Agent-Oriented Centralized Critic for Asynchronous Multi-Agent Reinforcement Learning

Sunghoon Hong*
LG AI Research
Seoul, South Korea
sunghoon.hong@lgresearch.ai

Whiyoung Jung*
LG AI Research
Seoul, South Korea
whiyoung.jung@lgresearch.ai

Deunsol Yoon*
LG AI Research
Seoul, South Korea
dsyoon@lgresearch.ai

Kanghoon Lee
LG AI Research
Seoul, South Korea
kanghoon.lee@lgresearch.ai

Woohyung Lim
LG AI Research
Seoul, South Korea
w.lim@lgresearch.ai

ABSTRACT

Multi-agent reinforcement learning (MARL) has been actively developed and successfully applied in various fields. In the conventional MARL setting, which most previous works consider, all agents simultaneously take their actions every time due to the same duration across actions. However, real-world scenarios often involve agents executing actions with different duration resulting in asynchronous action selection across the agents. The macro-action decentralized partially observable Markov decision process (MacDec-POMDP) provides a framework for modeling multi-agent decision-making, where the action selection among the agents occurs asynchronously across time. While several works have explored MARL methods for MacDec-POMDP, existing methods for such asynchronicity focused on how to utilize trajectories for training and simply adopt conventional MARL architectures. In this paper, we propose a novel approach named agent-oriented centralized critic (AOCC) for MacDec-POMDP, which 1) explicitly encode each agent's observation history with the timestep information when the agent start to perform a macro-action, and 2) explicitly aggregating them for agent-oriented critic learning. Our experimental evaluation on a macro-action-based multi-agent benchmark demonstrates that the proposed approach significantly outperforms other baseline methods for MacDec-POMDP.

KEYWORDS

Agent-Oriented Centralized Critic, MacDec-POMDP, Asynchronous Multi-Agent Reinforcement Learning

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) has witnessed significant advancements in recent years, enabling decentralized agents to collaborate and complete assigned tasks effectively [5, 6, 8, 12, 13, 20]. However, a prevailing limitation of existing MARL methods lies in their strong assumption of synchronous action selection, which poses challenges when applying these methods to real-world MARL scenarios with asynchronous agents.

Real-world scenarios often involve agents executing actions asynchronously, where they start and finish their actions at different times. This asynchronicity is prevalent in various domains, such as manufacturing, logistics, and traffic management. In such scenarios, the application of MARL methods that assume synchronous actions becomes impractical, necessitating the development of approaches explicitly tailored to handle asynchronicity.

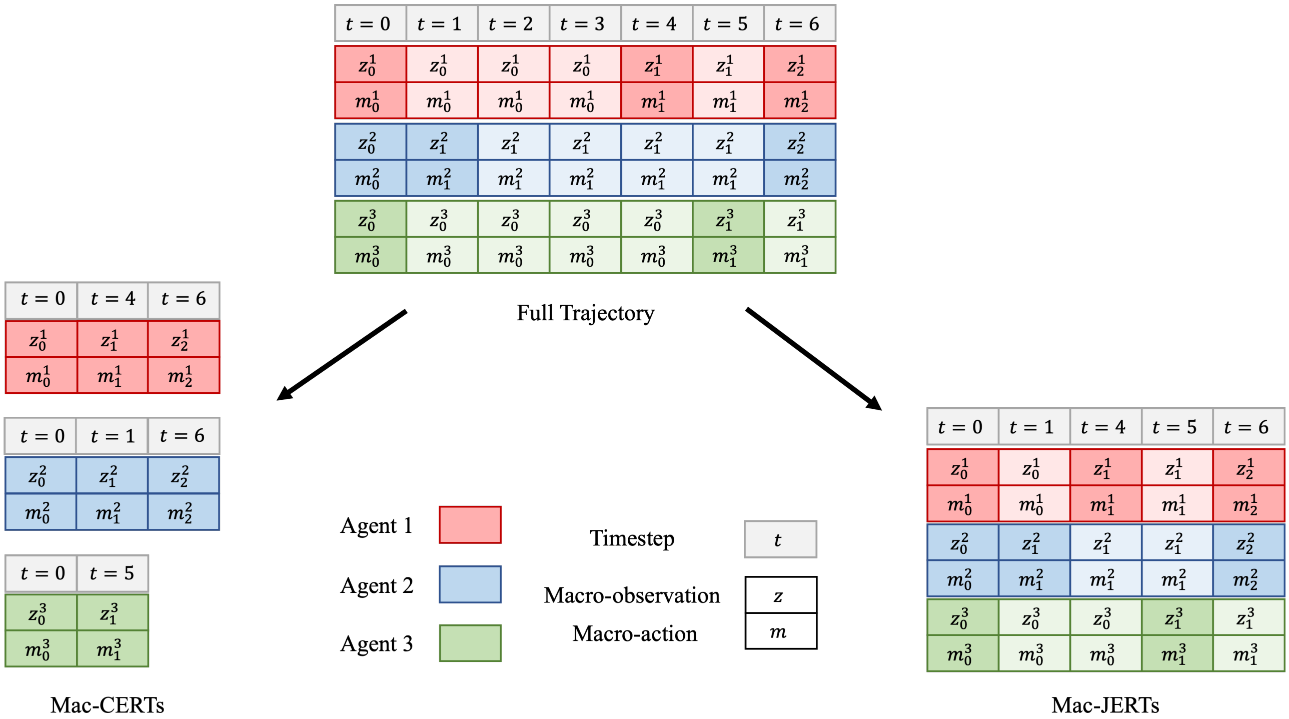
One promising framework for addressing the challenge of asynchronous MARL is the macro-action decentralized partially observable Markov decision process (MacDec-POMDP) [2, 3]. This framework extends the options framework [14] to the multi-agent domain, providing a flexible and general decision-making framework. In MacDec-POMDP, the time at which each agent's action starts and the duration of the action can be different. While several planning methods have been proposed for MacDec-POMDP [1, 7, 10, 11], recent research has also explored learning-based approaches to tackle the challenges posed by asynchronicity [17–19].

