q_{1'_c}, \cdots, Q_{n'_c}). \tag{4}$$

In asynchronous tasks, the agent who executes first must predict how later choices of other agents would affect its execution. Meanwhile, the agent who executes later needs to consider the impact of the current actions of other agents on its decision. This **conditional dependency** is represented in Eq. (4) as the **interaction** between Q_{i_d} and $Q_{i'_c}$. Considering the benefits of multiplicative interactions for fusing information streams and enabling conditional computation [Rumelhart and McClelland, 1987; Jayakumar *et al.*, 2020], we enrich Eq. (1) by incorporating multiplicative interactions to capture these dependencies. We propose the general Multiplicative Value Decomposition (MVD) formula:

$$Q_{tot} = k_0 + \sum_{i}^{n+n'_c} k_i Q_i + \sum_{i_d, i'_c} k_{i_d i'_c} Q_{i_d} Q_{i'_c}. \tag{5}$$

The detailed derivations are provided in Appendix C.1. We derive Eq. (5) by performing a Taylor expansion of Q_i near an optimal joint action, providing support for the multiplication operation present in the value decomposition process.

Furthermore, compared to the traditional additive interaction VD, the multiplicative interaction between Q_{i_d} and $Q_{i'_c}$ in MVD significantly boosts representational capability in learning interactions among agents. For Eq. (1), the gradient for updating Q_i is $\frac{\partial Q_{tot}}{\partial Q_i} = k_i$. In contrast, the gradients from Eq. (5) are $\frac{\partial Q_{tot}}{\partial Q_{i_d}} = k_{i_d} + \sum_{i'_c} k_{i_d i'_c} Q_{i'_c}$, $\frac{\partial Q_{tot}}{\partial Q_{i'_c}} = k_{i'_c} + \sum_{i_d} k_{i_d i'_c} Q_{i_d}$, and $\frac{\partial Q_{tot}}{\partial Q_{i_c}} = k_{i_c}$. Therefore, MVD integrates the nonlinear interactions as contextual information, allowing Q_{i_d} and $Q_{i'_c}$ to refine their policies based on their mutual influence. We prove that MVD bears advantages in handling asynchronous tasks over traditional VD.

Theorem 2. *Given an asynchronous decision-making task, define $\mathcal{Q}_{Add}$ as the function class for the additive VD operator $\mathcal{T}^*_{Add}$, and $\mathcal{Q}_{Mul}$ as the function class for the multiplicative interaction VD operator $\mathcal{T}^*_{Mul}$, then:*

$$\mathcal{Q}_{Add} \subsetneq \mathcal{Q}_{Mul}.$$

Proof. Please see Appendix B. □

5.2 High-Order Interaction Form

Eq. (5) primarily considers the interaction of Q_{i_d} and $Q_{i'_c}$. However, as illustrated in Figure 2a, the actions a_1^0 and a_3^1 re-executed by virtual proxies $\#1'$ and $\#3'$ at t_2 actually originate from different time steps. This implies interactions between $Q_{1'}$ and $Q_{3'}$, thereby indicating high-order interactions among $Q_{1'}$, $Q_{3'}$, and Q_2. Therefore, we propose a K-th order (where $1 \leq K \leq n$) interactive VD formula as follows:

$$\begin{aligned} Q_{tot} = k_0 + \sum_{i}^{n+n'_c} k_i Q_i + \sum_{i_d, i'_c} k_{i_d i'_c} Q_{i_d} Q_{i'_c} \\ + \cdots + \sum_{i_d, i'_{c,1}, \cdots, i'_{c,K-1}} k_{i_d i'_{c,1} \cdots i'_{c,K-1}} Q_{i_d} Q_{i'_{c,1}} \cdots Q_{i'_{c,K-1}}. \end{aligned} \tag{6}$$

The detailed derivations are provided in Appendix C.2. Based on the derivation of Eq. (6), as the order increases, the error introduced by Taylor expansion decreases and agents are able to obtain deeper interactive information. Nevertheless, higher-order interaction complicates the model and does not necessarily lead to better final performance [Wen *et al.*, 2019; Liu *et al.*, 2023]. Our ablation studies in Section 6.2 further confirm this conclusion. Therefore, we primarily focus on multiplicative interactions between Q_{i_d} and $Q_{i'_c}$.

