Agent-Centric Actor-Critic for Asynchronous Multi-Agent Reinforcement Learning

Whiyoung Jung [*1] **Sunghoon Hong** [*1] **Deunsol Yoon** [*1] **Kanghoon Lee** [1] **Woohyung Lim** [1]

Abstract

Multi-Agent Reinforcement Learning (MARL) struggles with coordination in sparse reward environments. Macro-actions —sequences of actions executed as single decisions— facilitate long-term planning but introduce asynchrony, complicating Centralized Training with Decentralized Execution (CTDE). Existing CTDE methods use padding to handle asynchrony, risking misaligned asynchronous experiences and spurious correlations. We propose the Agent-Centric Actor-Critic (ACAC) algorithm to manage asynchrony without padding. ACAC uses agent-centric encoders for independent trajectory processing, with an attention-based aggregation module integrating these histories into a centralized critic for improved temporal abstractions. The proposed structure is trained via a PPO-based algorithm with a modified Generalized Advantage Estimation for asynchronous environments. Experiments show ACAC accelerates convergence and enhances performance over baselines in complex MARL tasks.

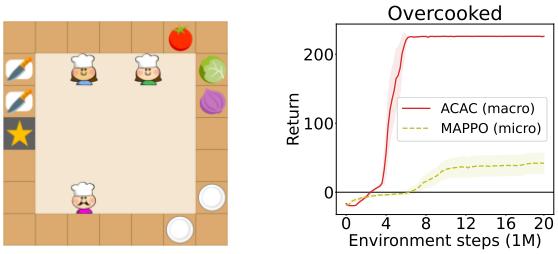


Figure 1. (Left) An example of Overcooked problem. (Right) Comparison of the learning efficiency when using micro-actions (dotted) versus macro-actions (solid).

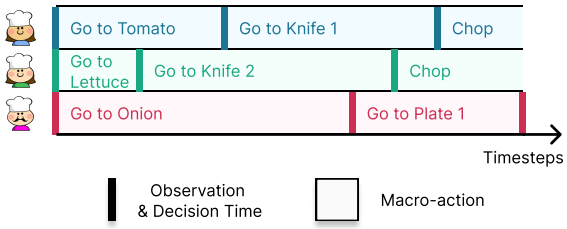


Figure 2. An example trajectory of asynchronous decisions.

1. Introduction

Multi-Agent Reinforcement Learning (MARL) has achieved significant progress (Omidshafiei et al., 2017; Foerster et al., 2018; Lowe et al., 2017; Rashid et al., 2018; Yu et al., 2022; Kuba et al., 2022), enabling distributed agents to learn coordinated strategies efficiently. Despite these advances, many real-world tasks remain notoriously challenging due to sparse rewards, where feedback signals are infrequent and less informative. In such environments, agents struggle to infer how their individual actions influence collective performance, often requiring extensive exploration.

One promising approach to address sparse rewards is to treat a well-defined sequence of actions[1] —referred to as a *macro-action*—as a single high-level decision (Xiao et al., 2020a;b). By selecting macro-actions, agents adopt long-term strategies that significantly enhance learning efficiency compared to relying solely on low-level (micro-)actions. An illustrative example of such an environment is the Overcooked problem, shown in the left side of Figure 1. In this setting, macro-actions like GO TO TOMATO each comprise a sequence of micro-actions aimed at completing a subtask. As depicted in the right side of Figure 1, using these macro-actions can significantly enhance learning efficiency.

However, employing macro-actions also causes the agents' action-selection timesteps to become asynchronous, as shown in Figure 2. In this figure, for instance, when the blue agent selects GO TO TOMATO, it spends four timesteps executing the macro-action before finally acquiring the tomato. After finishing a macro-action, each agent receives a new

[*]Equal contribution [1]LG AI Research, Seoul, Republic of Korea. Correspondence to: Woohyung Lim <w.lim@lgresearch.ai>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

[1]Macro-actions are not fixed sequences of micro-actions; see Section 2.3 for further details.

observation and chooses its next macro-action accordingly. Because macro-action durations vary across agents, the exact times when they complete their actions—and thus when they select new ones—differ as well. This variability in execution times underlies the asynchronous nature of the MARL environment.

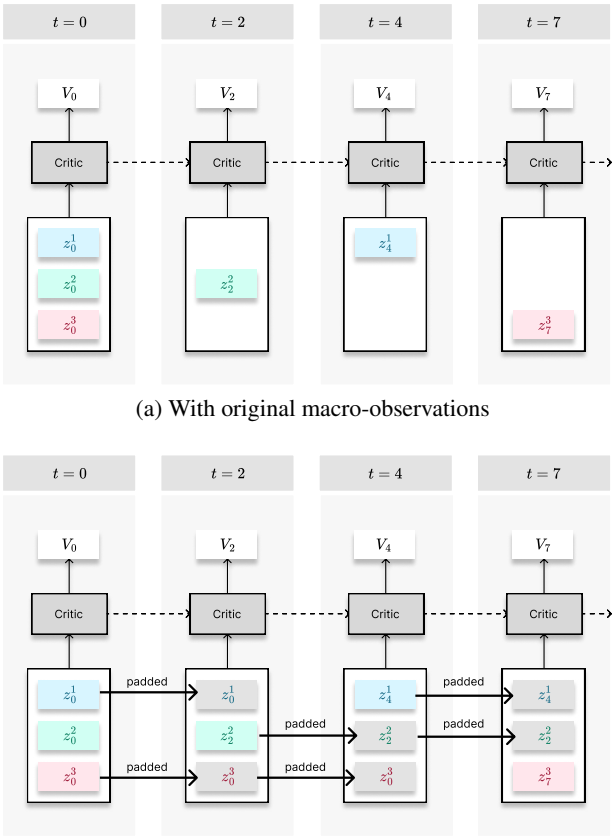
Centralized Training with Decentralized Execution (CTDE) is a widely adopted framework for efficient learning in synchronous MARL environments. While CTDE can also be considered to enhance learning efficiency in asynchronous MARL settings, its straightforward application becomes infeasible. In asynchronous environments, observation and macro-action information are available only for subsets of agents at any given timestep. Consequently, existing CTDE methods designed for synchronous MARL cannot be directly applied. To address this limitation, current approaches (Xiao et al., 2020a;b) typically pad the missing information for agents during centralized training using their most recent observations. However, as highlighted by (Liang et al., 2022), this padding technique can result in the abstraction of misleading information, potentially undermining the effectiveness of the learning process.

To address this challenge, we propose an *Agent-Centric Actor-Critic (ACAC)* algorithm. ACAC leverages *agent-centric history encoders* to process each agent's trajectory individually, followed by an attention-based mechanism that integrates these per-agent histories into a centralized critic. This design eliminates the redundancy inherent in padding-based methods, leading to more accurate temporal abstractions, faster learning, and higher-quality policies. To train the centralized critic and decentralized actors, we introduce a PPO-based algorithm incorporating a modified GAE suited for asynchronous settings. We evaluate ACAC on several macro-action-based multi-agent benchmarks, where experimental results show that it not only accelerates learning convergence under sparse rewards but also achieves higher final returns than conventional padding-based approaches.

2. Background

2.1. Synchronous Multi-Agent RL Settings

Cooperative Multi-Agent Reinforcement Learning is generally formulated as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Bernstein et al., 2002; Omidshafiei et al., 2015). A Dec-POMDP is formally defined as a tuple $\mathcal{D} = (\mathcal{I}, \mathcal{S}, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{R}, O, \gamma)$, where $\mathcal{I}$ represents the set of N agents, $\mathcal{S}$ denotes the environmental state space, and $\mathcal{A} = \times_{i\in\mathcal{I}}\mathcal{A}^i$ is the joint action space of the agents. $\Omega = \times_{i\in\mathcal{I}}\Omega^i$ is the joint observation space, $\mathcal{T}$ specifies the state transition probabilities, and $\mathcal{R}$ is the global reward function. O defines the observation function, and γ is the discount factor. In this synchronous setting, all agents act synchronously by selecting and executing their actions simultaneously at each discrete time step. The objective is to maximize a shared cumulative reward.



(a) With original macro-observations

(b) With padded macro-observations

Figure 3. Centralized critics for the example in Figure 2.

2.2. Asynchronous Multi-Agent RL Settings

In asynchronous settings, agents operate asynchronously by selecting and executing macro-actions that may vary in duration, enabling flexible and scalable coordination among agents. This framework can be modeled as a Macro-Action Decentralized Partially Observable Markov Decision Process (MacDec-POMDP) (Amato et al., 2014).

A MacDec-POMDP is defined as a tuple $\mathcal{D}' = (\mathcal{D}, \mathcal{M}, \zeta)$. Here, $\mathcal{M} = \times_{i\in\mathcal{I}}\mathcal{M}^i$ represents the joint macro-action space, where $\mathcal{M}^i$ is the macro-action space for each agent i, and $m^i \in \mathcal{M}^i$ denotes a macro-action chosen by agent i. Similarly, $\zeta = \times_{i\in\mathcal{I}}\zeta^i$ denotes the joint macro-observation space, where ζ^i is the macro-observation space for each agent i, $z^i \in \zeta^i$ represents the macro-observation of agent i, and $\tilde{z} \in \zeta$ is the aggregated joint macro-observation across all agents. For the full notation for the MacDec-POMDP can be found in Appendix B.

In a MacDec-POMDP, each agent selects a macro-action m^i

based on its current macro-observation z^i. Once a macro-action m^i is chosen, the sequence of micro-actions that constitute it is executed in order. Upon completion of these micro-actions, the macro-action m^i terminates. At this point, the agent receives both a new macro-observation z^i_{next} and select a new macro-action m^i_{next} based on the updated macro-observation z^i_{next}. This cycle continues until the episode concludes. The objective of MacDec-POMDP is identical to that of Dec-POMDP, but it incorporates macro-observations and macro-actions to facilitate higher-level decision-making and improve learning efficiency.

2.3. Macro-Actions in MacDec-POMDP

Within MacDec-POMDP, macro-actions are modeled as options in the Semi-Markov Decision Process framework (Sutton et al., 1999). Each macro-action is defined as $m = \langle \pi_m, I_m, \beta_m \rangle$, where π_m is an intra-option policy over micro-actions given a history of micro-observations, I_m is the initiation set from which the macro-action can start, and β_m is the termination condition, giving the probability that the macro-action ends given the current micro-observation history. The design of these intra-option policies and termination conditions enables inter-agent interaction and adaptation. This careful formulation allows cooperative behaviors to emerge naturally within the MacDec-POMDP framework.

2.4. Extending Sync. MARL to Async. MARL

Centralized Training with Decentralized Execution (CTDE) is a widely adopted framework for efficient learning in synchronous MARL environments. CTDE leverages a centralized critic to estimate the joint value function for all agents, facilitating coordinated policy development. While CTDE can also be considered to enhance learning efficiency in asynchronous MARL settings, its straightforward application becomes infeasible.