Existing MARL based methods for MacDec-POMDP have primarily focused on constructing a training buffer for asynchronous setting while adopting conventional MARL architectures originally designed for synchronous settings, resulting in the introduction of duplicate macro-observations for the centralized critic. Since the common centralized critic structure for the conventional synchronous MARL focuses on joint histories along the time axis, a timestep that one agent can start to perform a macro-action while other agents cannot, introduces a duplicate macro-observation for the non-macro-actionable agents in the centralized critic. However, with these duplicate macro-observations, the centralized critic may provide inaccurate evaluations.

To address the limitations of these approaches and enable efficient asynchronous learning, it is crucial to consider specialized architectures that can effectively capture the individual agent histories and facilitate reasoning across agents. In this paper, we propose a novel approach named agent-oriented centralized critic to address asynchronous MARL in MacDec-POMDP. By 1) succintly encoding the observation history of each agent independently with the timestep information when the agent start to perform a macro-action and 2) explicitly aggregating them, our approach enables an agent-oriented centralized critic learning in asynchronous settings. We evaluate the effectiveness of our proposed method on macro-action-based multi-agent benchmarks. The results of our experiments demonstrate the superiority of our approach compared to conventional methods, particularly in environments that require complex cooperation and generalization.

*Equal contribution

Proc. of the Adaptive and Learning Agents Workshop (ALA 2024), Avalos, Müller, Wang, Yates (eds.), May 6-7, 2024, Online, https://ala2024.github.io/. 2024.



Full Trajectory

$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
z_0^1	z_0^1	z_0^1	z_0^1	z_1^1	z_1^1	z_2^1
m_0^1	m_0^1	m_0^1	m_0^1	m_1^1	m_1^1	m_2^1
z_0^2	z_1^2	z_1^2	z_1^2	z_1^2	z_1^2	z_2^2
m_0^2	m_1^2	m_1^2	m_1^2	m_1^2	m_1^2	m_2^2
z_0^3	z_0^3	z_0^3	z_0^3	z_0^3	z_1^3	z_1^3
m_0^3	m_0^3	m_0^3	m_0^3	m_0^3	m_1^3	m_1^3

Mac-CERTs

$t=0$	$t=4$	$t=6$
z_0^1	z_1^1	z_2^1
m_0^1	m_1^1	m_2^1

$t=0$	$t=1$	$t=6$
z_0^2	z_1^2	z_2^2
m_0^2	m_1^2	m_2^2

$t=0$	$t=5$
z_0^3	z_1^3
m_0^3	m_1^3

Mac-JERTs

$t=0$	$t=1$	$t=4$	$t=5$	$t=6$
z_0^1	z_0^1	z_1^1	z_1^1	z_2^1
m_0^1	m_0^1	m_1^1	m_1^1	m_2^1
z_0^2	z_1^2	z_1^2	z_1^2	z_2^2
m_0^2	m_1^2	m_1^2	m_1^2	m_2^2
z_0^3	z_0^3	z_0^3	z_1^3	z_1^3
m_0^3	m_0^3	m_0^3	m_1^3	m_1^3

Figure 1: Example of a trajectory and two training buffers in MacDec-POMDP. In this example, the full trajectory represents macro-observations (denoted by z), macro-actions (denoted by m), and corresponding timesteps. Note that a set of macro-actions is predefined, each with a different duration. Mac-CERTs is a squeezed trajectory per agent while Mac-JERTs is a joint squeezed trajectory.

2 BACKGROUND

2.1 MacDec-POMDP

The macro-action decentralized partially observable Markov decision process (MacDec-POMDP) [3] incorporates the option framework [14] into the decentralized partially observable Markov decision process (Dec-POMDP) by defining a set of macro-actions for each agent.