5.3 Implementation

Finally, we discuss the issue of IGM consistency in the practical implementation of MVD. In Dec-POMDP with VSP, the agent i_c currently executing an action does not need to choose a new one, and virtual proxy i'_c can only execute the asynchronous actions of i_c. This implies that agent i_c and proxy i'_c do not need to satisfy the IGM condition. Consequently, we obtain the MVD-based IGM as follows:

$$\begin{aligned} \arg\max_{\boldsymbol{a}} Q_{tot}(s, \boldsymbol{a}) = (\arg\max_{a_{1_d}} Q_{1_d}(\tau_{1_d}, a_{1_d}), \\ \cdots, \arg\max_{a_{n_d}} Q_{n_d}(\tau_{n_d}, a_{n_d}), \\ a_{1_c}, \cdots, a_{n_c}, a_{1'_c}, \cdots, a_{n'_c}). \end{aligned} \tag{7}$$

[2]Since VSP does not affect the policy space, the joint policy can still be denoted by $\boldsymbol{\pi}$ from Dec-POMDP with n agents.

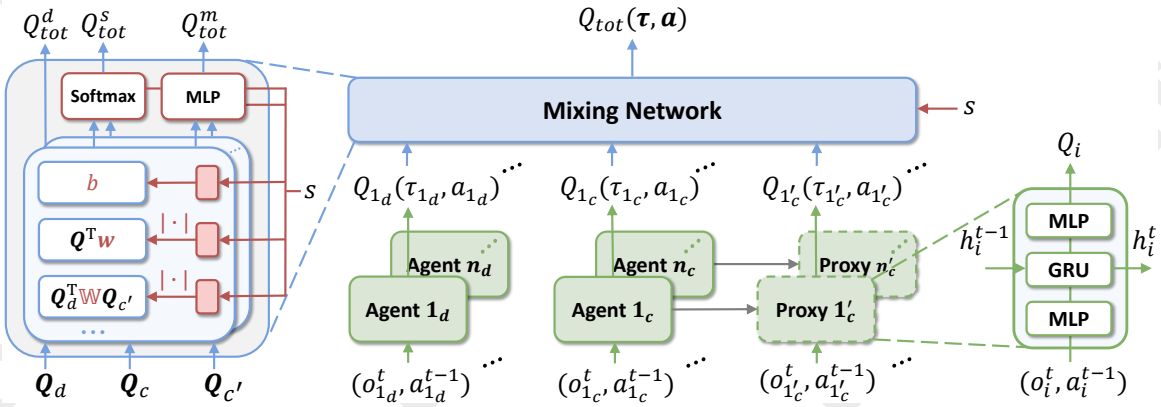


Figure 3: The overall framework of MVD. **Left:** Mixing network structure. In red are the hypernetworks that generate the weights and biases for mixing network. **Middle:** The overall MVD architecture. **Right:** Agent network structure.

To maintain the monotonicity between Q_{tot} and Q_{i_d}, i.e., $\frac{\partial Q_{tot}}{\partial Q_{i_d}} \geq 0$, we derive the following form that satisfies MVD-based IGM and employ hypernetworks [Ha *et al.*, 2017] $f_i(s)$ to learn the corresponding weights. Since $Q_{i'_c}$ may be less than 0, we track the minimum $Q_{i'_c}$ during training and denote it as Q_c^{min}, ensuring $Q_{i'_c} + Q_c^{min} \geq 0$. The detailed derivations are provided in Appendix C.3.

$$Q_{tot} \approx f_0 + \sum_{i}^{n+n'_c} |f_i| Q_i + \sum_{i_d, i'_c} |f_{i_d i'_c}| \frac{Q_{i'_c} + Q_c^{min}}{2} Q_{i_d}. \quad (8)$$

The overall framework of MVD is illustrated in Figure 3. We propose three distinct practical implementations of the mixing network. The first approach directly calculates the final global Q value based on Eq. (8), denoted as Q_{tot}^d. The second approach employs a multi-head structure that allows the mixing network to focus on asynchronous interaction information from different representation sub-spaces, thereby enhancing the representational capability and stability. Each head calculates a global Q-value based on Eq. (8), and inputs it into a Softmax function to obtain the final global Q-value, denoted as Q_{tot}^s. The third approach also employs a multi-head mechanism. To simplify the model, we use a ReLU activation function and an MLP to replace the Softmax function in the second implementation, denoted as Q_{tot}^m. In this paper, we primarily focus on the third implementation, and comparisons with the others are discussed in Section 6.2. The pseudo-code for MVD is in Appendix D.