In asynchronous environments, macro-observation information is available only for subsets of agents at any given timestep. Consequently, existing CTDE methods designed for synchronous MARL cannot be directly applied because the centralized critic requires joint macro-observations of varying sizes, as illustrated in Figure 3 (a). To address this limitation, current approaches (Xiao et al., 2020a;b) typically pad the missing information for agents using their most recent macro-observations, as shown in Figure 3 (b). This padding strategy represents the simplest way to apply synchronous MARL methods to asynchronous CTDE without considering specialized algorithmic or network structure modifications. However, as highlighted by (Liang et al., 2022), this padding technique can lead to the abstraction of misleading information, potentially undermining the effectiveness of the learning process.

Specifically, the padding-based method does not distinguish whether the newly obtained information is identical to previous information or merely reuses past macro-observations. As a result, both the presence of new information and the duration information derived from previous data become inaccurate, thereby reducing the accuracy of history abstraction. This lack of distinction compromises the centralized critic's ability to accurately interpret the agents' historical macro-actions and macro-observations, further diminishing the overall learning effectiveness in asynchronous MARL environments.

3. Method

3.1. Review of a CTDE Framework for Async. MARL

Centralized Critic. A centralized critic in the CTDE framework is usually trained using the combined histories of all agents. At each timestep t, the observation history of the i-th agent is denoted as:

$$z^i_{\leq t} = \left\{ z^i_u \mid z^i_u \text{ is available }, u \leq t \right\},$$

where $z^i_{\leq t}$ represents the sequence of all observations of agent i up to timestep t. The critic takes as input the set of all agents' histories and predict the value at timestep t as:

$$V_\psi(\tilde{z}_{\leq t}), \text{ where } \tilde{z}_{\leq t} = \{z^1_{\leq t}, z^2_{\leq t}, \ldots, z^N_{\leq t}\}$$

In asynchronous settings, not all agents can obtain macro-observations z^i_t at every timestep t. To address this, missing macro-observations are replaced with the latest valid macro-observations, $z^i_{t,\text{latest}}$, when constructing the history:

$$z^i_{\leq t} = \begin{cases} \{z^i_{\leq t-1}, z^i_t\}, & \text{if } z^i_t \text{ is available,} \\ \{z^i_{\leq t-1}, z^i_{t,\text{latest}}\}, & \text{if } z^i_t \text{ is unavailable.} \end{cases}$$

This mechanism ensures that the input to the critic $\tilde{z}_{\leq t}$ remains consistent across timesteps, even when macro-observations are missing. By leveraging the latest valid macro-observations, the centralized critic adapts effectively to asynchronous environments while maintaining compatibility with the training process.

Decentralized Actor. In decentralized execution, each agent independently receives its local macro-observation and selects an macro-action based on its current local macro-observation and past history. Unlike the critic, there is no structural difference between synchronous and asynchronous settings, as each agent operates independently. The macro-action m^i_t for the i-th agent at timestep t is sampled as follows:

$$m^i_t \sim \pi^i_{\theta^i}(\cdot \mid z^i_{\leq t}),$$

Actor-Critic Algorithm for Asynchronous MARL. We employ an actor-critic algorithm to train decentralized actors, each parameterized by θ^i, alongside a centralized critic

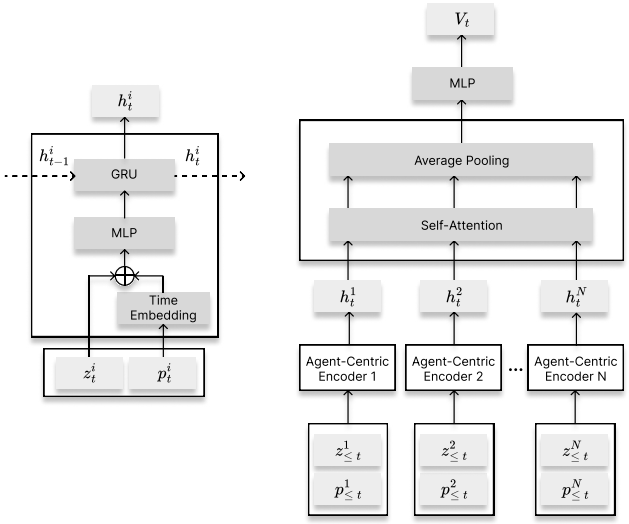


Figure 4. Overall structure of the agent-centric centralized critic: (Left) Agent-centric encoder, and (Right) Centralized critic integrating agent-centric histories.

parameterized by ψ. The training procedure is outlined as follows:

$$\mathcal{L}(\psi) = \mathbb{E}_\tau \left[\left(\left\{ \sum_{u=t}^{\infty} \gamma^{u-t} r_u \right\} - V_\psi(\tilde{z}_{\leq t}) \right)^2 \right]$$

$$\mathcal{L}(\theta^i) = -\mathbb{E}_\tau \left[\log \pi^i_{\theta^i} \left(m \mid z^i_{\leq t} \right) A_t \right]$$

where τ is sampled trajectories using the current actors and r_t denotes the reward at timestep t. The advantage function A_t is defined as:

$$A_t = \left\{ \sum_{u=t}^{t_{\text{next}}-1} \gamma^{u-t} r_u \right\} + \gamma^{t_{\text{next}}-t} V_\psi(\tilde{z}_{\leq t_{\text{next}}}) - V_\psi(\tilde{z}_{\leq t}).$$

where t_{next} is the next timestep at which a new macro-observation for any agent becomes available.

3.2. Agent-Centric Centralized Critic

In synchronous MARL, joint history abstraction is typically achieved by collecting joint observations at each timestep. However, we argue that this method does not align well with asynchronous MARL settings, where only a subset of agents may provide observations at any given timestep. Although padding can be employed to fill in missing observations, this approach often introduces redundant information and leads to inaccuracies in the joint history representation.

To address these issues, we propose a method that first constructs each agent's history on a per-agent basis and then integrates these agent-specific histories into a joint representation. Specifically, we introduce an agent-centric history encoder to capture each agent's macro-observation history.

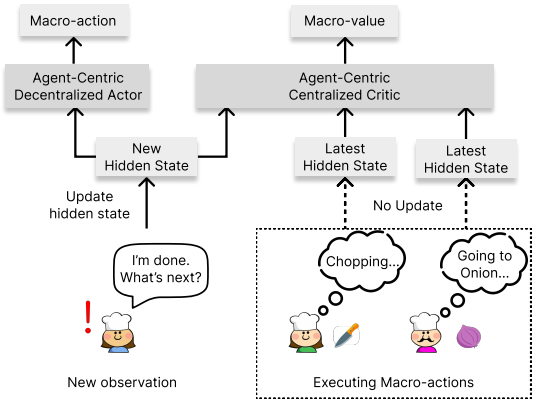


Figure 5. Illustration of the agent-centric actor-critic.

These agent-centric histories are then aggregated using an attention-based module, resulting in a more accurate joint history representation for value estimation. By focusing on encoding each agent's history before combining them, our approach effectively handles the lack of macro-observations typical in asynchronous MARL and mitigates the limitations of padding-based methods.

Agent-Centric Encoders. Each agent's history can be abstracted from its macro-observations using a recurrent network. However, in asynchronous MARL, it is essential to account for the time elapsed between consecutive macro-observations. Without this duration information, identical consecutive macro-observations lack context about whether the interval between them is one timestep or ten, which can lead to unstable or inaccurate learning of each agent's history due to varying durations.

To address this, we propose an agent-centric encoder that integrates both the macro-observation z^i_t and the corresponding timestep $p^i_t = t$ of each macro-observation. In the proposed encoder, the timestep information p^i_t is embedded using sinusoidal positional encoding (Vaswani et al., 2017). This embedded timestep is then concatenated with the macro-observation. The combined information is processed through a multilayer perceptron (MLP) and a gated recurrent unit (GRU) to effectively abstract each agent's history.

By incorporating timestep information p^i_t alongside macro-observation z^i_t, the agent-centric encoder ensures that each agent's history is learned accurately and consistently, even in asynchronous MARL settings. The left side of Figure 4 illustrates the structure of the agent-centric encoder.

Centralized Critic. The proposed agent-centric centralized critic consists of two main components: an agent-centric encoder that abstracts each agent's history and an attention-based aggregation module that combines these abstracted histories.

The components and overall structure of the agent-centric centralized critic are illustrated in Figure 4. The centralized critic at timestep t is computed as follows:

$$h_t^i = \text{Enc}_{\psi^i}\left(z_{\leq t}^i, p_{\leq t}^i\right)$$
$$\tilde{h}_t = \left\{h_t^1, h_t^2, \ldots, h_t^N\right\}$$
$$V_\psi(\tilde{h}_t) = f_\psi\left(\text{Aggr-Module}_\psi\left(\tilde{h}_t\right)\right)$$

where $p_{\leq t}^i = \left\{p_u^i \mid z_u^i \text{ is available}, u \leq t\right\}$ is the timestep information of macro-observations of agent i until timestep t.

Figure 5 provides an example of its input and output at timestep t of the agent-centric centralized critic and actor. In this example, the blue agent (on the left) can obtain a new macro-observation and is able to select a new macro-action. In contrast, the green (in the middle) and the red (on the right) agents cannot obtain a new macro-observation because their previous macro-action is still in progress. In such cases, agents that have acquired a new macro-observation input their macro-observations into their respective agent-centric encoders to generate updated histories. Conversely, agents that have not obtained a new macro-observation use their existing history as input to the aggregation module. This approach allows the centralized critic to accurately estimate the value by leveraging the most recent and relevant historical information from each agent, even in asynchronous MARL settings. We provide detailed comparison between padding-based methods and ACAC in Appendix C.

3.3. Agent-Centric Actor-Critic Algorithm

We introduce the Agent-Centric Actor-Critic (ACAC), a PPO-based MARL algorithm that utilizes an agent-centric centralized critic. In this section, we provide a comprehensive description of the ACAC algorithm.

First, leveraging the agent-centric encoder described previously, we construct each agent's actor, denoted as $\pi_{\theta^i}^i(\cdot \mid h_t^i)$. Each actor is trained using a loss function based on the estimated advantage, similar to the actor loss employed in MAPPO (Yu et al., 2022). The loss function is defined as:

$$\mathcal{L}(\theta^i) = \mathbb{E}_\tau\left[\min\left(\text{surr}_1, \text{surr}_2\right)\right]$$
$$\text{surr}_1 = \rho_t^i(\theta^i) A_t^\lambda$$
$$\text{surr}_2 = \text{clip}\left(\rho_t^i(\theta^i), 1-\epsilon, 1+\epsilon\right) A_t^\lambda$$

where the importance sampling ratio is given by

$$\rho_t^i(\theta^i) = \frac{\pi_{\theta^i}^i(m_t^i \mid h_t^i)}{\pi_{\theta_{\text{old}}^i}^i(m_t^i \mid h_t^i)}$$

representing the ratio of the current policy $\pi_{\theta^i}^i$ to the old policy $\pi_{\theta_{\text{old}}^i}^i$ for selecting macro-action m_t^i given the macro-observations and their corresponding timesteps up to timestep t. Here, A_t^λ denotes the advantage estimate at timestep t using Generalized Advantage Estimation (GAE) (Schulman et al., 2016) with a discounting factor λ, and ϵ is the clipping ratio.