Followed by the previous work [19], a MacDec-POMDP is represented as a tuple $\langle \mathcal{I}, \mathcal{S}, \tilde{\mathcal{A}}, \tilde{\mathcal{M}}, \tilde{\Omega}, \tilde{\zeta}, T, R, \tilde{O}, \tilde{Z} \rangle$, where $\mathcal{I} = \{1, \dots, N\}$ is a set of indices of agents, $\mathcal{S}$ is the state space, $\tilde{\mathcal{A}} = \prod_{i \in \mathcal{I}} \mathcal{A}^i$ is the joint primitive-action space, $\tilde{\mathcal{M}} = \prod_{i \in \mathcal{I}} \mathcal{M}^i$ is the joint macro-action space, $\tilde{\Omega} = \prod_{i \in \mathcal{I}} \Omega^i$ is the joint primitive-observation space, $\tilde{\zeta} = \prod_{i \in \mathcal{I}} \zeta^i$ is the joint macro-observation space, $T(s'|s, \tilde{a})$ is the state transition probability, R is the reward function shared over all agents, $\tilde{O}(\tilde{o}|s', \tilde{a}), \tilde{o} \in \tilde{\Omega}$ is the joint observation probability, and $\tilde{Z}(\tilde{z}|s', \tilde{m}), \tilde{z} \in \tilde{\zeta}$ is the joint macro-observation probability. Here, we denote $(\cdot)^i$ as a space or an element of agent i and $\tilde{(\cdot)}$ is a product or a joint of $(\cdot)^i$ over all agents, and we use these notations throughout this paper. Each macro-action, a.k.a. an option, is a tuple $m^i = \langle \beta_{m^i}, \mathcal{I}_{m^i}, \pi_{m^i} \rangle \in \mathcal{M}^i$ composed of a termination condition $\beta_{m^i} : \mathcal{H}^i_{\text{pri}} \to [0, 1]$, a initiation set $\mathcal{I}_{m^i} \subset \mathcal{H}^i_{\text{mac}}$, and a low-level policy $\pi_{m^i} : \mathcal{H}^i_{\text{pri}} \to \mathcal{A}^i$, where $\mathcal{H}^i_{\text{pri}}$ (or $\mathcal{H}^i_{\text{mac}}$) is the primitive(or macro)-action-observation history space. The objective in MacDec-POMDP is then to find a joint high-level policy $\tilde{\Psi} = \prod_{i \in \mathcal{I}} \Psi^i$ that maximizes the following expected discounted return from an initial state s_0 for given low-level policies, i.e., macro-actions:

$$\tilde{\Psi}^* = \arg\max_{\tilde{\Psi}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, \tilde{a}_t\right) \mid s_0, \tilde{\Psi} \right] \tag{1}$$

2.2 Asynchronous MARL

Several works [17, 18] have been proposed for learning a joint high-level policy $\tilde{\Psi}$ in MacDec-POMDP. In particular, these works propose special training buffers for asynchronous settings: Macro-action concurrent experience replay trajectories (Mac-CERTs) and macro-action joint experience replay trajectories (Mac-JERTs). We denote a macro-observation and a macro-action of agent i as z^i and m^i. In Mac-CERTs, the transition experience of each agent i is represented as a tuple $\langle z^i, m^i, z^{i\prime}, r^i \rangle$, where r^i represents the cumulative reward for the macro-action m^i starting at timestep t^i and lasting τ^i time steps, defined as $r^i = \sum_{t=t^i}^{t^i+\tau^i-1} \gamma^{t-t^i} r_t$.

We denote a joint macro-observation and a joint macro-action as $\tilde{z}$ and $\tilde{m}$, respectively. In Mac-JERTs, the transition experience is represented as a tuple $\langle \tilde{z}, \tilde{m}, \tilde{z}', \tilde{r} \rangle$, where $\tilde{r}$ represents the cumulative reward for the joint macro-action $\tilde{m}$ defined as $\tilde{r} = \sum_{t=\tilde{t}}^{\tilde{t}+\tilde{\tau}-1} \gamma^{t-\tilde{t}} r_t$. Unlike Mac-CERTs, $\tilde{t}$ denotes the timestep when any agent starts its own macro-action and $\tilde{t}+\tilde{\tau}-1$ is the ending timestep when any agent finishes its macro-action. For example, in Figure 1, the agent 2 starts its own action at timestep 1, resulting in the joint macro-action

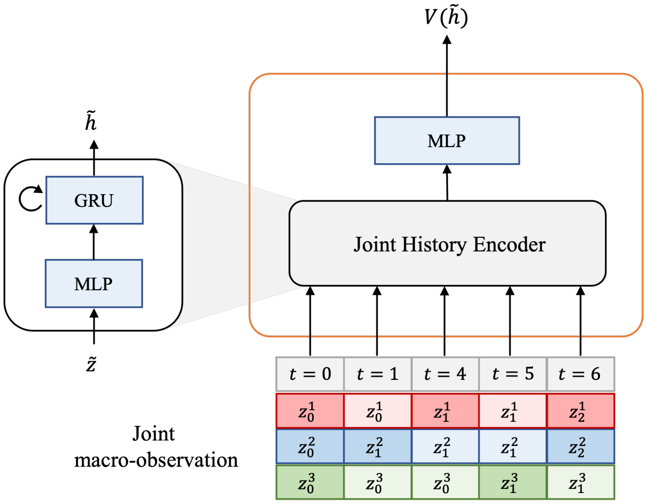


Figure 2: Centralized critic using joint history encoders at $t = 6$.

$[m_0^1, m_1^2, m_0^3]$. The next macro-action starts at timestep 4 from the agent 1, so the next joint-macro-action becomes $[m_1^1, m_1^2, m_0^3]$ and $\tilde{r}_k$ is $r_1 + r_2 + r_3$ if $\gamma = 1$. An example of how Mac-CERTs and Mac-JERTs are derived from a trajectory is illustrated in Figure 1. Note that Mac-CERTs contain squeezed trajectories per agent, where each agent's trajectory includes macro-observations and macro-actions collected only when the corresponding agent performs a new macro-action. In contrast, Mac-JERTs contain joint squeezed trajectories that include joint macro-observations and joint macro-actions collected only when any agent performs its macro-action.