6 Experiments

In this section, we evaluate our MVD on a modified asynchronous variant of SMAC and two existing asynchronous benchmarks: Overcooked and POAC. SMAC is a widely used testbed for MARL algorithms. Our asynchronous variant introduces asynchronous actions, where allied agents require different time steps to complete their movements. Overcooked requires agents to prepare ingredients in sequence and cooperate to make salads. Different actions, such as chopping, moving to ingredients, and delivering, span varying time steps. POAC is a confrontation wargame between two armies with three unit types, each having distinct attributes and action execution times. The goal is to learn asynchronous strategies to defeat the rule-based bots.

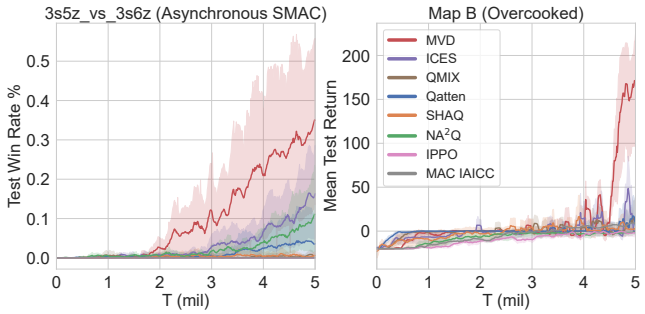


Figure 4: Performance on two challenging asynchronous scenarios

The baselines fall into three categories: (1) The decentralized training and decentralized execution (DTDE) method, IPPO [de Witt *et al.*, 2020], treats other agents as part of the environment and is applicable to asynchronous tasks. (2) The discarding type method, MAC IAICC, which is the most advanced algorithm in [Xiao *et al.*, 2022]. (3) Credit assignment algorithms based on padding type method, including QMIX, Qatten, SHAQ [Wang *et al.*, 2022], ICES [Li *et al.*, 2024], and NA^{2}Q that considers 2nd-order interactions[3].

Details of benchmarks, baselines, and our MVD are provided in Appendix E. The graphs illustrate the performance of all compared algorithms by plotting the mean and standard deviation of results obtained across five random seeds.

6.1 Benchmark Results

We run experiments across multiple benchmarks, focusing on three key aspects of our framework in asynchronous cooperation: the necessity of additional computation, effectiveness against baselines, and generalization in complex tasks.

[3]We excluded other asynchronous MARL algorithms in the section of Related Works due to their specificity to tasks with individual agent rewards or lack of open-source code.

