Asynchronous Actor-Critic for Multi-Agent Reinforcement Learning

Yuchen Xiao, Weihao Tan and Christopher Amato

***Abstract*— Synchronizing decisions across multiple agents in realistic settings is problematic since it requires agents to wait for other agents to terminate and communicate about termination reliably. Ideally, agents should learn and execute asynchronously instead. Such asynchronous methods also allow temporally extended actions that can take different amounts of time based on the situation and action executed. Unfortunately, current policy gradient methods are not applicable in asynchronous settings, as they assume that agents synchronously reason about action selection at every time step. To allow asynchronous learning and decision-making, we formulate a set of asynchronous multi-agent actor-critic methods that allow agents to directly optimize asynchronous policies in three standard training paradigms: decentralized learning, centralized learning, and centralized training for decentralized execution. Empirical results (in simulation and hardware) in a variety of realistic domains demonstrate the superiority of our approaches in large multi-agent problems and the effectiveness of our algorithms for learning high-quality and asynchronous solutions.**

I. INTRODUCTION

In recent years, multi-agent policy gradient methods using the actor-critic framework have achieved impressive success in solving a variety of cooperative and competitive domains [1], [2], [3], [4]. However, as these methods assume synchronized primitive-action execution over agents, they struggle to solve large-scale real-world multi-agent problems that involve long-term reasoning and asynchronous behavior.

Temporally-extended actions have been widely used in both learning and planning to improve scalability and reduce complexity. For example, they have come in the form of motion primitives [5], [6], skills [7], [8], spatial action maps [9] or macro-actions [10], [11], [12]. The idea of temporally-extended actions has also been incorporated into multi-agent approaches. In particular, we consider the *Macro-Action Decentralized Partially Observable Markov Decision Process* (MacDec-POMDP) [13]. The MacDec-POMDP is a general model for cooperative multi-agent problems with partial observability and (potentially) different action durations. Agents can start and end macro-actions at different time steps so decision-making can be asynchronous.

The MacDec-POMDP framework has shown strong scalability with planning-based methods (where the model is given) [14], [15], [16]. In terms of multi-agent reinforcement learning (MARL), there have been many hierarchical approaches, they don't typically address asynchronicity since they assume agents' high-level decisions with a same duration [17], [18], [19], [20]. Only limited studies consider asynchronicity [21], [22], [23], yet, none of them provides a general formulation for multi-agent policy gradients that allows agents to asynchronously learn and execute.

In this paper, we assume a set of macro-actions has been predefined for each domain. This is well-motivated by the fact that, in real-world multi-robot systems, each robot is already equipped with certain controllers (e.g., a navigation controller, and a manipulation controller) that can be modeled as macro-actions [14], [16], [24]. Similarly, as it is common to assume primitive actions are given in a typical RL domain, we assume the macro-actions are given in our case. The focus of the policy gradient methods is then on learning high-level policies over macro-actions.

Our contributions include a set of macro-action-based multi-agent actor-critic methods that generalize their primitive-action counterparts. First, we formulate a *macro-action-based independent actor-critic* (Mac-IAC) method. Although independent learning suffers from a theoretical curse of environmental non-stationarity, it allows fully online learning and may still work well in certain domains. Second, we introduce a *macro-action-based centralized actor-critic* (Mac-CAC) method, for the case where full communication is available during execution. We also formulate a centralized training for decentralized execution (CTDE) paradigm [25] variant of our method. CTDE has gained popularity since such methods can learn better decentralized policies by using centralized information. Current primitive-action-based multi-agent actor-critic methods typically use a centralized critic to optimize each decentralized actor. However, the asynchronous joint macro-action execution from the centralized perspective could be very different with the completion time being very different from each agent's decentralized perspective. To this end, we first present a *Naive Independent Actor with Centralized Critic* (Naive IACC) method that naively uses a joint macro-action-value function as the critic for each actor's policy gradient estimation; and then propose a novel *Independent Actor with Individual Centralized Critic* (Mac-IAICC) method that learns individual critics using centralized information to address the above challenge.

We evaluate our proposed methods on diverse macro-action-based multi-agent problems: a benchmark Box Pushing domain [23], a variant of the Overcooked domain [26] and a larger warehouse service domain [23]. Experimental results show that our methods are able to learn high-quality solutions while primitive-action-based methods cannot, and show the strength of Mac-IAICC for learning decentralized policies over Naive IAICC and Mac-IAC. Decentralized policies learned by using Mac-IAICC are deployed on real robots to solve a warehouse tool delivery task in an efficient way. To our knowledge, this is the first general formalization of macro-action-based multi-agent actor-critic frameworks.