For the centralized critic, the loss function is defined as:

$$\mathcal{L}(\psi) = \mathbb{E}_\tau\left[\left(\hat{R}_t - V_\psi(\tilde{h}_t)\right)^2\right]$$

where $\hat{R}_t$ is the return at timestep t computed by using GAE. We also adopt a value normalization technique, named PopArt (Hessel et al., 2019), which was used in MAPPO (Yu et al., 2022).

3.4. Generalized Advantage Estimation (GAE) for Asynchronous MARL

Generalized Advantage Estimation (GAE) (Schulman et al., 2016) is a common technique for estimating advantage functions, balancing the high variance of empirical returns with the high bias of temporal difference estimates through a hyperparameter λ. Specifically, $\lambda = 0$ relies on single-step temporal difference (TD) estimates, while $\lambda = 1$ uses the full empirical return minus the baseline. Intermediate values of λ provide a controlled balance between these extremes.

In the original GAE, λ-discounting is typically applied at the micro-timestep level, reflecting the fixed intervals between micro-actions in synchronous environments. However, applying this standard formulation directly to asynchronous MARL environments, particularly those employing temporal abstraction via macro-actions (e.g., in a MacDec-POMDP setting), presents a significant challenge. The intervals between consecutive macro-observations can vary substantially. This makes traditional micro-timestep-based discounting unsuitable because it doesn't align with the varying durations of macro-actions. Specifically, as a macro-action becomes longer, this discounting method applies a greater discount to its future rewards, thereby diminishing the perceived importance of that macro-action decision.

Proposed GAE for Asynchronous Settings. To address the challenges of asynchronous environments, we propose a modified GAE approach. Our key insight lies in shifting the λ-discounting from the *micro-timestep*[2] level to the *macro-timestep* level. This means that future TD errors are discounted based on the number of macro-action decision steps taken, rather than the timesteps elapsed.

Consider an example where macro-observations (and consequently, value estimates) are obtained at micro-timesteps

[2]To clarify the distinction, throughout the remainder of this section, we use *micro-timesteps* interchangeably with *timesteps*. A *macro-timestep*, on the other hand, is defined as the count of micro-timesteps at which any agent obtains a new macro-observation.

$l_{(0)}, l_{(1)}, l_{(2)}, l_{(3)}, \ldots$ (e.g., 0, 2, 5, 6, ...). Here the index in the parenthesis, k, represents macro-timestep corresponding to the micro-timestep $l_{(k)}$. The multi-step TD errors at these micro-steps are defined as:

$$\begin{aligned}
\delta_{l_{(0)}} &= (r_0 + \gamma r_1) + \gamma^2 V_{l_{(1)}} - V_{l_{(0)}}, \\
\delta_{l_{(1)}} &= \left(r_2 + \gamma r_3 + \gamma^2 r_4\right) + \gamma^3 V_{l_{(2)}} - V_{l_{(1)}}, \\
\delta_{l_{(2)}} &= (r_5) + \gamma V_{l_{(3)}} - V_{l_{(2)}}, \\
&\vdots \\
\delta_{l_{(k)}} &= \Big(\sum_{t=l_{(k)}}^{l_{(k+1)}-1} \gamma^{t-l_{(k)}} r_t\Big) + \gamma V_{l_{(k+1)}} - V_{l_{(k)}}
\end{aligned}$$

Given these TD errors, we have two primary choices for designing the advantage function, differing in their application of λ-discounting:

- Micro-level discounting: Discounts future TD errors based on the actual micro-timesteps elapsed between decision points. This is standard in many GAE implementations.

  $$\begin{aligned}
  A^{\lambda,\text{micro}}_{l_{(0)}} &:= \delta_{l_{(0)}} + \lambda^{(l_{(1)}-l_{(0)})}\gamma^{(l_{(1)}-l_{(0)})}\delta_{l_{(1)}} \\
  &\quad + \lambda^{(l_{(2)}-l_{(0)})}\gamma^{(l_{(2)}-l_{(0)})}\delta_{l_{(2)}} + \ldots
  \end{aligned}$$

  This can be generally written as:

  $$A^{\lambda,\text{micro}}_{l_{(0)}} = \sum_{k=0}^{\infty} (\lambda\gamma)^{(l_{(k)}-l_{(0)})}\delta_{l_{(k)}}$$

- Macro-level discounting: Discounts future TD errors based on the sequence of macro-action decision points. Each transition between decision points (e.g., $l_{(0)}$ to $l_{(1)}$) counts as a single macro-timestep, regardless of the actual micro-timesteps.

  $$\begin{aligned}
  A^{\lambda,\text{macro}}_{l_{(0)}} &:= \delta_{l_{(0)}} + \lambda\gamma^{(l_{(1)}-l_{(0)})}\delta_{l_{(1)}} \\
  &\quad + \lambda^2\gamma^{(l_{(2)}-l_{(0)})}\delta_{l_{(2)}} + \ldots
  \end{aligned}$$

  This can be generally written as:

  $$A^{\lambda,\text{macro}}_{l_{(0)}} = \sum_{k=0}^{\infty} \lambda^k\gamma^{(l_{(k)}-l_{(0)})}\delta_{l_{(k)}}$$

The key distinction lies in the exponent of λ: it is the macro-timestep k for macro-level discounting, whereas for micro-level discounting, it is the micro-timestep $l_{(k)}$.

For the ACAC algorithm, we adopt the *macro-level discounting* approach. This choice was driven by its ability to effectively handle the variable intervals inherent in our asynchronous setting. By aligning the discounting with the critical decision points associated with macro-actions, this method emphasizes both the significance of future rewards and the strategic importance of each macro-action choice. This approach directly contributed to the strong performance observed with ACAC in the asynchronous MacDec-POMDP environment.

This formulation preserves the key properties of the original GAE: when $\lambda = 0$, it reduces to the single-step temporal difference at $l_{(0)}$, and when $\lambda = 1$, it becomes the empirical return minus the baseline. Hence, λ continues to govern the bias–variance trade-off in advantage estimation as the original GAE does. For a more detailed derivation, please refer to Appendix D.

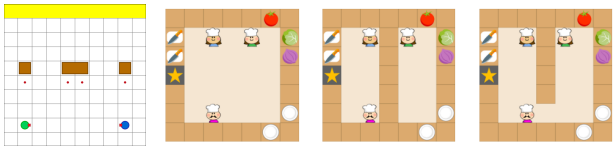
4. Related Work

Macro-action-based MARL, or Hierarchical MARL, has been actively studied in the field of multi-agent systems (Nachum et al., 2019; Yang et al., 2020; Wang et al., 2021a;b). Most works focus on learning macro-actions (options or skills) and typically enforce fixed durations, operating in synchronous settings. In contrast, our work addresses scenarios with pre-defined macro-actions of varying durations, focusing on training high-level policies for asynchronous MARL, thereby broadening MARL's applicability to more realistic environments.

Asynchronous MARL has recently gained attention, with several methods proposed to address specific tasks such as cooperative charging in sensor networks (Chen et al., 2021; Liang et al., 2022), bus holding control (Wang & Sun, 2021), warehouses logistics (Yoshitake & Abbeel, 2023; Krnjaic et al., 2024), petrochemical scheduling optimization (Hong et al., 2024). The studies most closely related to our work aim to address the more general problem of asynchronous MARL by defining it within the framework of MacDec-POMDPs (Amato et al., 2014) and conducting research based on this formulation.

Xiao et al. (2020a) presented two deep Q-networks (DQNs) (Mnih et al., 2013) based approaches that learn macro-action-value functions in decentralized manner and centralized manner. Xiao et al. (2020b) proposed DQN based approach for learning in CTDE setting. Xiao et al. (2022) extended the previous macro-action based DQN methods to an actor-critic method for asynchronous MARL. Yu et al. (2023) modeled multi-robot cooperative exploration, where agents can communicate with each other while performing actions, as an asynchronous MARL problem and applied MAPPO to address it.

Previous works have applied conventional actor-critic architectures from synchronous MARL by padding missing observations, which often introduces redundancy and inaccuracies in joint history representation. In contrast, we propose an agent-centric history encoding approach that



(a) BoxPushing (b) Map A (c) Map B (d) Map C

Figure 6. The collection of environments. (a) BoxPushing environment with grid size 10, and (b-d) Overcooked environments with different maps.

aggregates individual agent histories through an attention-based module, providing a more accurate joint history representation without relying on padding.

5. Experiment

We evaluate our method on two collections of MacDec-POMDP environments: BoxPushing and Overcooked (Xiao et al., 2020b; 2022).

BoxPushing. The BoxPushing environment, a benchmark for MacDec-POMDP, involves two agents (blue and green) cooperating to push a large box to a yellow goal area. It features three maps of varying sizes, including a 10×10 grid shown in Figure 6 (a). Agents observe one of five possible conditions for the cell ahead (e.g., EMPTY, BOUNDARY, BIG BOX) and can take actions like PUSH, MOVE-TO-BOX, or TURN.[3] While a small box can be pushed alone, pushing a large box requires both agents; otherwise, attempting to push it alone incurs a penalty. Agents receive a small reward for delivering a small box to the goal and a large reward for a large box. The episode ends as soon as any box reaches the goal, emphasizing the need for coordination to maximize rewards and avoid suboptimal strategies.

Overcooked. Adapted from Gym-Cooking (Wu et al., 2021), Overcooked environment involves three agents working together to prepare and deliver salads (e.g., tomato, onion). Vegetables must be chopped, plated, and delivered to a designated location (marked by a yellow star). The environment consists of three 7×7 grid maps (A, B, C), as shown in Figure 6 (b-d). Agents observe a 5×5 grid around themselves and execute macro-actions like GO TO TOMATO, CHOP, and DELIVER. Macro-actions terminate upon completion or if the goal is unreachable (e.g., in map B, separated regions prevent reaching a tomato). Agents receive rewards for chopping and delivering correct salads but incur penalties for incorrect deliveries and time delays. They are required to collaborate swiftly and efficiently to successfully prepare and deliver the salad.

[3]We use local macro-observations for training agents, while Xiao et al. (2022) allow access to the ground truth state.

Overcooked-Rand. Since the original Overcooked environments have no randomness and only limited uncertainty, we introduce a more complex setting where object and agent positions are randomized each episode. This introduces randomness and significantly increases uncertainty, preventing agents from memorizing a fixed action sequence and allowing evaluation of their generalization ability. To ensure all configurations remain solvable, we adjust any random placements that would make the scenario infeasible.

5.1. Evaluation

We compare our proposed method, **`ACAC`**, with two padding-based baselines: naive independent actor centralized critic (**`Mac-NIACC`**) and independent actor individual centralized critic (**`Mac-IAICC`**) (Xiao et al., 2022). The main distinction between these baselines is whether they use a single centralized critic (Mac-NIACC) or a individual critic for each agent (Mac-IAICC). Both methods employ an actor-critic-based CTDE framework with policy gradients. We also evaluate **`ACAC-Vanilla`**, a variant of ACAC that shares the same architecture but does not use the PPO-style loss function, instead adopting the same loss function as the two baselines.

The baselines labeled with the suffix **`micro`** are evaluated in environments without macro-actions (i.e., standard synchronous MARL). $\texttt{MAPPO}_{\texttt{micro}}$ denotes the original MAPPO algorithm applied to the micro-action-based environment. Each experiment was run with five random seeds, and we report the mean and standard error of returns.[4] The detailed hyperparameters can be found in Appendix F.

