Multi-Agent Asynchronous Cooperation with Hierarchical Reinforcement Learning

Xubo Lyu, Amin Banitalebi-Dehkordi, Mo Chen, Yong Zhang

Abstract. Hierarchical multi-agent reinforcement learning (MARL) has shown a significant learning efficiency by searching policy over higher-level, temporally extended actions (options). However, standard policy gradient-based MARL methods have a difficulty generalizing to option-based scenarios due to the asynchronous executions of multi-agent options. In this work, we propose a mathematical framework to enable policy gradient optimization over asynchronous multi-agent options by adjusting option-based policy distribution as well as trajectory probability. We study our method under a set of multi-agent cooperative setups with varying inter-dependency levels, and evaluate the effectiveness of our method on typical option-based multi-agent cooperation tasks.

Keywords: Multi-Agent Systems; Hierarchical Reinforcement Learning; Asynchronous Cooperation

1 Introduction

Multi-Agent Reinforcement Learning (MARL) enables multiple robots to automatically learn cooperative behaviors via interactions with their environment, and has achieved great success in gaming and robotics \cite{2,5,10,13,14,22}. However, most MARL methods assume robots with actions modelled as primitive operations at lowest-level time scale (e.g. action lasts one time step). Such assumption is inefficient and limits its use on many challenging, real-world multi-robot problems with long-term exploration and delayed, sparse reward \cite{8,11}.

Hierarchical RL (HRL) explores the use of higher-level, temporally extended actions for more efficient policy search on long-term reasoning tasks \cite{3,8,9,18}. As exemplified by the options framework \cite{18}, an option is a higher-level combination (a sequence) of lower-level primitive actions. By optimizing a policy over options instead of over actions, options-based HRL bring significant improvement to policy search efficiency and generalisability \cite{8,21}, especially on highly complex, multi-stage tasks with potentially sparse and delayed feedback. Extending HRL into multi-agent cases attracts more research interests as the learning efficiency becomes more vital when more robots are involved. The use of options for multi-agent system can lead to simpler but more efficient system modelling \cite{11,16}, and can speed up the skill discovery and transferring \cite{7,21}.

Despite the advantages, one primary technical challenge of applying options to a multi-agent system is that decision-making among multiple robots is no longer synchronized. This is because options can have varying time lengths and each agent’s option selection and completion can happen at different time steps.
Fig. 1: An illustration of synchronous and asynchronous decision making. Same color means same decision index. **Left:** synchronous decision making over primitive actions. Both agents choose and terminate actions at identical time step **Right:** Options have varying time lengths. Agent 1’s first option (red) is on-going while agent 2’s first option is finished. $k$ is lower-level time step.

Fig. 1 illustrates option asynchronicity between two agents. In fact, *asynchronicity* is an important aspect for option-based multi-agent system as a realistic multi-agent system has to naturally support agents to perform asynchronously without waiting for other agents’ options to terminate. However, this brings difficulties to many option-based MARL methods since it is hard to determine what information to use, as well as when and how to update agents’ policies [20].

One potential solution is to roughly synchronize [4, 21] by interrupting or extending options to align their time lengths for all agents, but this will either destroy the completeness of option or sacrifice data efficiency, and lose the benefits of temporal abstraction over actions. Christopher et. al [1, 19] propose MacDec-POMDP, a option-based MARL framework and use deep RL approach with asynchronous data alignment to learn hierarchical policy without interrupting the robot-environment interactions. But they mainly uses value-based RL, and has limited discussion on more possible multi-agent cooperative setups.

In this work, we propose a novel but simple mathematical framework that allows policy gradient-based hierarchical RL with asynchronous multi-agent options. This is achieved by first adjusting option-based policy distribution as well as trajectory probability according to a particular multi-agent cooperation mode, and then utilizing this trajectory probability along with option-wise samples to conduct standard policy gradient optimization. By seamlessly transforming off-the-shelf policy gradient-based MARL methods from action-based to option-based, our method is highly adaptive and can work with a variety of multi-agent cooperative learning schemes (e.g. decentralized or centralized training). Moreover, our method supports both discrete and continuous option space, and will not affect the robot-environment interactions.