Khoury College of Computer Sciences, Northeastern University, Boston, MA 02115, USA {xiao.yuch, w.tan, c.amato}@northeastern.edu

II. Background

A. MacDec-POMDPs

Formally, a macro-action decentralized partially observable Markov decision process (MacDec-POMDP) [13] is defined by a tuple $\langle I, S, A, M, \Omega, \zeta, T, R, O, Z, \mathbb{H}, \gamma \rangle$, where I is a set of agents; S is the environmental state space; $A = \times_{i \in I} A_i$ is the joint primitive-action space over each agent's primitive-action set A_i; $M = \times_{i \in I} M_i$ is the joint macro-action space over each agent's macro-action space M_i; $\Omega = \times_{i \in I} \Omega_i$ is the joint primitive-observation space over each agent's primitive-observation set Ω_i; $\zeta = \times_{i \in I} \zeta_i$ is the joint macro-observation space over each agent's macro-observation space ζ_i; $T(s, \vec{a}, s') = P(s'|s, \vec{a})$ is the environmental transition dynamics; and $R(s, \vec{a})$ is a global reward function. During execution, each agent independently selects a macro-action m_i using a high-level policy $\Psi_i : H_i^M \times M_i \rightarrow [0, 1]$ and captures a macro-observation $z_i \in \zeta_i$ according to the macro-observation probability function $Z_i(z_i, m_i, s') = P(z_i \mid m_i, s')$ when the macro-action terminates in a state s'. Each macro-action is represented as $m_i = \langle I_{m_i}, \pi_{m_i}, \beta_{m_i} \rangle$, where $I_{m_i} \subset H_i^M$ is the initiation set of a macro-action based on macro-observation-action history H_i^M; $\pi_{m_i} : H_i^A \times A_i \rightarrow [0, 1]$ is the low-level policy for achieving a macro-action, and during the running, the agent receives a primitive-observation $o_i \in \Omega_i$ based on the observation function $O_i(o_i, a_i, s) = P(o_i|a_i, s)$ at every time step; $\beta_{m_i} : H_i^A \rightarrow [0, 1]$ is a stochastic termination function that determines how to terminate a macro-action based on primitive-observation-action history H_i^A. The objective of solving MacDec-POMDPs with finite horizon $\mathbb{H}$ is to find a joint high-level policy $\vec{\Psi} = \times_{i \in I} \Psi_i$ that maximizes the value, $V^{\vec{\Psi}}(s_{(0)}) = \mathbb{E}\Big[\sum_{t=0}^{\mathbb{H}-1} \gamma^t r\big(s_{(t)}, \vec{a}_{(t)}\big) \mid s_{(0)}, \vec{\pi}, \vec{\Psi}\Big]$. where $\gamma \in [0, 1]$ is a discount factor.

B. Multi-Agent Actor-Critics with Primitive-Actions

The single-agent actor-critic algorithm [27] can be directly adapted to multi-agent problems such that each agent independently learns its own actor and critic [2]. We consider a variance reduction version of *independent actor-critic* (IAC) with the policy gradient as: $\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\vec{\pi}_{\vec{\theta}}}\Big[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|h_i)\big(r + \gamma V_{\mathbf{w}_i}^{\pi_{\theta_i}}(h_i') - V_{\mathbf{w}_i}^{\pi_{\theta_i}}(h_i)\big)\Big]$. Due to other agents' policy updating and exploring, from any agent's local perspective, the environment appears non-stationary which can lead to unstable learning of the critic without convergence guarantees [3]. This instability often prevents IAC from learning high-quality cooperative policies. To address the above difficulties in IAC, centralized training for decentralized execution (CTDE) provides agents with access to global information during training while allowing agents to rely on local information during execution. *Independent actor with centralized critic* (IACC) [2], [3] is a typical implementation of CTDE that trains a joint value function as the centralized critic and use it to compute gradients w.r.t. the parameters of each decentralized policy as: $\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\vec{\pi}_{\vec{\theta}}}\Big[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i \mid h_i)\big(r + \gamma V_{\mathbf{w}}^{\vec{\pi}_{\vec{\theta}}}(\mathbf{x}') - V_{\mathbf{w}}^{\vec{\pi}_{\vec{\theta}}}(\mathbf{x})\big)\Big]$ where, $\mathbf{x}$ represents the available centralized information (e.g., joint observation-action history, or the true state).