5.2. Comparison with Baselines

Figure 7 shows that ACAC effectively handles various environments, including BoxPushing, Overcooked, and Overcooked-Rand.

Micro-action baselines (labeled with **`micro`**) generally fail to reach optimal performance, likely due to the sparse reward structure requiring longer-term planning—an aspect better handled by macro-actions. Among these micro-action baselines, $\texttt{ACAC}_{\texttt{micro}}$ and $\texttt{MAPPO}_{\texttt{micro}}$, both PPO-based, perform relatively well in Overcooked, emphasizing the importance of the underlying PPO algorithm.

Compared to other macro-action baselines, ACAC converges more quickly and more reliably to optimal performance in all environments. For instance, in Overcooked, ACAC surpasses other baselines both in the speed of convergence and the stability of returns. This improvement stems from ACAC's novel architectural design, which avoids padding-related issues. Even in the more challenging

[4]We report undiscounted returns, whereas Xiao et al. (2022) reported discounted returns.

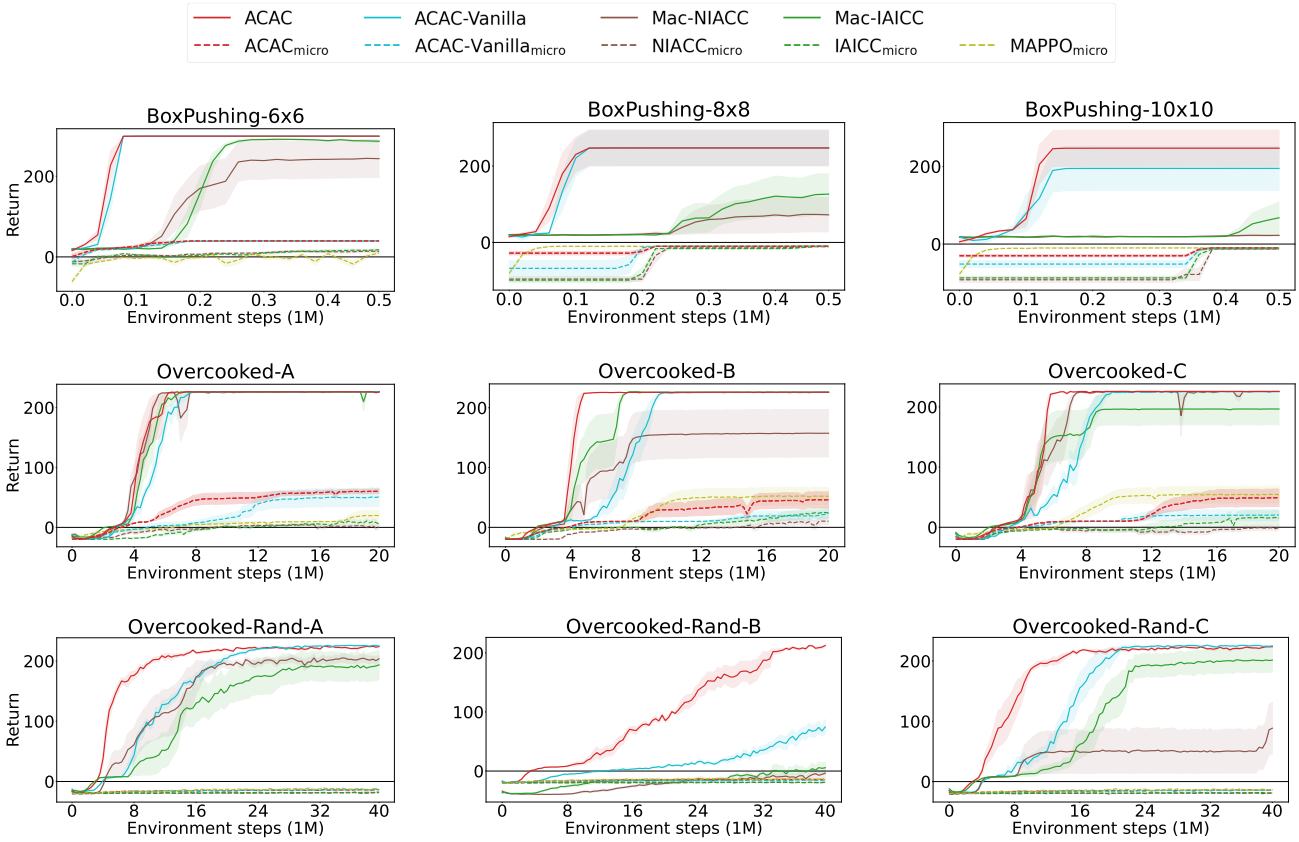


Figure 7. Training curves on (upper) BoxPushing, (middle) Overcooked, and (lower) Overcooked-Rand environments. The maximum training environment step depends on the difficulty of each environment.

Overcooked-Rand, ACAC maintains superior performance, underscoring its generalization capabilities.

ACAC-Vanilla outperforms most baselines, as its architecture effectively mitigates the padding issues in Mac-NIACC and Mac-IAICC. However, it trains slower than ACAC, highlighting the benefits of a PPO-based objective.

These overall results indicate that our approach more effectively handles history abstraction and structural reasoning across agent histories in MacDec-POMDP problems. We hypothesize that the padding in Mac-NIACC and Mac-IAICC complicates value function learning, reducing their overall performance compared to ACAC.

5.3. Further Comparison in More Challenging Scenario

To conduct a more rigorous evaluation, we introduce **Overcooked-Large**, a substantially harder environment. We increase the map size from 7×7 to 11×11, raise the number of agents from 3 to 6, and increase the number of tools (e.g., knives and plates) from 2 to 4, thus expanding both the joint macro-observation and macro-action spaces. These changes require significantly more complex cooperation. Since Overcooked-Large requires much longer planning horizons, most micro-action-based approaches fail to learn effectively. As a result, we limited our comparison here to macro-action-based models.

As shown in Figure 8, the performance gap between ACAC and other baselines is even more pronounced in Overcooked-Large. In nearly all tasks, ACAC achieves the optimal return, while the existing baselines often fail to learn effectively. These difficulties are likely attributable to prolonged macro-actions that require extensive padding in conventional architectures, making it harder to learn reliable value estimates.

5.4. Ablation Studies

Effect of Padding in History Encoders. To further assess the impact of eliminating padding in the joint history encoder, we introduce **`ACAC-Duplicate`**, an ablated version of ACAC that duplicates macro-observations as in Mac-IAICC. More specifically, ACAC-Duplicate retains the same overall architecture and algorithm as ACAC but differs in how it updates the agent-centric history encoder at each timestep. In ACAC, only newly received macro-observations are used to update an agent's encoder

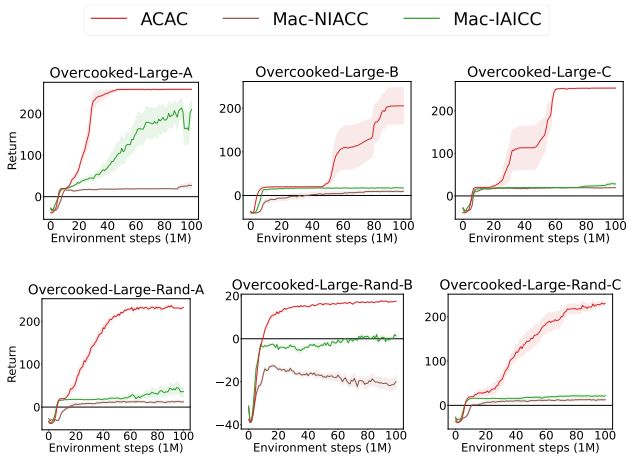


Figure 8. Training curves on Overcooked-Large environments.

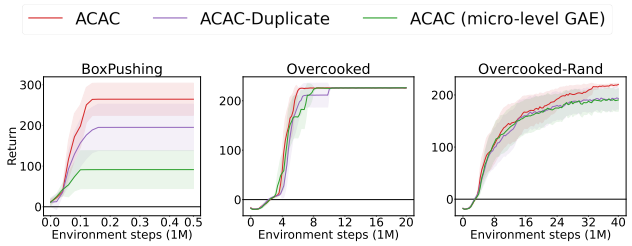


Figure 9. An ablation study examining the effects of removing padding and implementing macro-level λ-discounting in GAE. We report average return over all environments for each collections.

for value computation. By contrast, ACAC-Duplicate updates every agent's encoder at each timestep using that agent's most recent macro-observation—whether newly received or padded from a previous timestep—thereby duplicating macro-observations in the learning process. This setup allows us to directly compare how the presence (ACAC-Duplicate) or absence (ACAC) of duplicate macro-observations influences both learning speed and final performance.

In Figure 9, ACAC consistently achieves optimal performance, whereas ACAC-Duplicate converges more slowly or even settles into suboptimal solutions. We hypothesize that repetitive and potentially confusing training signals resulting from padded macro-observations hinder the learning of an accurate value function, highlighting the advantage of ACAC's design, which avoids such padding.

Effect of Macro-level λ-discounting in GAE. We now examine the choice of λ-discounting in GAE within asynchronous MARL. We compare the performance of GAE using the micro-level λ-discounting (referred to as **`ACAC (micro-level GAE)`**) against our proposed macro-level λ-discounting.

As Figure 9 illustrates, our results consistently show that macro-level discounting in GAE significantly outperforms its micro-level counterpart across various environments, including BoxPushing, Overcooked, and Overcooked-Rand. We believe this is because macro-level discounting better emphasizes the value of future rewards and the strategic importance of each macro-action choice. This strongly suggests that macro-level λ-discounting is a superior approach for GAE in asynchronous MARL.

For a complete set of ablation studies and additional results, please see Appendix E.

6. Conclusion

In this work, we introduced the *Agent-Centric Actor-Critic (ACAC)* algorithm to address the challenges of asynchronous multi-agent reinforcement learning (MARL) within the Centralized Training with Decentralized Execution framework. ACAC's architecture integrates two key components: 1) an *agent-centric history encoder* that captures each agent's macro-observation history along with the specific timestep at which their macro-action begins, and 2) an *aggregation module* that integrates these individual, agent-specific histories into a centralized critic. This design allows ACAC to effectively address the inefficiencies inherent in padding-based methods. To train this novel architecture, we further proposed a PPO-based algorithm and adapted the standard Generalized Advantage Estimation (GAE)—originally conceived for synchronous settings—to handle the varying durations of macro-actions in asynchronous MARL.

Our experimental results on macro-action-based multi-agent benchmarks demonstrate that ACAC consistently outperforms existing approaches, proving particularly effective in scenarios that demand complex coordination and robust generalization. By directly addressing padding-related issues through its agent-centric design and a PPO-based objective, ACAC achieves both accelerated training convergence and superior overall performance. Moreover, comprehensive ablation studies confirmed that eliminating duplicated macro-observations and using the modified GAE is crucial for preventing suboptimal outcomes, thereby underscoring ACAC's efficacy in asynchronous multi-agent environments.