We evaluate our method on a set of multi-agent cooperative tasks with long-term reasoning properties and observe significant improvement in learning efficiency. The results indicate our method has good scalability to environment with more agents, as well as compatibility with discrete/continuous option space.

The main contributions of our work can be summarized as:

- We provide an approach to enable hierarchical multi-agent policy gradient RL over asynchronous options.
- We conduct a comprehensive study of our approach under a collective of multi-agent cooperative modes.
We justify the effectiveness of our method on several representative option-based multi-agent cooperative tasks with varying number of agents.

2 Preliminaries

2.1 Hierarchical Multi-Agent RL

The fully cooperative multi-agent RL (MARL) problem can be described by a Decentralized Partially Observable Markov Decision Process (DEC-POMDP) [12], defined by $(S, A, Z, R, P, n, \lambda)$. $S$ is the state space with element $s \in S$ as global state. Parts of the action space $A$ can be shared for homogeneous agents, or vary among heterogeneous agents, denoted as $A_i$ for each agent $i$. $Z$ is observation space with element $z_i \in Z$ where $z_i = Z(s; i)$ is local observation for agent $i$ at global state $s$. $P(s'|s, A)$ denotes the transition probability from $s$ to $s'$ given the joint action selection $A = (a_1, ..., a_n)$ for all $n$ agents. Note that in the standard MARL setting described by DEC-POMDP, $a_i$ is usually a primitive action, a.k.a., the action executed at the lowest-level temporal scale. This is in contrast with hierarchical MARL [11] where the agent can instead take an option that may last for multiple primitive steps. All agents share same reward function $R(s, A)$. $\lambda$ is the discount factor.

By replacing primitive actions with options under a multi-agent setup, we have option-based hierarchical MARL. The options framework [13] introduces a generalization of primitive actions to include temporally extended actions. An option $o$ is defined as $\langle I^o, \beta^o, \pi^o \rangle$, where $I^o \subset S$ is the initiation set, $\beta^o : S \to [0, 1]$ is the stochastic option termination condition and $\pi^o : S \times A \to [0, 1]$ is a policy that selects a sequence of primitive actions to achieve the target, or objective, of an option. On reaching the termination condition in state $s'$, an agent can select a new option from the set $O(s') := \{o \mid s' \in I^o\}$. The use of options establishes a hierarchical decision making structure. With the notation $\langle I^{o_i}, \beta^{o_i}, \pi^{o_i} \rangle$ representing agent $i$ taking option $o_i$, we aim to find a high-level policy over option space to appropriately chooses options in order to maximize the long-term accumulated reward.

2.2 Asynchronous Multi-Agent Option Execution

An essential aspect of option-based multi-agent systems is the asynchronicity of option execution over agents, because when a group of agents perform options, their options may not terminate at the same time due to their varying time lengths. MARL methods designed for DEC-POMDPs fail in this case as they are all based on primitive actions that are synchronized by nature. Makar et.al [11] categorizes termination strategies to be synchronous and asynchronous. Typical synchronous strategies such as $\tau_{any}$ and $\tau_{all}$ force all agents’ options to be synchronized with the earliest ($\tau_{any}$) or latest ($\tau_{all}$) completed option, by either interrupting the unfinished options or waiting for the all options to finish. The synchronous schemes destroy the completeness of options and lose the benefits of temporal abstraction and can bring unstable training outcome. Typical

\footnote{not to be confused with “observation”, which is also often denoted $o$ in related papers}
asynchronous strategies such as $\tau_{\text{continue}}$ allow a subset of the agents to choose new options at each decision epoch while the others continue their old options. In this work, we use $\tau_{\text{continue}}$ strategy that allows training data collection without interrupting multi-agent option execution.

2.3 Option-Based Multi-Agent Policy Gradient Optimization

Policy gradient-based MARL methods, such as MAPPO \cite{22}, directly optimize a policy with gradients calculated from collected trajectories, and enjoys unbiased and stable learning especially for policies with high-dimensional input. However, these methods are mostly designed for multiple agents with synchronized “one-step” actions and cannot be directly applied to MARL scenarios with asynchronous options.