Xiao et al. [17] presented two deep Q-networks (DQNs) [9] based approaches that learn macro-action-value functions in decentralized manner and centralized manner through Mac-CERTs and Mac-JERTs, respectively. Combining Mac-CERTs and Mac-JERTs, Xiao et al. [18] proposed DQN based approach for learning centralized training with decentralized execution (CTDE). Xiao et al. [19], closely related to our work, extended the previous macro-action based DQN methods to an actor-critic method for asynchronous MARL in CTDE setting, and used Mac-JERTs for centralized critic learning.

The previous works mainly focused on constructing a training buffer for asynchronous learning and simply adopted conventional MARL architectures originally designed for synchronous settings to learn value functions. Instead, we provide a novel architecture using agent-oriented encoders specifically for learning value functions in asynchronous settings.

3 METHOD

3.1 Motivation

As mentioned in the subsection 2.2, the previous work focused on the training buffer for MacDec-POMDP [19], rather than the architecture itself. The centralized critic of previous work focused on joint histories of all agents along the time axis and thus utilizes a joint history encoder to abstract joint macro-observations, defined as the concatenation of the most recent macro-observations of each agent. However, consecutive joint macro-observations may contain duplicated information from the local macro-observations at timesteps when one agent can start to perform a macro-action but other agents cannot, and the introduction of the duplicated local macro-observations can result in an inaccurate centralized critic.

To illustrate this, let us consider a trajectory example of MacDec-POMDP, as shown in Figure 1. In this example, the joint macro-observation history abstracted by the joint history encoder at timestep $t = 6$ is written as

$$\begin{aligned}\tilde{h} &= \text{Enc}\left(\tilde{z}_0, \tilde{z}_1, \tilde{z}_2, \tilde{z}_3, \tilde{z}_4\right) \qquad (2)\\ &= \text{Enc}\left(\begin{bmatrix} z_0^1 \\ z_0^2 \\ z_0^3 \end{bmatrix}, \begin{bmatrix} z_0^1 \\ z_1^2 \\ z_0^3 \end{bmatrix}, \begin{bmatrix} z_1^1 \\ z_1^2 \\ z_0^3 \end{bmatrix}, \begin{bmatrix} z_1^1 \\ z_1^2 \\ z_1^3 \end{bmatrix}, \begin{bmatrix} z_2^1 \\ z_2^2 \\ z_1^3 \end{bmatrix}\right),\end{aligned}$$

where $\tilde{z}_k$ represents the k-th joint macro-observation and z_k^i represents the k-th macro-observation of the i-th agent. This is also illustrated below in Figure 2. As demonstrated in this example, the conventional joint history encoder utilizes the same macro-observations multiple times (e.g., z_1^2 three times), which can hinder the accurate capture of both the local history of all agents and the reasoning among agents.

To address this issue, we propose a novel structure for a centralized critic in MacDec-POMDP, named the agent-oriented centralized critic (AOCC). The AOCC consists of two components: 1) Agent-oriented encoder with positional encoding, 2) Aggregation module for incorporating histories of all agents. Figure 3 illustrates the proposed structure, which offers advantages in extracting agent histories and identifying reasoning compared to the conventional structure. Detailed explanations of each component and their respective roles are provided in the following subsections.

3.2 Agent-Oriented Centralized Critic

The proposed agent-oriented structure has N agent-oriented encoders, each of which takes the latest local macro-observations of the corresponding agent as input and abstracts the history of the local macro-observations as output. It is noteworthy that this encoder avoids utilizing the same local macro-observations repeatedly to abstract its history, a departure from the centralized encoder in the previous structure. Consequently, the agent-oriented encoders focus solely on capturing the history of the corresponding agent, ensuring accurate capture of the local history. Following the previous work, we employ gate recurrent unit (GRU) [4] for the agent-oriented encoders without parameter sharing across agents. Then the history of the agent i is written as

$$h^i := \text{Enc}^i\left(z_0^i, \ldots, z_k^i\right), \qquad (3)$$

where Enc^i is an agent-oriented encoder of agent i and z_k^i is the latest local macro-observation.

However, the agent-oriented encoder breaks a temporal alignment between agents, as it only encodes the local macro-observations of each agent, not their joint observations. The recurrent model captures the order of the local macro-observation sequence within its

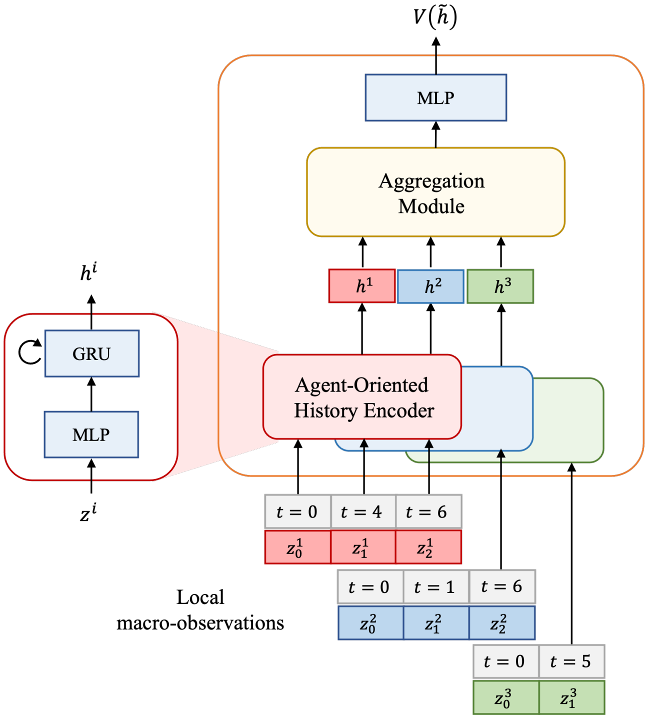


Figure 3: Agent-oriented centralized critic at $t = 6$.