C. Learning Macro-Action-Based Deep Q-Nets

Recently, macro-action-based multi-agent DQNs have been proposed for MacDec-POMDPs [23]. For decentralized learning, a new buffer, *Macro-Action Concurrent Experience Reply Trajectories* (Mac-CERTs), is designed for collecting each agent's macro-observation, macro-action, and reward information. In this buffer, the transition experience of each agent i is represented as a tuple $\langle z_i, m_i, z_i', r_i^c \rangle$, where $r_i^c = \sum_{t=t_{m_i}}^{t_{m_i}+\tau_{m_i}-1} \gamma^{t-t_{m_i}} r_{(t)}$ is a cumulative reward of the macro-action taking τ_{m_i} time steps to be completed from its beginning time step t_{m_i}. In the training phase, a mini-batch of concurrent sequential experiences is sampled, and each agent independently squeezes its own trajectories by removing the transitions during each macro-action execution (i.e., removing time information), which produces a mini-batch of transitions when the corresponding macro-action terminates. Each agent's macro-action-value function $Q_{\phi_i}(h_i, m_i)$ is updated only when a macro-action is complete by minimizing a TD loss over the 'squeezed' data. For centralized learning, the objective is to learn a joint macro-action-value function $Q_\phi(\vec{h}, \vec{m})$. To this end, a special buffer called *Macro-Action Joint Experience Replay Trajectories* (Mac-JERTs) is developed for collecting agents' joint transition experience at every time step and each is represented as a tuple $\langle \vec{z}, \vec{m}, \vec{z}', \vec{r}^c \rangle$, where $\vec{r}^c = \sum_{t=t_{\vec{m}}}^{t_{\vec{m}}+\vec{\tau}_{\vec{m}}-1} \gamma^{t-t_{\vec{m}}} r_{(t)}$ is a shared joint cumulative reward from the beginning time step $t_{\vec{m}}$ of the joint macro-action $\vec{m}$ to its termination, defined as when *any* agent finishes its own macro-action, after $\vec{\tau}_{\vec{m}}$ time steps. In each training iteration, the joint macro-action-value function is optimized over a mini-batch of 'squeezed' (depending on each joint macro-action termination) sequential joint experiences via TD learning.

III. Approach

MARL with asynchronous macro-actions is more challenging as it is difficult to determine *when* to update each agent's policy and *what* information to use. Although the macro-action-based DQN methods [23] give us the base to learn macro-action value functions, they do not directly extend to the policy gradient case, particularly in the case of centralized training for decentralized execution (CTDE). In this section, we propose principled formulations of on-policy macro-action-based multi-agent actor-critic methods for decentralized learning (Section III-A), centralized learning (Section III-B), and CTDE (Section III-C). We use h_i to represent an agent's local macro-observation-action history, and $\vec{h}$ to represent the joint history.

A. Macro-Action-Based Independent Actor-Critic (Mac-IAC)

Similar to the idea of IAC with primitive-actions, a straightforward extension is to have each agent independently optimize its own macro-action-based policy (actor) using a local value function (critic). In our case, we learn a local history value function $V_{\mathbf{w}_i}^{\Psi_{\theta_i}}(h_i)$ via the decentralized way

mentioned in Section II-C as each agent's critic and have the corresponding policy gradient as:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\vec{\Psi}_{\vec{\theta}}}\Big[\nabla_{\theta_i} \log \Psi_{\theta_i}(m_i \mid h_i) A(h_i, m_i)\Big] \quad (1)$$

$$A(h_i, m_i) = r_i^c + \gamma^{\tau_{m_i}} V_{\mathbf{w}_i}^{\Psi_{\theta_i}}(h_i') - V_{\mathbf{w}_i}^{\Psi_{\theta_i}}(h_i) \quad (2)$$

where, the cumulative reward r_i^c is w.r.t. the execution of agent i's macro-action m_i.

B. Macro-Action-Based Centralized Actor-Critic (Mac-CAC)

In the fully centralized learning case, we treat all agents as a single joint agent to learn a centralized actor $\Psi_\theta(\vec{m} \mid \vec{h})$ with a centralized critic. Similarly, to achieve a lower variance optimization for the actor, we learn a centralized history value function $V_{\mathbf{w}}^{\Psi_\theta}(\vec{h})$ by minimizing a TD-error loss over joint trajectories squeezed in the centralized way presented in Section II-C. Accordingly, the policy's updates are performed when each joint macro-action is completed by ascending the following gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\Psi_\theta}\Big[\nabla_\theta \log \Psi_\theta(\vec{m} \mid \vec{h}) A(\vec{h}, \vec{m})\Big] \quad (3)$$

$$A(\vec{h}, \vec{m}) = \vec{r}^{\,c} + \gamma^{\vec{\tau}_{\vec{m}}} V_{\mathbf{w}}^{\Psi_\theta}(\vec{h}') - V_{\mathbf{w}}^{\Psi_\theta}(\vec{h}) \quad (4)$$

where the cumulative reward $\vec{r}^{\,c}$ is w.r.t. the execution of the joint macro-action $\vec{m}$.

C. Macro-Action-Based Independent Actor with Centralized Critic (Mac-IACC)

In order to learn better decentralized macro-action-based policies, we propose two macro-action-based actor-critic algorithms using the CTDE paradigm. The difference between a joint macro-action termination from the centralized perspective and a macro-action termination from each agent's local perspective gives rise to a new challenge: *what kind of centralized critic should be learned and how should it be used to optimize decentralized policies where some have completed and some have not*, which we investigate below.