While ACAC excels in macro-action-based MARL, extending its capabilities to continuous action spaces is a key future direction. We believe ACAC is adaptable for continuous control. However, a primary challenge is the current lack of asynchronous MARL benchmarks that combine continuous actions with defined macro-actions. Developing such benchmarks is crucial future work. Successfully adapting ACAC for these settings would significantly broaden its applicability in dynamic fields like robotics and autonomous driving, enhancing its versatility.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Amato, C., Konidaris, G., and Kaelbling, L. P. Planning with macro-actions in decentralized POMDPs. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1273–1280, 2014.

Amato, C., Konidaris, G., Kaelbling, L. P., and How, J. P. Modeling and planning with macro-actions in decentralized POMDPs. *Journal of Artificial Intelligence Research*, 64:817–859, 2019.

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

Chen, Y., Wu, H., Liang, Y., and Lai, G. VarLenMARL: A framework of variable-length time-step multi-agent reinforcement learning for cooperative charging in sensor networks. In *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, 2021.

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

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and Van Hasselt, H. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3796–3803, 2019.

Hong, S., Yoon, D., Jung, W., Lee, J., Yoo, H., Ham, J., Jung, S., Moon, C., Jung, Y., Lee, K., Lim, W., Jeon, S., Lee, M., Hong, S., Lee, J., Jang, H., Kwak, C., Park, J., Kang, C., and Kim, J. Naphtha cracking center scheduling optimization using multi-agent reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2024.

Krnjaic, A., Steleac, R. D., Thomas, J. D., Papoudakis, G., Schäfer, L., To, A. W., Lao, K., Cubuktepe, M., Haley, M., Börsting, P., and Albrecht, S. V. Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 677–684, 2024.

Kuba, J. G., Chen, R., Wen, M., Wen, Y., Sun, F., Wang, J., and Yang, Y. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Liang, Y., Wu, H., and Wang, H. ASM-PPO: Asynchronous and scalable multi-agent PPO for cooperative charging. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 798–806, 2022.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Nachum, O., Ahn, M., Ponte, H., Gu, S. S., and Kumar, V. Multi-agent manipulation via locomotion using hierarchical Sim2Real. In *Conference on Robot Learning (CORL)*, pp. 110–121, 2019.

Omidshafiei, S., Agha-Mohammadi, A.-A., Amato, C., and How, J. P. Decentralized control of partially observable markov decision processes using belief space macro-actions. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5962–5969. IEEE, 2015.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., and Vian, J. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning (ICML)*, pp. 2681–2690. PMLR, 2017.

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 4295–4304. PMLR, 2018.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sutton, R. S., Precup, D., and Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2): 181–211, 1999.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Wang, J. and Sun, L. Reducing bus bunching with asynchronous multi-agent reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 426–433, 2021.

Wang, R., Kew, J. C., Lee, D., Lee, T.-W., Zhang, T., Ichter, B., Tan, J., and Faust, A. Model-based reinforcement learning for decentralized multiagent rendezvous. In *Conference on Robot Learning (CORL)*, pp. 711–725, 2021a.

Wang, T., Gupta, T., Mahajan, A., Peng, B., Whiteson, S., and Zhang, C. RODE: Learning roles to decompose multi-agent tasks. In *International Conference on Learning Representations (ICLR)*, 2021b.

Wu, S. A., Wang, R. E., Evans, J. A., Tenenbaum, J. B., Parkes, D. C., and Kleiman-Weiner, M. Too many cooks: Coordinating multi-agent collaboration through inverse planning. *Topics in Cognitive Science*, 2021.

Xiao, Y., Hoffman, J., and Amato, C. Macro-action-based deep multi-agent reinforcement learning. In *Conference on Robot Learning (CORL)*, pp. 1146–1161, 2020a.

Xiao, Y., Hoffman, J., Xia, T., and Amato, C. Learning multi-robot decentralized macro-action-based policies via a centralized Q-net. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10695–10701, 2020b.

Xiao, Y., Tan, W., and Amato, C. Asynchronous actor-critic for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

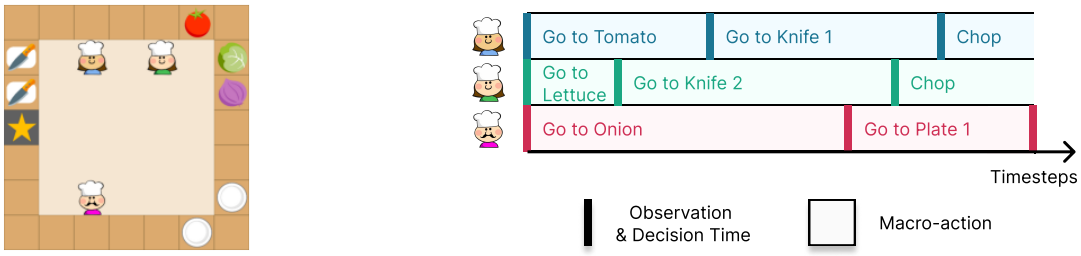
Yang, J., Borovikov, I., and Zha, H. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 1566–1574, 2020.

Yoshitake, H. and Abbeel, P. The impact of overall optimization on warehouse automation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1621–1628, 2023.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, pp. 24611–24624, 2022.

Yu, C., Yang, X., Gao, J., Chen, J., Li, Y., Liu, J., Xiang, Y., Huang, R., Yang, H., Wu, Y., and Wang, Y. Asynchronous multi-agent reinforcement learning for efficient real-time multi-robot cooperative exploration. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1107–1115, 2023.

A. Detailed Explanation of the Overcooked Example in Figure 2



(a) Overcooked Problem

(b) Asynchronous Decisions with Macro-Actions

Figure 10. (a) Example of an Overcooked problem. (b) Example trajectory of asynchronous decisions.

This section provides a detailed explanation of Figure 2 introduced in the main text. The same figure is reproduced here as Figure 10.

Figure 10 (a) illustrates an Overcooked environment where three agents collaborate to efficiently prepare and deliver salads (e.g., tomato, onion). In this environment, tasks such as chopping vegetables, plating them, and delivering them to a designated delivery point (yellow star) need to be performed. The agents' macro-actions include GO TO TOMATO, CHOP, DELIVER, and more. Once a macro-action is selected, the agent executes a sequence of micro-actions (e.g., navigating along a path) to reach the target location and complete the assigned task.

Figure 10 (b) demonstrates an example of agent behavior using macro-actions, with their actions depicted across different timesteps. Below is a step-by-step breakdown of the agents' behavior:

- Timestep 0:
  Each agent selects a macro-action based on its current macro-observation:
  - Agent 1 selects GO TO TOMATO.
  - Agent 2 selects GO TO LETTUCE.
  - Agent 3 selects GO TO ONION.
- Timestep 2:
  Agent 2 completes its GO TO LETTUCE macro-action and receives a new macro-observation. Based on this, Agent 2 selects a new macro-action: GO TO KNIFE 2.
- Timestep 4:
  Agent 1 completes its GO TO TOMATO macro-action and receives a new macro-observation. Based on this, Agent 1 selects a new macro-action: GO TO KNIFE 1.
- Timestep 7:
  Agent 3 completes its GO TO ONION macro-action and receives a new macro-observation. Based on this, Agent 3 selects a new macro-action: GO TO PLATE 1.
- Timestep 8:
  Agent 2 completes its GO TO KNIFE 2 macro-action and receives a new macro-observation. Based on this, Agent 2 selects a new macro-action: CHOP.
- Timestep 9:
  Agent 1 completes its GO TO KNIFE 1 macro-action and receives a new macro-observation. Based on this, Agent 1 selects a new macro-action: CHOP.
- Timestep 11:
  Agent 3 completes its GO TO PLATE 1 macro-action and receives a new macro-observation. Based on this, Agent 3 selects its next macro-action.

This process demonstrates the asynchronous decision-making dynamics in the Overcooked environment.

B. Formal Definition of MacDec-POMDP

The Macro-Action Decentralized Partially Observable Markov Decision Process (MacDec-POMDP) (Amato et al., 2019) incorporates the options framework (Sutton et al., 1999) into the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) by defining a set of macro-actions (or options) for each agent.

Followed by the previous work (Xiao et al., 2022), a MacDec-POMDP is represented as a tuple $\langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{M}, \Omega, \zeta, \mathcal{T}, \mathcal{R}, O, Z, \gamma \rangle$, where $\mathcal{I} = \{1, \ldots, N\}$ is a set of indices of agents, $\mathcal{S}$ is the state space, $\mathcal{A} = \prod_{i \in \mathcal{I}} \mathcal{A}^i$ is the joint micro-action space, $\mathcal{M} = \prod_{i \in \mathcal{I}} \mathcal{M}^i$ is the joint macro-action space, $\Omega = \prod_{i \in \mathcal{I}} \Omega^i$ is the joint micro-observation space, $\zeta = \prod_{i \in \mathcal{I}} \zeta^i$ is the joint macro-observation space, $\mathcal{T}(s'|s, a)$ is the state transition probability, $\mathcal{R}$ is the reward function shared over all agents, $O(o|s', a), o \in \Omega$ is the joint observation probability, $Z(z|s', m), z \in \zeta$ is the joint macro-observation probability, and γ is the discounting factor. Note that this paper uses "micro-action" instead of "primitive-action" for intuitive understanding.

Each macro-action 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 $\beta_{m^i} : \mathcal{H}^i_{\text{mic}} \to [0, 1]$, a initiation set $\mathcal{I}_{m^i} \subset \mathcal{H}^i_{\text{mac}}$, and a low-level policy $\pi_{m^i} : \mathcal{H}^i_{\text{mic}} \to \mathcal{A}^i$, where $\mathcal{H}^i_{\text{mic}}$ (or $\mathcal{H}^i_{\text{mac}}$) is the micro(or macro)-action-observation history space.

The objective in MacDec-POMDP is then to find a joint high-level policy $\Psi = \prod_{i \in \mathcal{I}} \Psi^i$ that maximizes the following expected discounted return from an initial state s_0 for given low-level policies, i.e., macro-actions:

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

B.1. Example of Macro-Actions in the Overcooked Environment

To illustrate how inter-agent interaction is handled in practice, consider the Overcooked environment:

- Intra-option policy: A macro-action like GO TO TOMATO may involve a path planning. If another agent is blocking the path, the intra-option policy handles this by having the agent wait until the path is clear, then continue. This behavior is naturally encoded into the policy without requiring the macro-action to terminate.
- Termination condition: If the task goal becomes invalid—for instance, another agent picks up the tomato—the macro-action terminates automatically according to its predefined termination condition. These behaviors are not considered "interruptions" in the conventional sense. Rather, they are expected outcomes under the macro-action's design. The inter-agent interaction and adaptation are achieved through well-defined intra-option policies and termination conditions, enabling cooperative behaviors to naturally emerge within the framework.

C. Comparison of the Procedures for Padding-Based Methods and ACAC

In this section, we provide a detailed explanation of how two different approaches—*Padding-based methods* and our proposed *ACAC*—operate within an Asynchronous MARL setting. To illustrate these procedures, we refer to the example in Figure 10. Both methods follow a CTDE (Centralized Training and Decentralized Execution) framework under a (MacDec-)POMDP setting, meaning they rely on macro-observation histories. In practice, these histories are typically processed by recurrent networks (e.g., RNNs or LSTMs) in both the decentralized actors and the centralized critic to combine past and current macro-observations effectively.