agent, but consideration must also be given to the order of macro-observations between agents for appropriate critic learning. To address this critical issue, we inject time information by adopting the sinusoidal positional encoding [15] aimed at injecting temporal markers for the macro-observation of each agent.

We encode the timestep of a local macro-observation as a positional encoding vector p^i and concatenate it with the local macro-observation z^i. Consequently, the history of agent i can be rewritten, extending Equation 3, as follows:

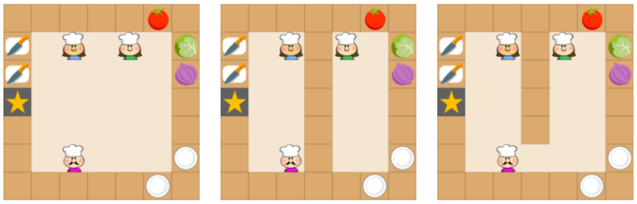
$$h^i := \mathrm{Enc}^i\left([z_0^i, p_0^i], \ldots, [z_k^i, p_k^i]\right), \tag{4}$$

where p_k^i is the positional encoding vector of the timestep when the macro-action z_k^i is performed. The positional encoding enables the centralized critic to encompass overall temporal information, ensuring a coherent understanding of the temporal ordering and duration of macro-actions across all agents.

The agent-oriented histories abstracted from the agent-oriented encoders are further processed by the aggregation module to approximate the value function. There are several options for the aggregation module, including attention networks or multi-layer perceptron (MLP) followed by concatenation or summation. However, for the sake of simplicity, we choose concatenation with MLP layers. In particular, the input of the aggregation module is the concatenation of the latest local histories, expressed as follows:

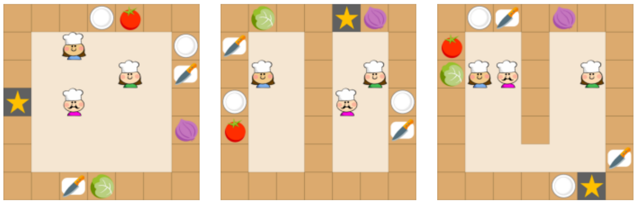
$$\tilde{h} := \left[h^1, \ldots, h^N\right] \tag{5}$$

Then, the subsequent MLP layers convert the joint history into its value $V(\tilde{h})$.



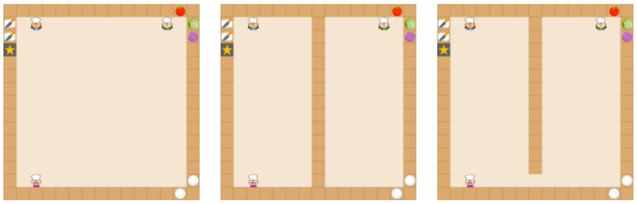
(a) Map A **(b) Map B** **(c) Map C**

Figure 4: The collection of the 7×7 Overcooked Environments.



(a) Map A **(b) Map B** **(c) Map C**

Figure 5: The collection of the randomized Overcooked Environments.



(a) Map A **(b) Map B** **(c) Map C**

Figure 6: The collection of the 15×15 Overcooked Environments.

3.3 Training Actor with Agent-Oriented Centralized Critic

Following the previous work [19], we also utilize actor-critic algorithm for CTDE with the vanilla policy gradient. To be specific, we train a parameterized value function $V_w(\cdot)$ as follows:

$$\mathcal{J}_V(w) = \mathbb{E}_{\Psi_\theta}\left[\left(y - V_w(\tilde{h})\right)^2\right], \tag{6}$$

$$\text{where } y = \tilde{r} + \gamma^{\tilde{\tau}} V_\theta(\tilde{h}') \tag{7}$$

The corresponding a parameterized policy Ψ_{θ_i} for agent i is optimized as follows:

$$\nabla_{\theta_i}\mathcal{J}_\Psi(\theta_i) = \mathbb{E}_{\Psi_{\theta_i}}[\nabla_{\theta_i} \log \Psi_{\theta_i}(m^i \mid h^i)\Phi], \tag{8}$$

$$\text{where } \Phi = \left(r^i + \gamma^{\tau^i} V_w(\tilde{h}') - V_w(\tilde{h})\right) \tag{9}$$

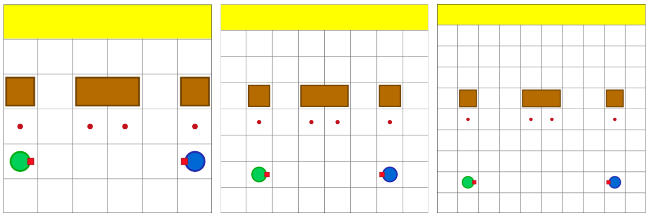

(a) 6 × 6 **(b)** 8 × 8 **(c)** 10 × 10

Figure 7: The collection of the BoxPushing Environments.