Naive Mac-IACC. A naive way of incorporating macro-actions into a CTDE-based actor-critic framework is to directly adapt the idea of the primitive-action-based IACC [2] to have a shared joint value function $V_{\mathbf{w}}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x})$ in each agent's decentralized macro-action-based policy gradient as:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\vec{\Psi}_{\vec{\theta}}}\Big[\nabla_{\theta_i} \log \Psi_{\theta_i}(m_i \mid h_i) A(\mathbf{x}, \vec{m})\Big] \quad (5)$$

$$A(\mathbf{x}, \vec{m}) = \vec{r}^{\,c} + \gamma^{\vec{\tau}_{\vec{m}}} V_{\mathbf{w}}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x}') - V_{\mathbf{w}}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x}) \quad (6)$$

Here, the critic is trained in the fully centralized manner described in section II-C while allowing it to access additional global information (e.g., joint macro-observation-action history, ground truth state or both) represented by the symbol $\mathbf{x}$. However, updates of each agent's policy $\Psi_{\theta_i}(m_i \mid h_i)$ only occur at the agent's own macro-action termination time steps rather than depending on joint macro-action terminations in the centralized critic training.

Independent Actor with Individual Centralized Critic (Mac-IAICC). Note that naive Mac-IACC is technically incorrect. The cumulative reward $\vec{r}^{\,c}$ in Eq. 6 is based on the corresponding joint macro-action's termination that is defined as when *any* agent finishes its own macro-action, which produces two potential issues: a) $\vec{r}^{\,c} + \gamma^{\vec{\tau}_{\vec{m}}} V_{\mathbf{w}}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x}')$ may not estimate the value of the macro-action m_i well as the reward does not depend on m_i's termination; b) from agent i's perspective, its policy gradient estimation may involve higher variance associated with the asynchronous macro-action terminations of other agents.

To tackle the aforementioned issues, we propose to learn a separate centralized critic $V_{\mathbf{w}_i}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x}')$ for each agent via TD-learning. In this case, the TD-error for updating $V_{\mathbf{w}_i}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x}')$ is computed by using the reward r_i^c that is accumulated purely based on the execution of the agent i's macro-action m_i. With this TD-error estimation, each agent's decentralized macro-action-based policy gradient becomes:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\vec{\Psi}_{\vec{\theta}}}\Big[\nabla_{\theta_i} \log \Psi_{\theta_i}(m_i \mid h_i) A(\mathbf{x}, m_i)\Big] \quad (7)$$

$$A(\mathbf{x}, m_i) = r_i^c + \gamma^{\tau_{m_i}} V_{\mathbf{w}_i}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x}') - V_{\mathbf{w}_i}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x}) \quad (8)$$

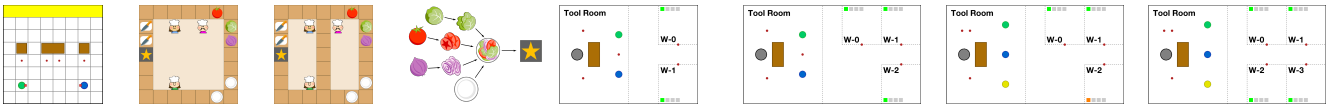
Now, from agent i's perspective, $r_i^c + \gamma^{\tau_{m_i}} V_{\mathbf{w}_i}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x}')$ is able to offer a more accurate value prediction for the macro-action m_i, since both the reward, r_i^c and the value function $V_{\mathbf{w}_i}^{\vec{\Psi}_{\vec{\theta}}}(\mathbf{x}')$ depend on agent i's macro-action termination. Also, unlike the case in Naive Mac-IACC, other agents' terminations cannot lead to extra noisy estimated rewards w.r.t. m_i anymore so that the variance on policy gradient estimation gets reduced. Then, updates for both the critic and the actor occur when the corresponding agent's macro-action ends and take the advantage of information sharing.

IV. Simulation Experiments

A. Domain Setup

We evaluate our algorithms over a variety of multi-agent problems with macro-actions (Fig. 1): Box Pushing [23], Overcooked [26], and a larger Warehouse Tool Delivery [23] domain. Macro-actions are defined using prior domain knowledge as they are straightforward in these tasks. We include primitive-actions into macro-action set (as one-step macro-actions), which gives agents the chance to learn more complex policies that use both when it is necessary.