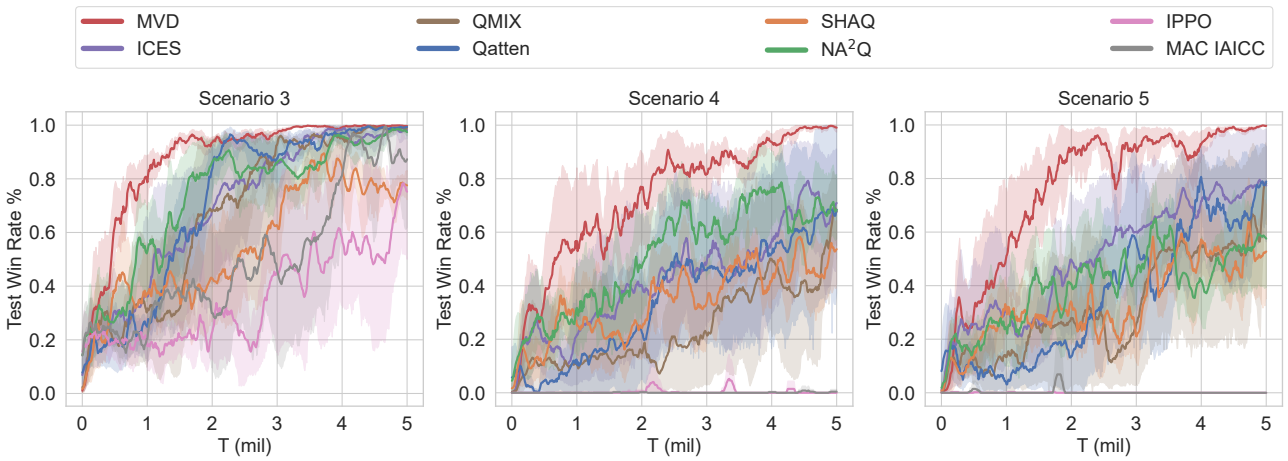


Figure 5: Test win rate % on three scenarios of POAC benchmark.

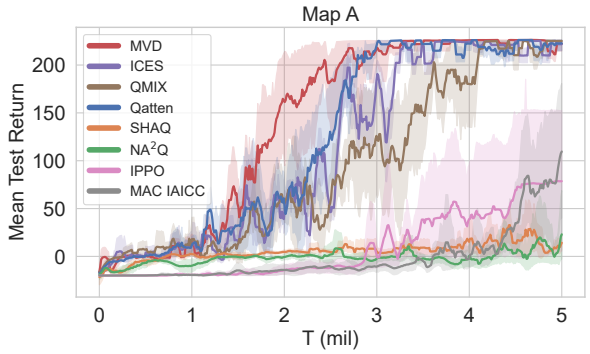


Figure 6: Mean test return on Overcooked benchmark.

We first investigate why the additional computations introduced by VSP and MVD are necessary for asynchronous cooperation. As shown in Figure 4, in the super hard scenario of asynchronous SMAC and map B of Overcooked, both discarding and padding methods fail to handle asynchronous cooperation effectively. In contrast, MVD employs virtual proxies and multiplicative interactions to better capture the marginal contributions of asynchronous actions, significantly accelerating convergence. The complete experimental results and analysis are provided in Appendix F. These results show that while most baselines perform well in simple asynchronous scenarios, they struggle in complex tasks.

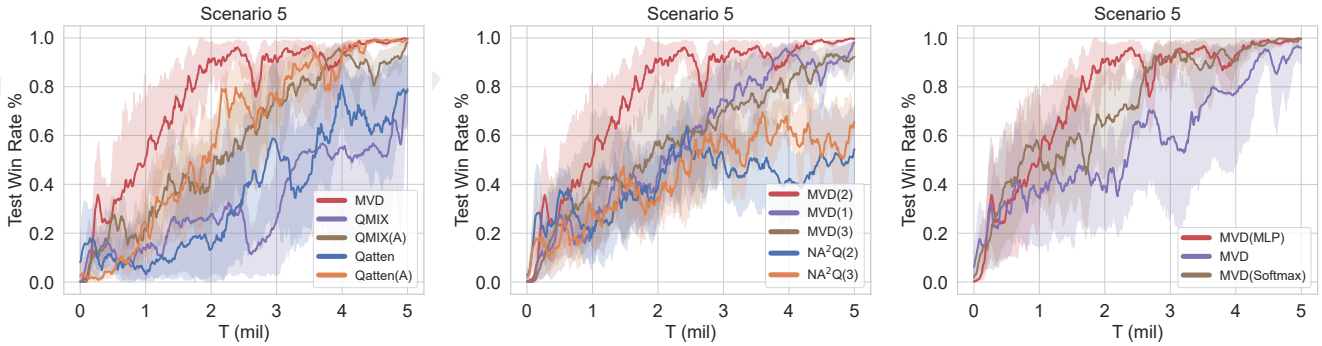
We then analyze the specific performance comparison against baselines on map A of Overcooked, as shown in Figure 6. The results indicate that MVD surpasses all baselines. Both IPPO and MAC IAICC exhibit slower training speeds. This suggests the discarding type methods suffer from low efficiency. NA^{2}Q and SHAQ mistakenly consider the influence among padding actions, resulting in non-convergence. This implies that Dec-POMDP with padding action is also unsuitable for asynchronous settings. Although ICES enhances exploration, it is less effective than MVD as it neglects the interplay between asynchronous actions. QMIX and Qatten perform better than NA^{2}Q because they use simpler models to handle credit assignment, leading to stronger robustness to padding action. Furthermore, MVD outperforms baselines on other maps of Overcooked. The complete experimental results and analysis are in Appendix F.2.