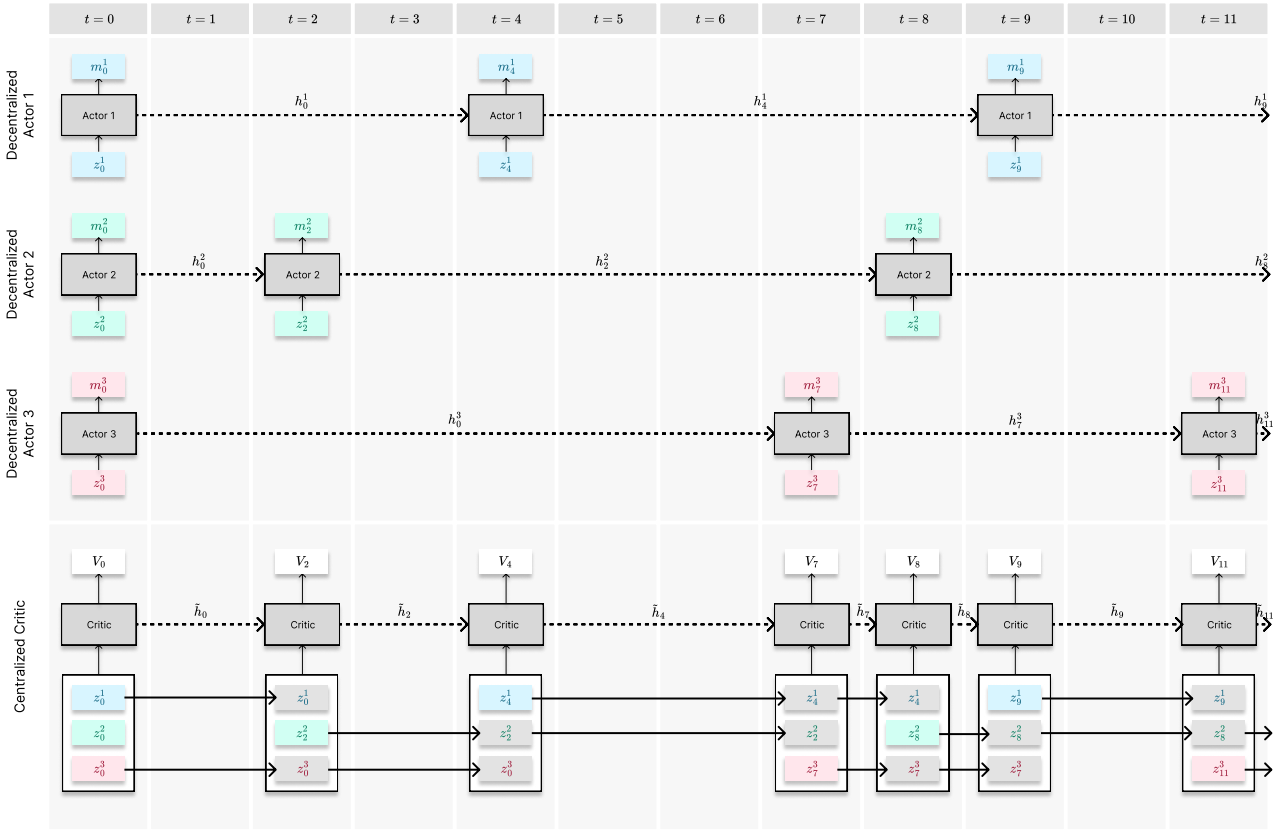


Figure 11. Full CTDE procedure of padding-based methods for the example in Figure 10

C.1. Procedure of Padding-Based Methods

In the *padding-based* approach, each agent's *decentralized actor* selects a macro-action solely based on its own macro-observation. The actor is only active at timesteps when a *new* macro-observation arrives (i.e., whenever a new macro-action can be chosen). At each such timestep, the agent updates its actor's hidden state to incorporate the newly received macro-observation.

Meanwhile, the *centralized critic* takes as input the concatenation of all agents' macro-observations, along with the concatenated history of these macro-observations, to estimate the joint value. When any agent obtains a new macro-observation, a joint macro-observation must be constructed to compute the value for that timestep. For agents that do not receive a new observation at that timestep, the critic uses each agent's most recent macro-observation. The centralized critic then updates its hidden state with this joint macro-observation and, based on the updated hidden state, provides a value estimate.

Figure 11 illustrates how the decentralized actors and the centralized critic operate under the Padding approach in the example from Figure 10.

C.2. Procedure of ACAC

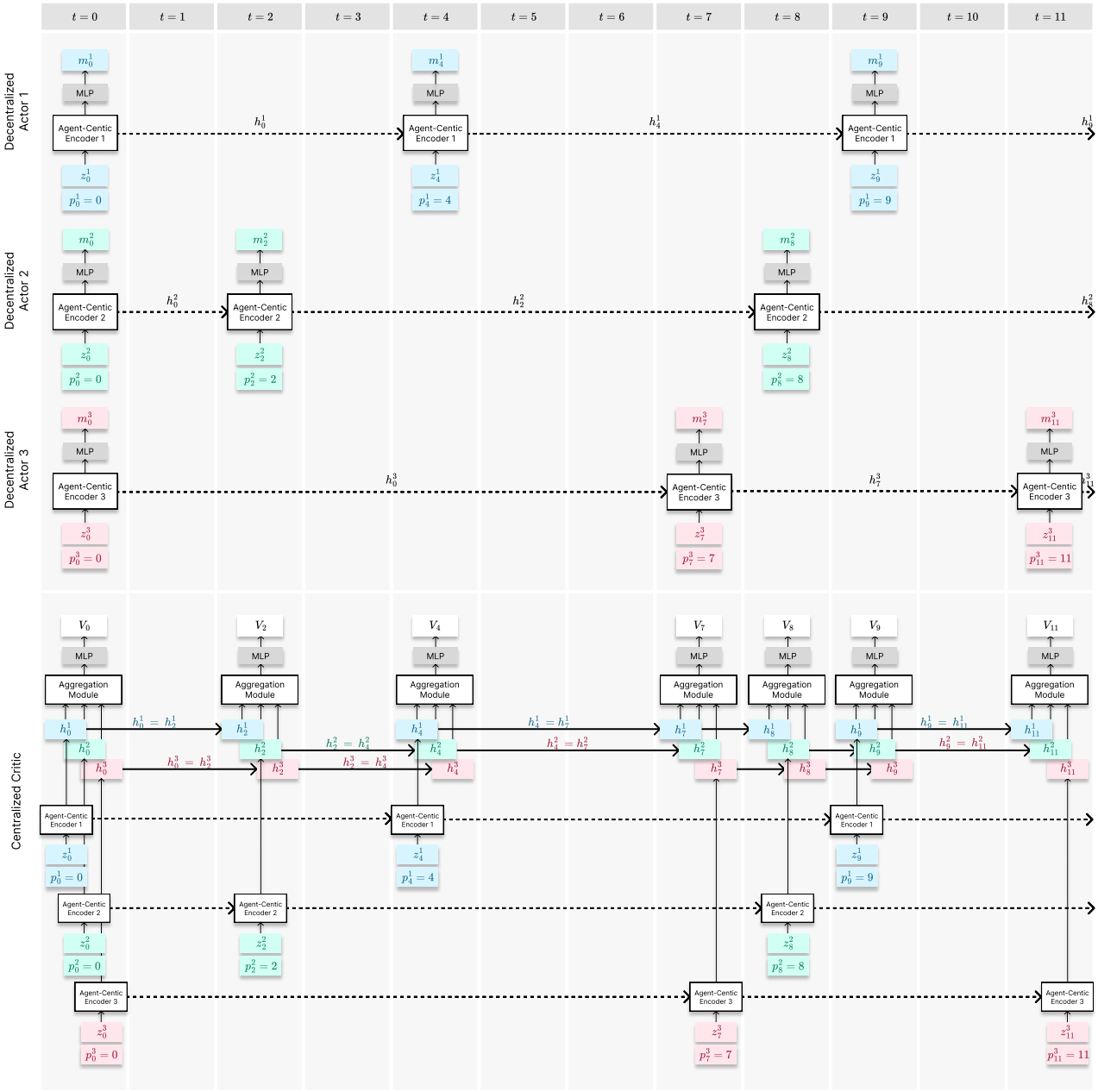


Figure 12. Full CTDE procedure of ACAC for the example in Figure 10

In ACAC, the decentralized actor functions similarly to that in padding-based methods, with one key difference: in addition to receiving the macro-observation as input, it also receives the timestep at which the macro-observation was obtained. This provides information about the duration between consecutive macro-observations, allowing for a richer history representation.

Structurally and operationally, ACAC's centralized critic differs significantly from padding-based methods. In the padding-based approach, all agents' macro-observations must be available at each timestep to compute the value; when no new macro-observation is obtained by a particular agent, its most recent observation is reused (i.e., *padded*). By contrast, the centralized critic in ACAC is designed to accept only new macro-observations as input. More concretely, each agent's macro-observations are processed by an agent-centric history encoder, which updates that agent's hidden state. At a timestep

when an agent receives a new macro-observation, it updates its hidden state through its agent-centric history encoder. The centralized critic then aggregates each agent's most recently updated hidden state—whether from the current or a previous timestep—via an aggregation module to estimate the value. This design enables the critic to combine agent-specific history representations without padding stale observations from other agents.

Figure 12 illustrates the structure and operation of ACAC's decentralized actor and centralized critic for the example in Figure 10.

D. GAE Derivation

D.1. Motivation and Objectives of GAE

Generalized Advantage Estimation (GAE) (Schulman et al., 2016) is a widely used technique in policy gradient methods such as PPO (Schulman et al., 2017), primarily because it helps strike a balance between the high variance of empirical returns and the high bias of temporal difference (TD) estimates. GAE introduces a hyperparameter λ that adjusts this bias–variance trade-off. Specifically, when $\lambda = 0$, GAE relies solely on TD estimates, while when $\lambda = 1$, it uses the empirical returns minus the current value estimate. Setting λ to an intermediate value between 0 and 1 allows for a smooth trade-off between these two extremes.

D.2. Illustrating GAE in Asynchronous MARL

In asynchronous MARL, intervals between consecutive macro-observations can vary. This irregularity renders the standard GAE formula (originally designed for synchronous MARL) inapplicable. To address this issue, we propose a modified GAE procedure that accounts for the asynchronous nature of MARL.