4 EXPERIMENT

4.1 Environments

We evaluate our method on two collections of environments for MacDec-POMDP; Overcooked and BoxPushing [18, 19].

Overcooked. Overcooked environment is derived from the Gym-Cooking environment [16]. The objective is for three agents to cook ingredients (e.g., tomato, onion) and deliver the prepared salad to the destination quickly. To be specific, all vegetables must be chopped, and the chopped vegetables must be placed on a single plate and delivered to the destination (a cell marked as yellow star). The environment has three maps of 7 × 7 grid (A, B, and C) as shown in Figure 4. Agents observe a 5 × 5 grid centered around themselves and select macro-actions, such as GO TO TOMATO, CHOP and DELIVER. Once an agent performs a macro-action, the macro-action terminates if the goal of the macro-action is achieved or the goal cannot be achieved in the current position of the agent. For example, in map A, once an agent performs GO TO TOMATO, then the macro-action terminates when the agent touches the tomato. On the other hand, in map B, which is separated into two, the agent in the upper-left side of the map cannot be move to tomato, so the macro-action GO TO TOMATO terminates at the timestep that the agent starts to perform the macro-action. The team receives a small bonus reward when they chop vegetables, and large reward for delivering the correct salad. They get penalty when deliver wrong salad, such as a salad without tomato, and additionally get tiny penalty for every timestep.

Moreover, we make the original Overcooked environments more complicated as shown in Figure 5 and Figure 6.

Overcooked-Rand. We randomize initial positions of objects and agents in each episode to give randomness and uncertainty in the environment, since agents in the original environment may solve a given task by simply memorizing the action sequence without considering observation. If the random positions are infeasible, the environment readjusts the positions.

Overcooked-Large. We increase the original grid size 7 × 7 to 15 × 15 where the duration of macro-action becomes longer, resulting in more frequent duplication of local macro-observations.

Overcooked-Large-Rand. For more challenging environments, we apply both randomization and up-scaling.

BoxPushing. We also evaluate our method on another benchmark for MacDec-POMDP, named BoxPushing. The objective of BoxPushing environment is for two agents, blue and green, to push big box together in the middle of the map to the yellow area. The environment has three maps with different grid size as shown in Figure 7. Each agent can observe one of five conditions, i.e., EMPTY, ANOTHER AGENT, BOUNDARY, SMALL BOX AND BIG BOX, of an unit cell in front of it, and can do TURN-LEFT, TURN-RIGHT, STAY, MOVE-TO-SMALL-BOX, MOVE-TO-BIG-BOX and PUSH. Each agent can push small box alone and receives a small reward when they push it to the yellow area. On the other hand, the big box only moves when both agents push it at the same time, and they get a large when they successfully push it to the yellow area. The team gets penalty when any agent hits the boundary or pushes the big box alone. In this respect, this environment necessitate the cooperation between two agents in that they should avoid attractive sub-optimality (pushing small boxes individually) and try to get optimality (pushing a big box together). Different from Xiao et al. [19], we do not allow additional access to the ground truth state (agents' poses and boxes' positions) in centralized critic learning.

4.2 Evaluation

We compare our proposed method, AOCC, with baseline methods using the joint history encoder, naive independent actor centralized critic (NIACC) and independent actor individual centralized critic (IAICC) [19] where the difference lies in whether to maintain a single centralized critic (NIACC) or have independent critics for each agent (IAICC). Both baselines utilize actor-critic based CTDE with vanilla policy gradient. Each experiment is run with five random seeds to report the mean and standard error of returns.

4.3 Performance Comparison

In fixed Overcooked environments (Figure 8), AOCC shows better stability in that it consistently reaches the optimality in all environments, while the baselines shows slower convergence. Yet, the final returns are similar due to the simplicity of the environment itself. In Overcooked-Rand, AOCC significantly outperforms the baselines in both sample efficiency and final return.

In Overcooked-Large (Figure 9) where the length of the macro action is longer so that duplicated macro-observations occurs more frequently, the performance gap between AOCC and other baselines becomes even larger except for Overcooked-Large-B and Overcooked-Large-Rand-B where all methods fails due to its difficulty.

As in Figure 10, all the baselines learn a sub-optimal policy at the initial stage in BoxPushing environments.[*] However, the baselines fail to reach the optimality except only IAICC in 6 × 6 grid and cannot escape from the sub-optimality. On the other hand, only AOCC succeed to reach optimality as learning progress.

These results show that our approach is more effective for MacDec-POMDP problems than previous approach, joint history representation learning. Due to the redundancy in joint history of NIACC and IAICC, we conjecture, both methods suffer a difficulty in learning accurate value function, leading to lower performance than AOCC, which verifies the effectiveness of agent-oriented centralized critic.

[*] We use local macro-observations for training agents, while Xiao et al. [19] allow access to ground truth state.