Box Pushing (Fig. 1a). The optimal solution for the robots is to cooperatively push the big box to the yellow goal area for a terminal reward, but partial observability makes this difficult. Robots have four primitive-actions: *move forward*, *turn-left*, *turn-right* and *stay*. In the macro-action case, each robot has three one-step macro-actions: ***Turn-left***, ***Turn-right***, and ***Stay***, as well as three multi-step macro-actions: ***Move-to-small-box(i)*** and ***Move-to-big-box(i)*** navigate the robot to the red spot below the corresponding box and terminate with the robot facing the box; ***Push*** causes the robot to keep moving forward until arriving at the world's boundary (potentially pushing the small box or trying to push the big one). The big box only moves if both agents push it together. Each robot can only observe the status (*empty*, *teammate*, *boundary*, *small or big box*) of the cell in front of it. A penalty is issued when any robot hits the boundary or pushes the big box alone.



(a) BP (b) O-A (c) O-B (d) Recipe (e) WTD-A (f) WTD-B (g) WTD-C (h) WTD-D

Fig. 1: Experimental environments: Box Pushing (BP), Overcooked(O), and Warehouse Tool Delivery (WTD).

Overcooked (Fig. 1b - 1d). Three agents must learn to cooperatively prepare a lettuce-tomato-onion salad and deliver it to the 'star' cell. The challenge is that the salad's recipe (Fig. 1d) is unknown to agents. With primitive-actions (*move up*, *down*, *left*, *right*, and *stay*), agents can move around and achieve picking, placing, chopping and delivering by standing next to the corresponding cell and moving against it (e.g., in Fig. 1b, the pink agent can *move right* and then *move up* to pick up the tomato). Each agent's macro-action set consists of: a) five one-step macro-actions that are the same as the primitive ones; b) ***Chop***, cuts a raw vegetable into pieces when the agent stands next to a cutting board and an unchopped vegetable is on the board, otherwise it does nothing; c) long-term navigation macro-actions: ***Get-Lettuce***, ***Get-Tomato***, ***Get-Onion***, ***Get-Plate-1/2***, ***Go-Cut-Board-1/2*** and ***Deliver***, which navigate the agent to the location of the corresponding object with various possible terminal effects (e.g., holding a vegetable in hand, placing a chopped vegetable on a plate, arriving at the cell next to a cutting board, delivering an item to the star cell, or immediately terminating when any property condition does not hold, e.g., no path is found or the vegetable/plate is not found); d) ***Go-Counter*** (only available in Overcook-B, Fig. 1c), navigates an agent to the center cell in the middle of the map when the cell is not occupied, otherwise, it moves to an adjacent cell. If the agent is holding an object or one is at the cell, the object will be placed or picked up. Each agent only observes the *positions* and *status* of the entities within a 5×5 square centered on the robot.

Warehouse Tool Delivery (Fig. 1e - 1h). In each workshop (e.g., W-0), a human is working on an assembly task (involving 4 sub-tasks that each takes a number of time steps to complete) and requires three different tools for future subtasks to continue. A robot arm (grey) must find tools for each human on the table (brown) and pass them to mobile robots (green, blue and yellow) who are responsible for delivering tools to humans. Note that, the correct tools needed by each human are unknown to robots, which has to be learned during training in order to perform efficient delivery. A delayed delivery leads to a penalty. We consider variants with two or three mobile robots and two to four humans to examine the scalability of our methods (Fig. 1f - 1h). We also consider one faster human (orange) to check if robots can prioritize him (Fig. 1g). Mobile robots have the following macro-actions: ***Go-W(i)***, moves to the waypoint (red) at workshop i; ***Go-TR***, goes to the waypoint at the right side of the tool room (covered by the blue robot in Fig. 1g and 1h); and ***Get-Tool***, navigates to a pre-allocated waypoint (that is different for each robot to avoid collisions) next to the robot arm and waits there until either receiving a tool or 10 time steps have passed. The robot arm's macro-actions are: ***Search-Tool(i)***, finds tool i and places it in a staging area (containing at most two tools) on the table, and otherwise, it freezes the robot for the amount of time the action would take when the area is fully occupied; ***Pass-to-M(i)***, passes the first staged tool to mobile robot i; and ***Wait-M***, waits for 1 time step. The robot arm only observes the *type* of each tool in the staging area and *which mobile robot* is waiting at the adjacent waypoints. Each mobile robot always knows its *position* and the *type* of tool that it is carrying, and can observe the *number* of tools in the staging area or the *sub-task* a human is working on only when at the tool room or the workshop respectively.

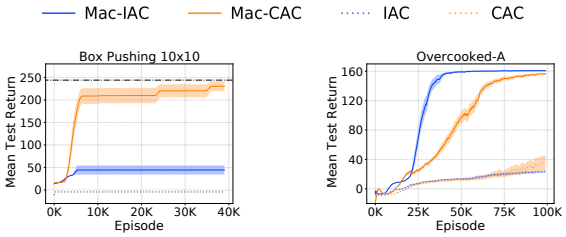


Fig. 2: Decentralized learning and centralized learning with macro-actions vs primitive-actions.