Rather than presenting the formula immediately, we begin with a concrete example illustrating how to compute GAE in an asynchronous setting (see Figure 10). In this example, new macro-observations occur at timesteps $t = 0, 2, 4, 7, 8, 9, 11$. From the perspective of macro-action decision points, the temporal differences (TDs) can be computed as follows:

- Timestep 0: $\delta_0 := \{r_0 + \gamma r_1\} + \gamma^2 V_2 - V_0$
- Timestep 2: $\delta_2 := \{r_2 + \gamma r_3\} + \gamma^2 V_4 - V_2$
- Timestep 4: $\delta_4 := \left\{r_4 + \gamma r_5 + \gamma^2 r_6\right\} + \gamma^3 V_7 - V_4$
- Timestep 7: $\delta_7 := \{r_7\} + \gamma V_8 - V_7$
- Timestep 8: $\delta_8 := \{r_8\} + \gamma V_9 - V_8$
- Timestep 9: $\delta_9 := \{r_9 + \gamma r_{10}\} + \gamma^2 V_{11} - V_9$

By taking the γ-discounted sum of these TDs, we obtain various advantage estimates at timestep 0. For instance:

$$
\begin{aligned}
A_0^{(1)} &:= \delta_0 \\
&= -V_0 + \{r_0 + \gamma r_1\} + \gamma^2 V_2 && (2)\\
A_0^{(2)} &:= \delta_0 + \gamma^2 \delta_2 \\
&= -V_0 + \left\{r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3\right\} + \gamma^4 V_4 && (3)\\
A_0^{(3)} &:= \delta_0 + \gamma^2 \delta_2 + \gamma^4 \delta_4 && (4)\\
&= -V_0 + \left\{r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \gamma^3 r_4 + \gamma^5 r_5 + \gamma^6 r_6\right\} + \gamma^7 V_7 && (5)\\
A_0^{(4)} &:= \delta_0 + \gamma^2 \delta_2 + \gamma^4 \delta_4 + \gamma^7 \delta_7 && (6)\\
&= -V_0 + \left\{r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \gamma^3 r_4 + \gamma^5 r_5 + \gamma^6 r_6 + \gamma^7 r_7\right\} + \gamma^8 V_8 && (7)\\
&\vdots \\
A_0^{(\infty)} &= -V_0 + \sum_{t=0}^{\infty} \gamma^t r_t && (8)
\end{aligned}
$$

Next, we apply an exponential weighting with parameter λ, analogous to the standard GAE approach, to produce the final

advantage estimate:

$$A^{(\lambda)}_{\text{GAE},0} := (1-\lambda)\Big(A^{(1)}_0 + \lambda A^{(2)}_0 + \lambda^2 A^{(3)}_0 + \lambda^3 A^{(4)}_0 + \dots\Big) \tag{9}$$

$$= (1-\lambda)\Big(\delta_0 + \lambda\{\delta_0 + \gamma^2\delta_2\} + \lambda^2\{\delta_0 + \gamma^2\delta_2 + \gamma^4\delta_4\} + \lambda^3\{\delta_0 + \gamma^2\delta_2 + \gamma^4\delta_4 + \gamma^7\delta_7\} + \dots\Big) \tag{10}$$

$$= (1-\lambda)\Big(\delta_0\{1 + \lambda + \lambda^2 + \dots\} + \gamma^2\delta_2\{\lambda + \lambda^2 + \lambda^3 + \dots\} \tag{11}$$

$$\quad + \gamma^4\delta_4\{\lambda^2 + \lambda^3 + \lambda^4 + \dots\} + \gamma^7\delta_7\{\lambda^3 + \lambda^4 + \lambda^5 + \dots\} + \dots\Big)$$

$$= (1-\lambda)\Big(\tfrac{1}{1-\lambda}\delta_0 + \gamma^2\tfrac{\lambda}{1-\lambda}\delta_2 + \gamma^4\tfrac{\lambda^2}{1-\lambda}\delta_4 + \gamma^7\tfrac{\lambda^3}{1-\lambda}\delta_7 + \dots\Big) \tag{12}$$

$$= \delta_0 + \gamma^2\lambda\delta_2 + \gamma^4\lambda^2\delta_4 + \gamma^7\lambda^3\delta_7 + \dots \tag{13}$$

Note that this version of GAE, derived for the asynchronous case, differs from the standard GAE used in synchronous MARL. The main adaptation is in how the TDs are accumulated across irregular macro-observation intervals to accurately reflect asynchronous dynamics.

D.3. Generalized Asynchronous GAE Formulation

Building on the above example, we now present a generalized formulation for asynchronous GAE. First, define the timesteps at which any agent obtains a new macro-observation. These timesteps can be defined recursively:

$$l_{(0)} = t, \tag{14}$$

$$l_{(k+1)} = \min\Big\{u \;\Big|\; \tilde{z}_{\le u} \neq \tilde{z}_{\le l_{(k)}},\; u > l_{(k)}\Big\} \quad \text{for } k \ge 0, \tag{15}$$

where $\tilde{z}_{\le u}$ denotes the histories of *all* agents up to timestep u. Intuitively, $l_{(k+1)}$ is the first timestep after $l_{(k)}$ at which any new macro-observation arises. Given these macro-observable timesteps $l_{(k)}$, we define the temporal difference at each $l_{(k)}$ as

$$\delta_{l_{(k)}} := -V_{l_{(k)}} \;+\; \Big\{\sum_{u=l_{(k)}}^{l_{(k+1)}-1} \gamma^{u-l_{(k)}} r_u\Big\} \;+\; \gamma^{l_{(k+1)}-l_{(k)}} V_{l_{(k+1)}}. \tag{16}$$

Next, for $t = l_{(k)}$, we define the advantage estimates $A^{(m)}_t$ using $\delta_{l_{(k)}}$ as follows:

$$A^{(1)}_t := \delta_{l_{(k)}}$$

$$= -V_{l_{(k)}} \;+\; \Big\{\sum_{u=l_{(k)}}^{l_{(k+1)}-1} \gamma^{u-l_{(k)}} r_u\Big\} \;+\; \gamma^{l_{(k+1)}-l_{(k)}} V_{l_{(k+1)}}, \tag{17}$$

$$A^{(2)}_t := \delta_{l_{(k)}} + \gamma^{l_{(k+1)}-l_{(k)}}\delta_{l_{(k+1)}}$$

$$= -V_{l_{(k)}} \;+\; \Big\{\sum_{u=l_{(k)}}^{l_{(k+2)}-1} \gamma^{u-l_{(k)}} r_u\Big\} \;+\; \gamma^{l_{(k+2)}-l_{(k)}} V_{l_{(k+2)}}, \tag{18}$$

$$A^{(3)}_t := \delta_{l_{(k)}} + \gamma^{l_{(k+1)}-l_{(k)}}\delta_{l_{(k+1)}} + \gamma^{l_{(k+2)}-l_{(k)}}\delta_{l_{(k+2)}}$$

$$= -V_{l_{(k)}} \;+\; \Big\{\sum_{u=l_{(k)}}^{l_{(k+3)}-1} \gamma^{u-l_{(k)}} r_u\Big\} \;+\; \gamma^{l_{(k+3)}-l_{(k)}} V_{l_{(k+3)}}, \tag{19}$$

$$\vdots$$

$$A^{(m)}_t := \delta_{l_{(k)}} \;+\; \gamma^{l_{(k+1)}-l_{(k)}}\delta_{l_{(k+1)}} \;+\; \gamma^{l_{(k+2)}-l_{(k)}}\delta_{l_{(k+2)}} \;+\; \cdots \;+\; \gamma^{l_{(k+m)}-l_{(k)}}\delta_{l_{(k+m)}}$$

$$= -V_{l_{(k)}} \;+\; \Big\{\sum_{u=l_{(k)}}^{l_{(k+m)}-1} \gamma^{u-l_{(k)}} r_u\Big\} \;+\; \gamma^{l_{(k+m)}-l_{(k)}} V_{l_{(k+m)}}. \tag{20}$$

We then compute an exponentially weighted sum of these terms with parameter λ (analogous to the original GAE):

$$A^{(\lambda)}_{\text{GAE},t} := (1-\lambda)\Big(A^{(1)}_t + \lambda A^{(2)}_t + \lambda^2 A^{(3)}_t + \cdots\Big) \tag{21}$$

$$= (1-\lambda)\Big(\delta_{l_{(k)}} + \lambda\{\delta_{l_{(k)}} + \gamma^{l_{(k+1)}-l_{(k)}}\delta_{l_{(k+1)}}\} + \lambda^2\{\delta_{l_{(k)}} + \gamma^{l_{(k+1)}-l_{(k)}}\delta_{l_{(k+1)}} + \gamma^{l_{(k+2)}-l_{(k)}}\delta_{l_{(k+2)}}\} + \cdots\Big)$$

$$= (1-\lambda)\Big(\delta_{l_{(k)}}\{1+\lambda+\lambda^2+\ldots\} + \gamma^{l_{(k+1)}-l_{(k)}}\delta_{l_{(k+1)}}\{\lambda+\lambda^2+\lambda^3+\ldots\} \tag{22}$$

$$+ \gamma^{l_{(k+2)}-l_{(k)}}\delta_{l_{(k+2)}}\{\lambda^2+\lambda^3+\lambda^4+\ldots\} + \ldots\Big)$$

$$= (1-\lambda)\Big(\tfrac{1}{1-\lambda}\delta_{l_{(k)}} + \gamma^{l_{(k+1)}-l_{(k)}}\tfrac{\lambda}{1-\lambda}\delta_{l_{(k+1)}} + \gamma^{l_{(k+2)}-l_{(k)}}\tfrac{\lambda^2}{1-\lambda}\delta_{l_{(k+2)}} + \cdots\Big) \tag{23}$$

$$= \delta_{l_{(k)}} + \gamma^{l_{(k+1)}-l_{(k)}}\lambda\delta_{l_{(k+1)}} + \gamma^{l_{(k+2)}-l_{(k)}}\lambda^2\delta_{l_{(k+2)}} + \cdots \tag{24}$$

$$= \sum_{m=k}^{\infty}\gamma^{l_{(m)}-l_{(k)}}\lambda^{m-k}\delta_{l_{(m)}}. \tag{25}$$

Notably, this asynchronous GAE formulation still satisfies the original GAE objectives. When $\lambda = 0$, it reduces to using only temporal differences:

$$A^{(0)}_{\text{GAE},t} = \delta_{l_{(k)}},$$

where $t = l_{(k)}$. When $\lambda = 1$, it relies on the empirical returns minus the current value function:

$$A^{(1)}_{\text{GAE},t} = \sum_{m=k}^{\infty}\gamma^{l_{(m)}-l_{(k)}}\,\delta_{l_{(m)}} \tag{26}$$

$$= \delta_{l_{(k)}} + \gamma^{l_{(k+1)}-l_{(k)}}\delta_{l_{(k+1)}} + \gamma^{l_{(k+2)}-l_{(k)}}\delta_{l_{(k+2)}} + \ldots \tag{27}$$

$$= \left(-V_{l_{(k)}} + \Big\{\sum_{u=l_{(k)}}^{l_{(k+1)}-1}\gamma^{u-l_{(k)}}r_u\Big\} + \gamma^{l_{(k+1)}-l_{(k)}}V_{l_{(k+1)}}\right) \tag{28}$$

$$+ \gamma^{l_{(k+1)}-l_{(k)}}\left(-V_{l_{(k+1)}} + \Big\{\sum_{u=l_{(k+1)}}^{l_{(k+2)}-1}\gamma^{u-l_{(k+1)}}r_u\Big\} + \gamma^{l_{(k+2)}-l_{(k+1)}}V_{l_{(k+2)}}\right) + \ldots$$

$$= \Big\{\sum_{u=l_{(k)}}^{\infty}\gamma^{u-l_{(k)}}r_u\Big\} - V_{l_{(k)}} \tag{29}$$

$$= \Big\{\sum_{u=t}^{\infty}\gamma^{u-t}r_u\Big\} - V_t. \tag{30}$$

E. Additional Results

E.1. Full Results on Effect of Removing Padding and Implementing Macro-level λ-discounting for GAE