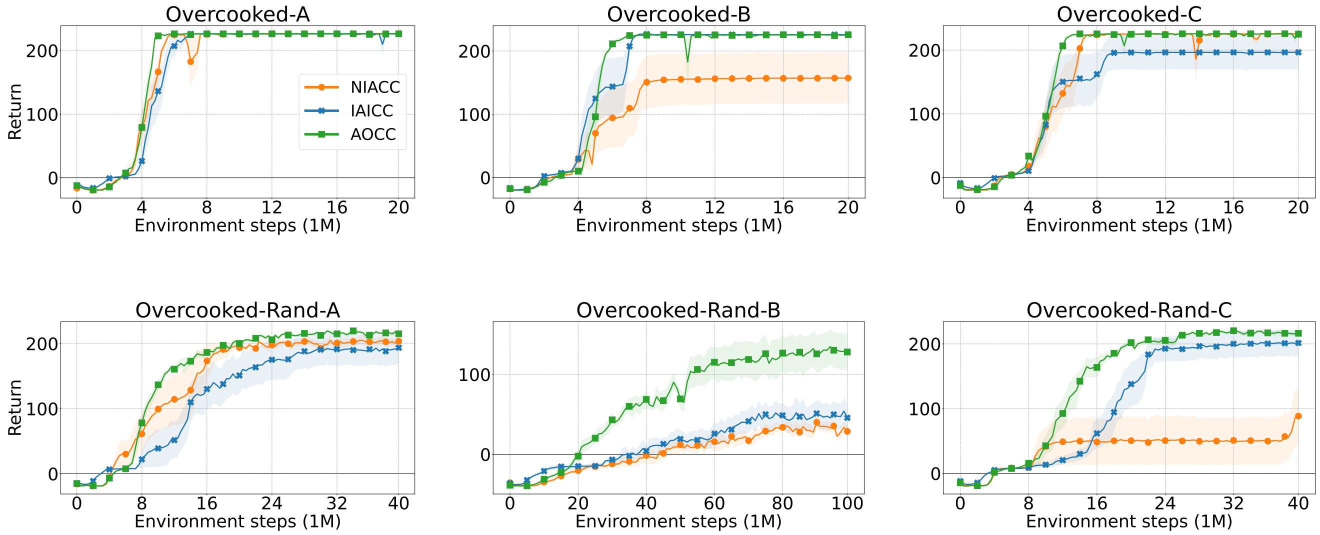


Figure 8: Training curves on Overcooked environments. The maximum training environment step depends on the difficulty of each environment.

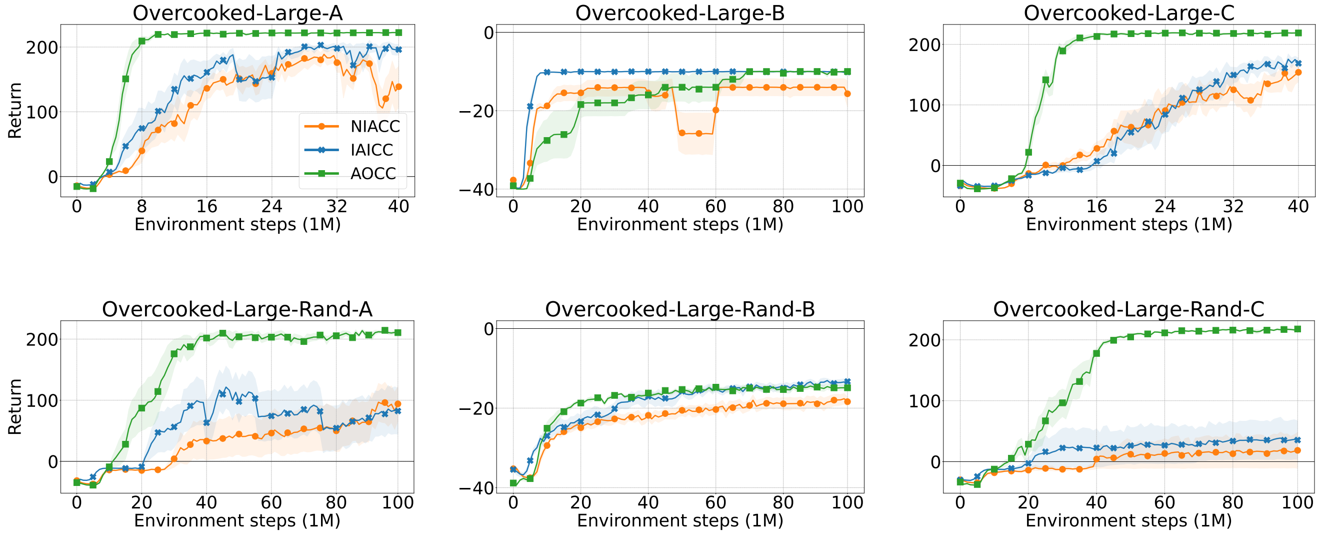


Figure 9: Training curves on Overcooked-Large environments. The maximum training environment step depends on the difficulty of each environment.

4.4 Ablation Study

To analyze the effect of removing the redundancy in joint history encoder, we evaluate ablated version of AOCC, referred to AOCC-dup, which has the same network architecture but the history consists of the duplicated macro-observations as in IAICC. To be specific, each agent encoder in AOCC-dup uses the squeezed trajectory of the corresponding agent in Mac-JERTs while that in AOCC uses the squeezed trajectory of the corresponding agent in Mac-CERTs. The experiment is conducted in the various Overcook-C environment, where the performance gap with other baselines is the largest.