B. Results and Discussions

The metrics for a training trial is the mean discounted return measured by periodically (every 100 episodes) testing the learned policies over 10 episodes. We plot the average performance of each method over 20 independent runs with one standard error and smooth the curves over 10 neighbors.

Advantages of learning with macro-actions. We first present a comparison of our macro-action-based actor-critic methods against the primitive-action-based methods in fully decentralized and fully centralized cases. In Fig. 2, the results show significant performance improvements of using macro-actions over primitive-actions. More concretely, in the Box Pushing domain, reasoning about primitive movements at every time step makes the problem intractable so the robots cannot learn any good behaviors in primitive-action-based approaches other than to keep moving around. Conversely, Mac-CAC reaches near-optimal performance (dash-dot line), enabling the robots to push the big box together. Unlike the centralized critic which can access joint information, even in the macro-action case, it is hard for each robot's decentralized critic to correctly measure the responsibility for a penalty caused by a teammate pushing the big box alone. Mac-IAC thus converges to a local-optima of pushing two small boxes in order to avoid getting the penalty.

In the Overcooked domain, an efficient solution requires the robots to asynchronously work on independent subtasks (e.g., in scenario A, one robot gets a plate while another two robots pick up and chop vegetables; and in scenario B, the

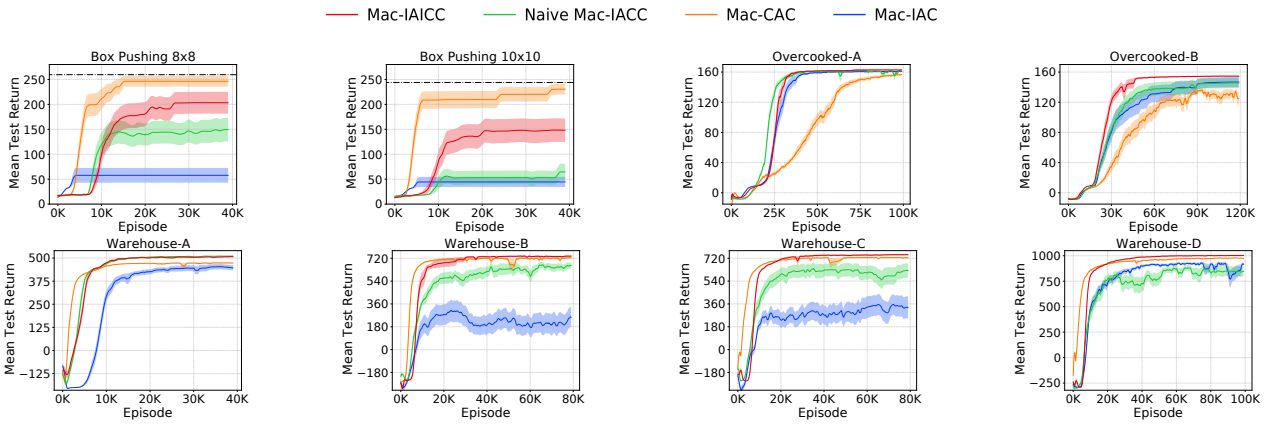


Fig. 3: Comparison of macro-action-based asynchronous actor-critic methods.

right robot transports items while the left two robots prepare the salad). This large amount of independence explains why Mac-IAC can solve the task well, and indicates that using local information is enough for robots to achieve high-quality behaviors. Mac-CAC learns slower because it must figure out the redundant part of joint information in much larger joint macro-level history and action spaces than the spaces in the decentralized case. The primitive-action-based methods begin to learn, but perform poorly in such long-horizon tasks.

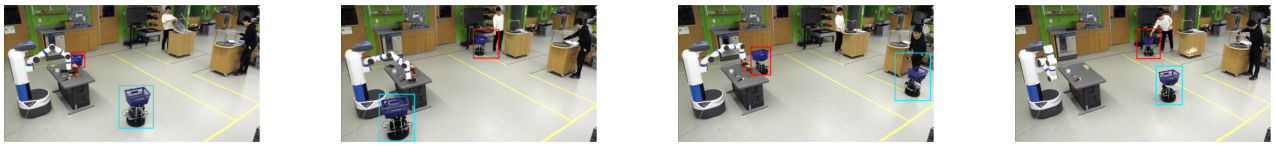
Advantages of having individual centralized critics. In the Box Pushing task (the left two in the top row in Fig. 3), Naive Mac-IACC (green) can learn policies almost as good as the ones for Mac-IAICC (red) for the smaller domain, but as the grid world size grows, Naive Mac-IACC performs poorly while Mac-IAICC keeps its performance near the centralized approach. From each agent's perspective, the bigger the world size is, the more time steps a macro-action could take, and the less accurate the critic of Naive Mac-IACC becomes since it is trained depending on any agent's macro-action termination. Conversely, Mac-IAICC gives each agent a separate centralized critic trained with the reward associated with its own macro-action execution.