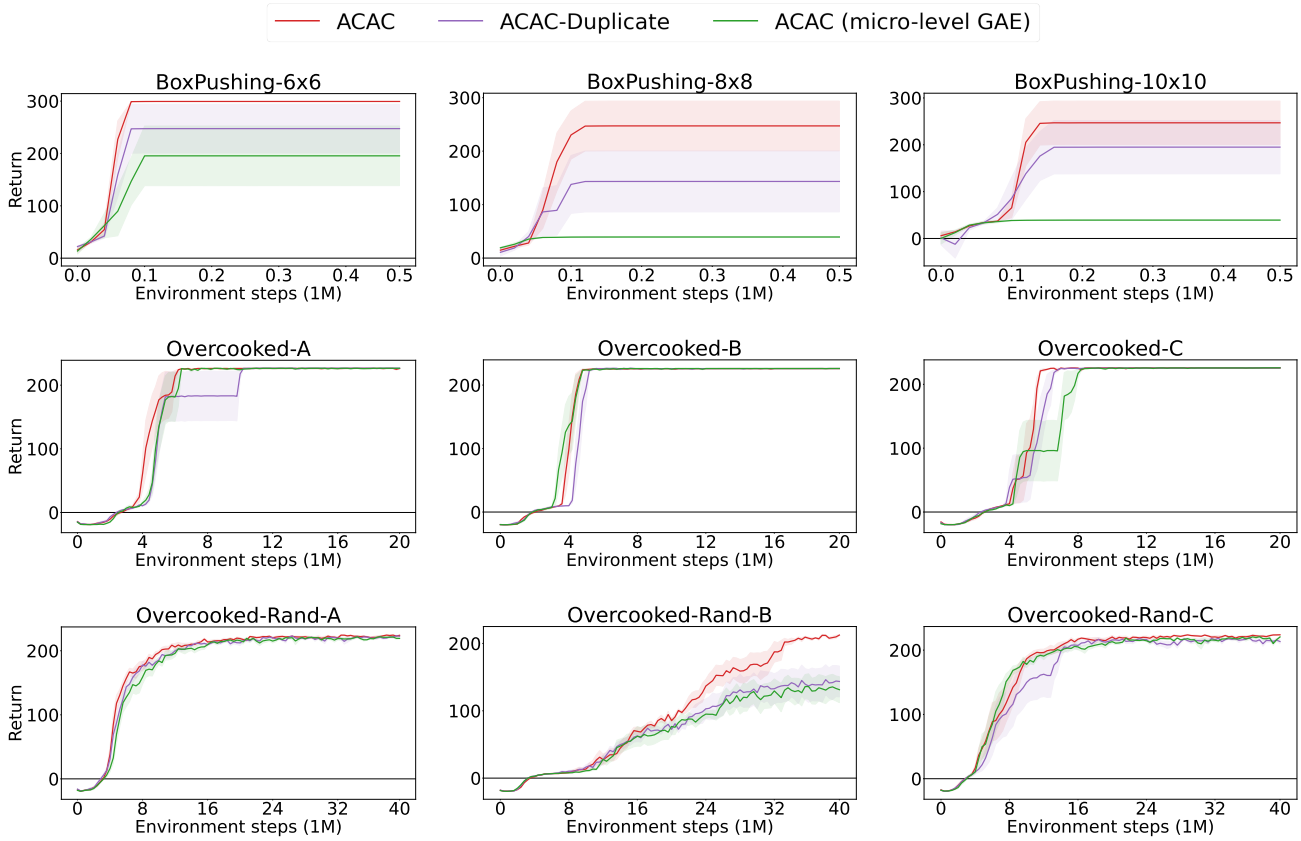


Figure 13. Ablation study examining the effects of removing padding and implementing macro-level λ-discounting for GAE

E.2. Further Sensitivity Analysis and Ablation Study

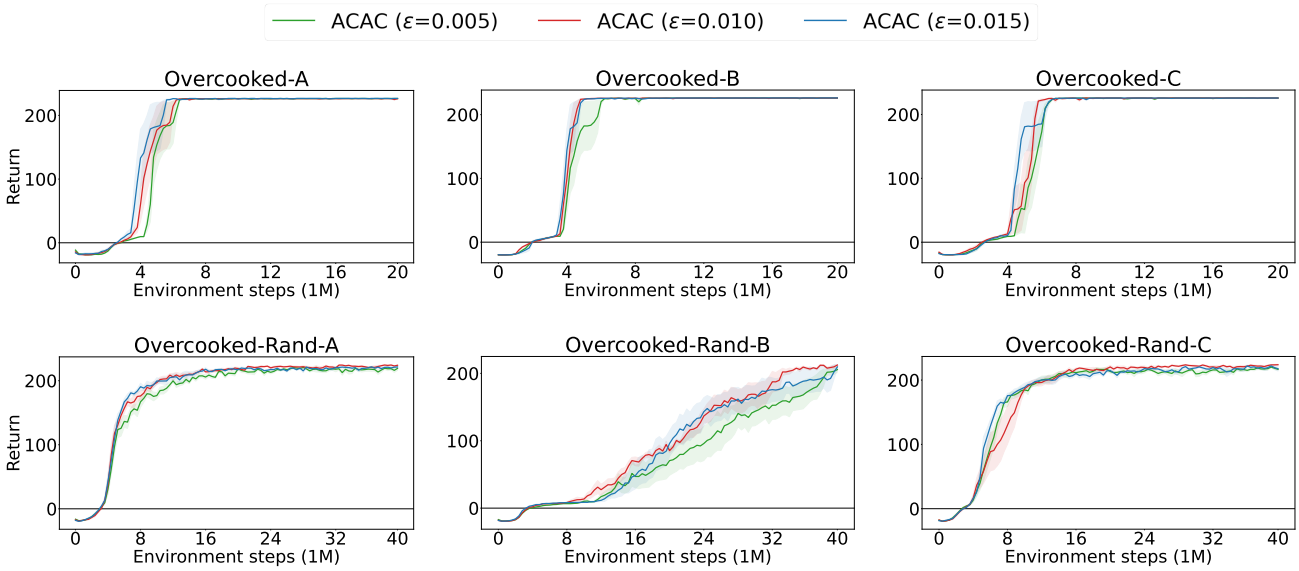


Figure 14. Sensitivity analysis of the effect of the clipping ratio ϵ

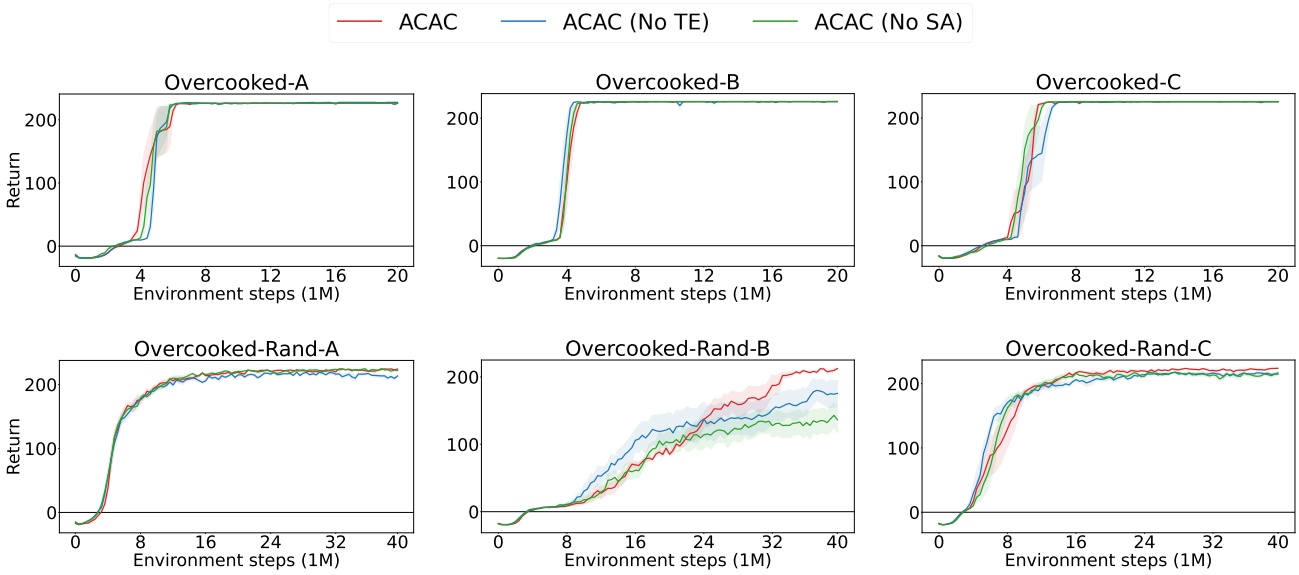


Figure 15. Ablation study on the effects of time embedding (TE) and self-attention (SA)

F. Hyper-parameters

The hyper-parameter used for our proposed method, ACAC, is described in Table 1. We mostly follow experimental setting of (Xiao et al., 2022), while we used the better value for some hyper-parameters.

Table 1. Common hyper-parameters used across the environment collections

Hyper-parameter	**BoxPushing**	**Overcooked**	**Overcooked-Rand**	**Overcooked-Large(-Rand)**
Total training timesteps	500K	20M	40M	100M
MLP Layer Size (Actor)	[32, 32]	[32, 32]		[128, 64]
RNN Layer Size (Actor)	32	32		64
MLP Layer Size (Critic)	[32, 32]	[128, 64]		
RNN Layer Size (Critic)	32	64		
Discount factor γ	0.98	0.99		
Episodes per train	16	8		
Episodes per target critic update	64	32		
Clipping ratio ϵ	0.05	0.01		
Epochs	2	5		
Episodes per train	16	8		
Max episode length	100	200		
Learning Rate (Actor)	3e-4			
Learning Rate (Critic)	3e-4			
Minibatch size	8			
GAE λ	0.95			

G. Computational Resources for Experiments

We used AMD EPYC 7453 28-Core Processor and A10 for our experiments, and the running time of the proposed ACAC typically ranges from approximately 24 hours to 150 hours on Overcooked environments and from approximately 30 minutes to 2 hours on BoxPushing environments.

H. Implementation and License Information

The implementation code for this work can be found at `https://github.com/LGAI-Research/acac`. The implementation code is based on the original implementation code for the previous work (Xiao et al., 2022), but there is no license information in the repository at `https://github.com/yuchen-x/MacroMARL`. We use the transformer implementation from Hugging Face, which is licensed under the Apache 2.0 License. The gym-cooking environment (Wu et al., 2021) at `https://github.com/rosewang2008/gym-cooking` and its macro-action version (Xiao et al., 2022) at `https://github.com/WeihaoTan/gym-macro-overcooked`, which were used for the evaluation of the proposed method and baseline algorithms, are licensed under the MIT license.