We further validate the generalization of MVD on the challenging POAC. Figure 5 shows the win rate across scenarios with increasing difficulty levels. We observe that MVD demonstrates increasingly better performance. IPPO and MAC IAICC train slowly in simple scenarios and fail in complex tasks. Compared to Overcooked, POAC contains fewer asynchronous actions, resulting in less padding when using Dec-POMDP. Therefore, NA^{2}Q performs relatively better among the baselines. However, due to the interference from padding action and the complexity of the adopted model, the training efficiency of NA^{2}Q is consistently inferior to that of MVD, a result also observed in SHAQ and ICES. The additive interaction VD algorithms, QMIX and Qatten, do not yield satisfactory performance, since they cannot handle the mutual influence among agents in complex asynchronous tasks. Furthermore, MVD demonstrates highly competitive performance in other scenarios of POAC. The complete experimental results and analysis are in Appendix F.3.

6.2 Ablation Studies

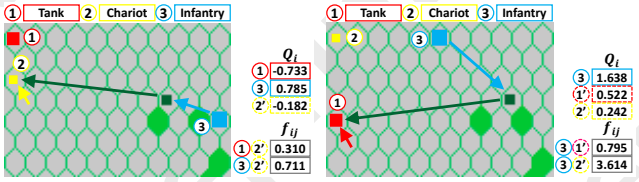
To obtain a deeper insight into our proposed VSP and MVD, we perform ablation studies to illustrate the impact of the following factors on the performance: (1) The introduction of virtual proxies. (2) The interactions of different orders among agents. (3) The distinct practical implementations of MVD.

For evaluating the impact of factor (1), we extend QMIX and Qatten with VSP, denoting them as QMIX(A) and Qatten(A). As shown in Figure 7a, the introduction of virtual proxies does not complicate the problem but instead notably improves the performance of QMIX and Qatten, highlighting the powerful advantages of VSP in an asynchronous setting. However, they fail to capture the mutual influence among agents, leading to poorer performance than MVD. We also extend other VD algorithms with VSP. The complete ablation experiments and analysis are in Appendix G.1.



(a) Virtual proxies (b) Different orders interactions (c) Different implementations

Figure 7: Ablation Studies of MVD on scenario 5 of POAC benchmark.



(a) (b)

Figure 8: Visualization of evaluation for (a) MVD and (b) NA^{2}Q.

For evaluating the impact of factor (2), we consider MVD and NA^{2}Q with l-th order interactions under VSP, denoting them as MVD(l) and NA^{2}Q(l). As shown in Figure 7b, the performance of MVD(2) incorporating multiplicative interactions consistently outperforms MVD(1) which does not consider interactions between agents. Further, both MVD(3) and NA^{2}Q(3) suffer from performance degradation due to increased model complexity. Therefore, additive interactions, which ignore interactions, are insufficient for solving the asynchronous credit assignment problem, while higher-order interactions increase complexity and degrade performance. In contrast, the multiplicative interaction between Q_{i_d} and $Q_{i'_c}$ efficiently addresses these issues. The complete ablation experiments and analysis are in Appendix G.2.

For evaluating the impact of factor (3), we compared three different practical implementations of MVD. As shown in Figure 7c, directly applying Eq. (8) to obtain Q^d_{tot} converges to the optimal joint policy, yet it suffers from slow training speed and instability. Employing a multi-head structure can effectively address these issues. However, using Softmax to obtain Q^s_{tot} excessively complicates the entire mixing network. Therefore, MVD derives the greatest benefit from MLP with the ReLU activation function. The complete ablation experiments and analysis are in Appendix G.3.