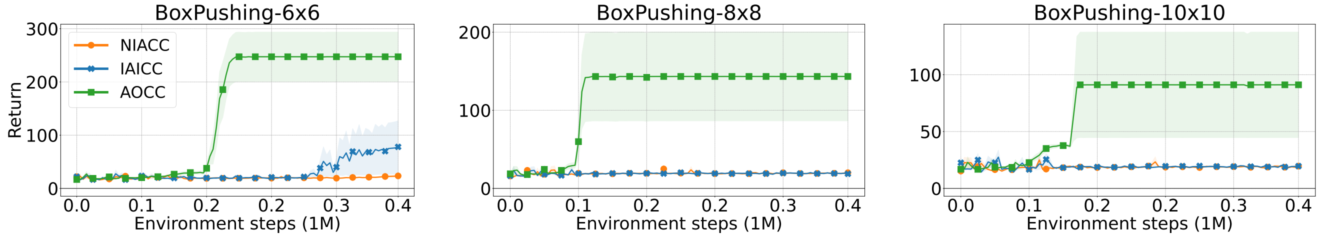


Figure 10: Training curves on BoxPushing environments.

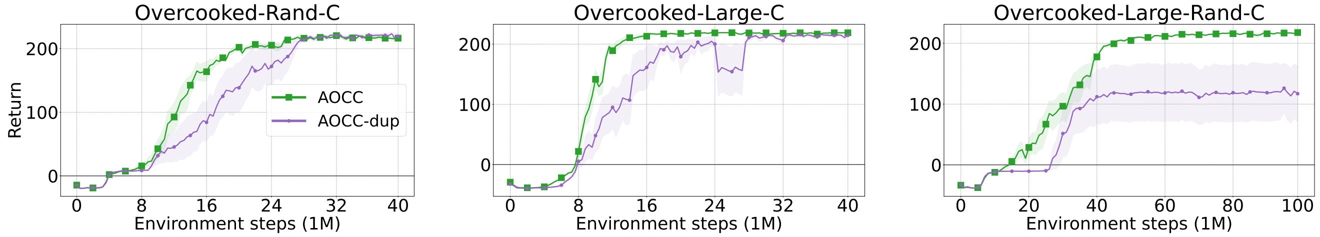


Figure 11: Ablation study on redundancy in history encoders.

As in Figure 11, both AOCC and AOCC-dup stably reach the optimal point although AOCC-dup is quite slower. However, as the grid size gets larger and complicated, AOCC-dup shows unstable learning while AOCC consistently converges to the optimal point. We hypothesize that during centralized critic learning, repetitive but unnecessary learning signals, stemming from the redundancy, interfere with the accurate learning of a value function.

5 CONCLUSION

In this work, we addressed the challenge of asynchronous multi-agent reinforcement learning by introducing the use of agent-oriented representations learning within the MacDec-POMDP framework. Our approach encodes the observation history of each agent independently with temporal information through positional encoding, and aggregate them explicitly, enabling efficacious centralized critic learning in asynchronous settings. Through experiments on a macro-action-based multi-agent benchmark, we demonstrated the superiority of our proposed method compared to conventional approaches, particularly in environments that require complex co-operation and generalization. Also, we studied the effect of redundancy from duplicated macro-observation history in critic learning, and showed reducing it brings better performance. For future work, further qualitative analysis on representation from agent-oriented encoder in the perspective of neural architecture would be helpful to understand AOCC and the reason of degradation in centralized critic learning with redundancy.

REFERENCES

[1] Christopher Amato, George Konidaris, Ariel Anders, Gabriel Cruz, Jonathan P How, and Leslie P Kaelbling. 2016. Policy search for multi-robot coordination under uncertainty. *The International Journal of Robotics Research* 35, 14 (2016), 1760–1778.

[2] Christopher Amato, George Konidaris, and Leslie P. Kaelbling. 2014. Planning with Macro-Actions in Decentralized POMDPs. In *2014 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*. 1273–1280.

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

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[5] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[6] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. 2022. Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=EcGGFkNTxdJ

[7] Miao Liu, Christopher Amato, Emily Anesta, John Griffith, and Jonathan How. 2016. Learning for decentralized control of multiagent systems in large, partially-observable stochastic environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[8] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, Vol. 30.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[10] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Christopher Amato, Shih-Yuan Liu, Jonathan P. How, and John Vian. 2017. Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions. *Int. J. Robotics Res.* 36, 2 (2017), 231–258. https://doi.org/10.1177/0278364917692864

[11] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Christopher Amato, Shih-Yuan Liu, Jonathan P How, and John Vian. 2017. Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions. *The International Journal of Robotics Research* 36, 2 (2017), 231–258.

[12] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning*. PMLR, 2681–2690.

[13] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*. PMLR, 4295–4304.

[14] Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.* 112, 1–2 (aug 1999), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

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

[17] Yuchen Xiao, Joshua Hoffman, and Christopher Amato. 2020. Macro-action-based deep multi-agent reinforcement learning. In *Conference on Robot Learning*. PMLR, 1146–1161.

[18] Yuchen Xiao, Joshua Hoffman, Tian Xia, and Christopher Amato. 2020. Learning multi-robot decentralized macro-action-based policies via a centralized Q-net. In *2020 IEEE International conference on robotics and automation (ICRA)*. IEEE, 10695–10701.

[19] Yuchen Xiao, Weihao Tan, and Christopher Amato. 2022. Asynchronous actor-critic for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*. https://openreview.net/forum?id=K_LtkDGdonK

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