In Overcooked-A (the third one at the top row in Fig. 3), as Mac-IAICC's performance is determined by the training of three agents' critics, it learns slower than Naive Mac-IACC in the early stage but converges to a slightly higher value and has better learning stability than Naive Mac-IACC in the end. The result of scenario B (the last one at the top row in Fig. 3) shows that Mac-IAICC outperforms other methods in terms of achieving better sample efficiency, a higher final return and a lower variance. The middle wall in scenario B limits each agent's moving space and leads to a higher frequency of macro-action terminations. The shared centralized critic in Naive Mac-IACC thus provides more noisy value estimations, so that it performs worse with more variance. Mac-IAICC, however, does not get hurt by such environmental dynamics change. Both Mac-CAC and Mac-IAC are not competitive with Mac-IAICC in this domain.

In the Warehouse scenarios (the bottom row in Fig. 3), Mac-IAC (blue) performs the worst due to its natural limitations and the domain's partial observability. In particular, it is difficult for the gray robot (arm) to learn an efficient way to find the correct tools purely based on local information and very delayed rewards that depend on the mobile robots' behaviors. In contrast, in the fully centralized Mac-CAC (orange), both the actor and the critic have global information so it can learn faster in the early training stage. However, Mac-CAC eventually gets stuck at a local-optimum in all five scenarios due to the exponential dimensionality of joint history and action spaces over robots. By leveraging the CTDE paradigm, both Mac-IAICC and Naive Mac-IACC perform the best in warehouse A. Yet, the weakness of Naive Mac-IACC is clearly exposed when the problem is scaled up in Warehouse B, C and D. In these larger cases, the robots' asynchronous macro-action executions (e.g., traveling between rooms) become more complex and cause more mismatching between the termination from each agent's local perspective and the termination from the centralized perspective. Therefore, Naive Mac-IACC's performance significantly deteriorates, even getting worse than Mac-IAC in Warehouse-D. In contrast, Mac-IAICC can maintain its outstanding performance, converging to a higher value with much lower variance than others. This outcome confirms not only Mac-IAICC's scalability but also the effectiveness of having an individual critic for each agent to handle variable degrees of asynchronicity in agents' high-level decision-making.

V. Hardware Experiments

We also extend scenario A of the Warehouse Tool Delivery task to a hardware domain. Fig. 4 shows the sequential collaborative behaviors of the robots in one hardware trial. Fetch was able to find tools in parallel such that two tape measures (Fig. 4a), two clamps (Fig. 4b) and two electric drills were found instead of finding all three types of tool for one human and then moving on to the other which would result in one of the humans waiting. Fetch's efficiency is also reflected in the behaviors such that it passed a tool to the Turtelbot who arrived first (Fig. 4b) and continued to find the next tool when there was no Turtlebot waiting beside it (Fig. 4c). Meanwhile, Turtlebots were clever such that they successfully avoid delayed delivery by sending tools one by one to the nearby workshop (e.g., T-0 focused on W-0 shown in Fig. 4b and 4d, and T-1 focused on W-1 shown in Fig. 4c), rather than waiting for all tools before delivering, traveling



(a) (b) (c) (d)

Fig. 4: Collaborative behaviors generated by running the decentralized policies learned by Mac-IAICC where Turtlebot-0 (T-0) bounded in red and Turtlebot-1 (T-1) is bounded in blue. (a) After staging a tape measure at the left, Fetch looks for the 2nd one while Turtlebots approach the table; (b) T-0 deliveries a tap measure to W-0 and T-1 waits for a clamp from Fetch; (c) T-1 deliveries a clamp to W-1, while T-0 carries the other clamp and goes to W-0, and Fetch searches for an electric drill; (d) T-0 deliveries an electric drill (the last tool) to W-0 and the entire delivery task is completed.

a longer distance to serve the human at the diagonal, or prioritizing one of the humans altogether.

VI. Conclusion

This paper proposes a decentralized actor-critic method (Mac-IAC), a centralized actor-critic method (Mac-CAC), and two CTDE-based actor-critic methods (Naive Mac-IACC and Mac-IAICC). These are the first approaches to be able to incorporate controllers that may require different amounts of time to complete (macro-actions) in a general asynchronous multi-agent actor-critic framework. Empirically, our methods can learn high-quality macro-action-based policies allowing agents to asynchronously collaborate in large and long-horizon problems. The practicality of our approach is validated in a real-world multi-robot setup based on a warehouse domain. This work provides a foundation for future macro-action-based MARL algorithm development, including methods which also learn the macro-actions.