6.3 Interpretability

To visually illustrate the asynchronous credit assignment process, we exhibit key frames from scenario 5 of POAC and compare the converged MVD and NA^{2}Q with VSP. The arrows in the figures represent the movements or attacks of the agents. We highlight the individual Q_i and crucial weights within the mixing network to demonstrate their alignment with agents' asynchronous behaviors. f_{ij} represents the collaborative contribution of two agents to the global outcome.

Figures 8a and 8b depict similar scenarios: one of our units lures the enemy deeper into the field, while another attack. Since the former is executing a movement action and the latter is making an attack decision, their decision-making is asynchronous. As shown in Figure 8a, the infantry successfully attacks an enemy, leading to a higher Q_3, while the chariot is under attack, resulting in a negative $Q_{2'}$. MVD accurately attributes the importance of their asynchronous cooperation to the entire team using multiplicative interaction and assigns a higher credit $f_{32'}$. As shown in Figure 8b, similarly, the infantry that successfully attacks enemies has a higher Q_3, whereas the tank used to kite enemies has a lower $Q_{1'}$. Both the tank and the chariot are executing actions, yet NA^{2}Q mistakenly regards the asynchronous infantry-chariot cooperation as more important than infantry-tank cooperation. Therefore, even though both strategies ultimately achieve victory, MVD offers a superior ability to capture the interplay between agents' asynchronous decision-making, providing higher interpretability.

7 Conclusion and Future Work

In this paper, we propose an asynchronous credit assignment framework incorporating the VSP mechanism and the MVD algorithm. Our framework fully captures the dependencies between asynchronous decisions and provides a solid basis for further exploration of asynchronous MARL. VSP synchronizes asynchronous actions without disrupting task equilibrium or VD convergence. MVD introduces multiplicative interactions, strictly extending the function class and effectively capturing the interplay between asynchronous actions. Extensive experiments demonstrate that MVD outperforms baselines, particularly in complex asynchronous tasks, and provides interpretability for asynchronous cooperation. One direction for future work involves exploring effective representations of higher-order asynchronous interactions and addressing asynchronous cooperation in large-scale systems.

Acknowledgments

This paper was mainly supported by the National Natural Science Foundation of China (NSFC) under Grant 62272497 to H. Wu.

References

[Amato *et al.*, 2019] Christopher Amato, George Konidaris, Leslie P Kaelbling, and Jonathan P How. Modeling and planning with macro-actions in decentralized pomdps. *Journal of Artificial Intelligence Research*, 64:817–859, 2019.

[Arulkumaran *et al.*, 2019] Kai Arulkumaran, Antoine Cully, and Julian Togelius. Alphastar: An evolutionary computation perspective. In *Proceedings of the genetic and evolutionary computation conference companion*, pages 314–315, 2019.

[Chen *et al.*, 2021] Yuxin Chen, Hejun Wu, Yongheng Liang, and Guoming Lai. Varlenmarl: A framework of variable-length time-step multi-agent reinforcement learning for cooperative charging in sensor networks. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking*, pages 1–9. IEEE, 2021.

[Daganzo and Ouyang, 2019] Carlos F Daganzo and Yanfeng Ouyang. *Public transportation systems: Principles of system design, operations planning and real-time control*. World Scientific, 2019.

[de Witt *et al.*, 2020] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge?, 2020.

[Dolev *et al.*, 2018] Shlomi Dolev, Chryssis Georgiou, Ioannis Marcoullis, and Elad M. Schiller. Practically-self-stabilizing virtual synchrony. *Journal of Computer and System Sciences*, 96:50–73, 2018.

[Foerster *et al.*, 2018] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Ha *et al.*, 2017] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

[Hastie, 2017] Trevor J Hastie. Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge, 2017.

[Jayakumar *et al.*, 2020] Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. Multiplicative interactions and where to find them. In *International Conference on Learning Representations*, 2020.

[Jia *et al.*, 2020] Hangtian Jia, Yujing Hu, Yingfeng Chen, Chunxu Ren, Tangjie Lv, Changjie Fan, and Chongjie Zhang. Fever basketball: A complex, flexible, and asynchronized sports game environment for multi-agent reinforcement learning, 2020.

[Li *et al.*, 2022] Jiahui Li, Kun Kuang, Baoxiang Wang, Furui Liu, Long Chen, Changjie Fan, Fei Wu, and Jun Xiao. Deconfounded value decomposition for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 12843–12856. PMLR, 2022.

[Li *et al.*, 2024] Xinran Li, Zifan Liu, Shibo Chen, and Jun Zhang. Individual contributions as intrinsic exploration scaffolds for multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2024.

[Liang *et al.*, 2022] Yongheng Liang, Hejun Wu, and Haitao Wang. ASM-PPO: Asynchronous and scalable multi-agent ppo for cooperative charging. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 798–806, 2022.

[Liang *et al.*, 2023] Yongheng Liang, Hejun Wu, and Haitao Wang. Asynchronous multi-agent reinforcement learning for collaborative partial charging in wireless rechargeable sensor networks. *IEEE Transactions on Mobile Computing*, 2023.

[Liu *et al.*, 2023] Zichuan Liu, Yuanyang Zhu, and Chunlin Chen. NA^2Q: Neural attention additive model for interpretable multi-agent q-learning. In *International Conference on Machine Learning*, pages 22539–22558. PMLR, 2023.

[Oliehoek *et al.*, 2008] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

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

[Oroojlooy and Hajinezhad, 2023] Afshin Oroojlooy and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, 53(11):13677–13722, 2023.

[Rashid *et al.*, 2020] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

[Rumelhart and McClelland, 1987] David E. Rumelhart and James L. McClelland. *A General Framework for Parallel Distributed Processing*, pages 45–76. 1987.

[Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, pages 2186–2188, 2019.

[Shapley, 1953] Lloyd S Shapley. A value for n-person games. 1953.

[Son *et al.*, 2019] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR, 2019.

[Sundararajan *et al.*, 2017] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[Sunehag *et al.*, 2018] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 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*, page 2085–2087, 2018.

[Theocharous and Kaelbling, 2003] Georgios Theocharous and Leslie Kaelbling. Approximate planning in pomdps with macro-actions. *Advances in neural information processing systems*, 16, 2003.

[Wang and Sun, 2021] Jiawei Wang and Lijun Sun. Reducing bus bunching with asynchronous multi-agent reinforcement learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence,*, pages 426–433, 2021.

[Wang *et al.*, 2016] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

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

[Wang *et al.*, 2020b] Rose E Wang, Sarah A Wu, James A Evans, Joshua B Tenenbaum, David C Parkes, and Max Kleiman-Weiner. Too many cooks: Coordinating multi-agent collaboration through inverse planning. 2020.

[Wang *et al.*, 2021] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2021.

[Wang *et al.*, 2022] Jianhong Wang, Yuan Zhang, Yunjie Gu, and Tae-Kyun Kim. Shaq: Incorporating shapley value theory into multi-agent q-learning. *Advances in Neural Information Processing Systems*, 35:5941–5954, 2022.

[Wen *et al.*, 2019] Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2019.

[Wolpert and Tumer, 2001] David H Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(02n03):265–279, 2001.

[Xiao *et al.*, 2022] Yuchen Xiao, Weihao Tan, and Christopher Amato. Asynchronous actor-critic for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:4385–4400, 2022.

[Yang *et al.*, 2020a] Yaodong Yang, Jianye Hao, Guangyong Chen, Hongyao Tang, Yingfeng Chen, Yujing Hu, Changjie Fan, and Zhongyu Wei. Q-value path decomposition for deep multiagent reinforcement learning. In *International Conference on Machine Learning*, pages 10706–10715. PMLR, 2020.

[Yang *et al.*, 2020b] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multi-agent reinforcement learning, 2020.

[Yao *et al.*, 2021] Meng Yao, Qiyue Yin, Jun Yang, Tongtong Yu, Shengqi Shen, Junge Zhang, Bin Liang, and Kaiqi Huang. The partially observable asynchronous multi-agent cooperation challenge, 2021.

[Yu *et al.*, 2022] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[Zhou *et al.*, 2020] Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:11853–11864, 2020.

[Zhou *et al.*, 2023] Xiaobo Zhou, Zhihui Ke, and Tie Qiu. Recommendation-driven multi-cell cooperative caching: A multi-agent reinforcement learning approach. *IEEE Transactions on Mobile Computing*, 2023.