References

[1] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," in *Proceedings of the International Conference on Learning Representations*, 2020.

[2] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, February 2018.

[3] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Proceedings of the Conference on Neural Information Processing Systems*, 2017.

[4] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," 2021.

[5] M. Dalal, D. Pathak, and R. Salakhutdinov, "Accelerating robotic reinforcement learning via parameterized action primitives," in *Proceedings of the Conference on Neural Information Processing Systems*, 2021.

[6] F. Stulp and S. Schaal, "Hierarchical reinforcement learning with movement primitives," in *11th IEEE-RAS International Conference on Humanoid Robots*, 2011.

[7] G. D. Konidaris, S. Kuindersma, R. A. Grupen, and A. G. Barto, "Autonomous skill acquisition on a mobile manipulator," in *Proceedings of the AAAI Conference on Artificial Intelligence*, W. Burgard and D. Roth, Eds., 2011.

[8] G. D. Konidaris, L. P. Kaelbling, and T. Lozano-Pérez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018.

[9] J. Wu, X. Sun, A. Zeng, S. Song, J. Lee, S. Rusinkiewicz, and T. Funkhouser, "Spatial action maps for mobile manipulation," in *Proceedings of the Robotics: Science and Systems Conference*, 2020.

[10] R. He, A. Bachrach, and N. Roy, "Efficient planning under uncertainty for a target-tracking micro-aerial vehicle," in *Proceedings of the International Conference on Robotics and Automation*, 2010.

[11] Y. Lee, P. Cai, and D. Hsu, "MAGIC: learning macro-actions for online POMDP planning," in *Proceedings of the Robotics: Science and Systems Conference*, 2021.

[12] G. Theocharous and L. Kaelbling, "Approximate planning in pomdps with macro-actions," in *Advances in Neural Information Processing Systems*, 2004.

[13] C. Amato, G. D. Konidaris, and L. P. Kaelbling, "Planning with macro-actions in decentralized POMDPs," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2014.

[14] C. Amato, G. D. Konidaris, A. Anders, G. Cruz, J. P. How, and L. P. Kaelbling, "Policy search for multi-robot coordination under uncertainty," in *Proceedings of the Robotics: Science and Systems Conference*, 2015.

[15] S. Omidshafiei, A. Agha-mohammadi, C. Amato, S.-Y. Liu, J. P. How, and J. Vian, "Graph-based cross entropy method for solving multi-robot decentralized POMDPs," in *Proceedings of the International Conference on Robotics and Automation*, 2016.

[16] S. Omidshafiei, A. Agha-mohammadi, C. Amato, and J. P. How, "Decentralized control of multi-robot partially observable markov decision processes using belief space macro-actions," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 231–258, 2017.

[17] D. Han, W. Böhmer, M. J. Wooldridge, and A. Rogers, "Multi-agent hierarchical reinforcement learning with dynamic termination," in *PRICAI (2)*, ser. Lecture Notes in Computer Science, vol. 11671. Springer, 2019, pp. 80–92.

[18] O. Nachum, M. Ahn, H. Ponte, S. S. Gu, and V. Kumar, "Multi-agent manipulation via locomotion using hierarchical sim2real," in *Proceedings of the Conference on Robot Learning*, 2019.

[19] Z. Xu, Y. Bai, B. Zhang, D. Li, and G. Fan, "HAVEN: hierarchical cooperative multi-agent reinforcement learning with dual coordination mechanism," *arXiv preprint*, vol. abs/2110.07246, 2021.

[20] J. Yang, I. Borovikov, and H. Zha, "Hierarchical cooperative multi-agent reinforcement learning with skill discovery," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2020.

[21] J. Chakravorty, P. N. Ward, J. Roy, M. Chevalier-Boisvert, S. Basu, A. Lupu, and D. Precup, "Option-critic in cooperative multi-agent systems," *arXiv preprint*, vol. arXiv:1911.12825, 2019.

[22] K. Menda, Y. Chen, J. Grana, J. W. Bono, B. D. Tracey, M. J. Kochenderfer, and D. H. Wolpert, "Deep reinforcement learning for event-driven multi-agent decision processes," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 4, pp. 1259–1268, 2019.

[23] Y. Xiao, J. Hoffman, and C. Amato, "Macro-action-based deep multi-agent reinforcement learning," in *Proceedings of the Conference on Robot Learning*, 2019.

[24] J. Wu, X. Sun, A. Zeng, S. Song, S. Rusinkiewicz, and T. Funkhouser, "Spatial intention maps for multi-agent mobile manipulation," in *Proceedings of the International Conference on Robotics and Automation*, 2021.

[25] F. A. Oliehoek, M. T. J. Spaan, and N. A. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 289–353, 2008.

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

[27] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proceedings of the Conference on Neural Information Processing Systems*, 2000, pp. 1008–1014.