Here we take a well-known, GAE-Based \cite{15} policy gradient optimization shown in Eq. (1) as an example. Eq. (1) is defined upon low-level time scale $k$ and calculates policy gradient of RL objective $J(\pi_{\theta_i})$ over a batch of trajectories $\tau$. $G_k^\lambda$ is TD($\lambda$) return \cite{17} and $V_{\pi_{\theta_i}}$ is the value baseline. $i$ represents $i$th agent.

$$
\nabla \theta J(\pi_{\theta_i}) = E_{\tau \sim \pi_{\theta_i}} \left[ \sum_{k=0}^{K} \nabla \theta_i \log \pi_{\theta_i}(a_k | s_k) \left( G_k^\lambda - V_{\pi_{\theta_i}}(s_k) \right) \right].
$$

Despite the use of low-level actions $a_t$ in Eq. (1), we can easily figure out a trivial way to adapt Eq. (1) from action-wise to option-wise by simply substituting action-based policy $\pi(a|s)$ with option-based policy $\pi(o|s)$ and viewing an option as an action with “one huge step”. In this way, Eq. (1) still holds but is not on the basis of primitive time $k$ but option time $t$ defined by the option termination. However, this method requires multi-agent options to be rigorously synchronized ($\tau_{\text{any}}$ or $\tau_{\text{all}}$) and destroys the completeness of options. In this work, we propose a novel framework to adjust $\pi(o|s)$ and make use of Eq. (1) for option-based policy gradient calculation while maintain the asynchronicity of multi-agent option selections.

3 Problem Formulation

We formulate our problem in a fully cooperative setting that can be described by an option-based DEC-POMDP \cite{12}, in which we consider a multi-agent task environment where $N$ robots aim to collaboratively finish a task and optimize a share reward. For each agent $i$ given current local observation $z_t^i$, it can choose an option $o_t^i$ from a finite option set $O_i$. Note that in general one can also take advantage of a history of observations for option-based decision making \cite{19}.

In the whole work, we use $t$ to denote option-level time step and $k$ to denote low-level, primitive time step. At each time step $k$, while multiple agents either determine or continue to execute their higher-level options, the environment transitions to the next state at a lower-level according to the transition function $P(s' | s, A)$ where $A = (a_1, a_2, ..., a_N)$ represents a set of primitive actions selected by multi-agent options $(\pi_1^o, \pi_2^o, ..., \pi_N^o)$ at the current time step $k$. In our problem, we do not assume that the transition function is known.
While it is possible to automatically learn a set of reasonable options for each agent \([3][21]\), in this paper, we assume the options are given or pre-defined and are appropriate for completing the given task. Moreover, the termination function \(\beta_{A_i}\) for each agent \(i\) is deterministic and only terminates when encounters specific underlying states. The options framework generally supports hierarchical decision structures with multiple layers of options. In our work, we assume a two-level hierarchy with a higher-level option space and lower-level primitive action space as it is sufficient to illustrate the core idea of our approach.

4 Modes of Multi-Agent Cooperation

Hierarchical multi-agent reinforcement learning (MARL) often involves the use of asynchronous options. However, many policy gradient-based MARL methods are designed for synchronous, primitive actions and is not compatible with option-based MARL problems. In this section, we introduce our framework for asynchronous option-based policy gradient optimization. In particular, we establish our framework towards four important multi-agent cooperative modes associated with centralized and decentralized training schemes. This section mainly examines policy distribution adjustment at a single, option-level time step \(t\). The conclusion is extended to trajectory-based policy gradient calculation in Sec. \([5]\).

Without loss of generality, for all cases, we consider the situation at option-level time step \(t\) when any one of the agents terminates its previous option. Note that this is unlike the synchronized termination \(\tau_{\text{any}}\) as we do not interrupt agents’ ongoing options. Suppose a subset of agents \(U_i\) complete their previous options and need to choose new options at time step \(t\) while a complimentary set of agents \(\overline{U}_i\) continue their ongoing options. This is a reasonable assumption as options between multiple agents may execute asynchronously.

4.1 Fully-Decentralized

Decentralized learning assumes each agent only has access to its own options and local observations, and the option selection is only dependent on its own local observation. With the decentralized training setup, we have \(N\) policies \(\pi_{\theta_i}\), one for each agent. Note that this \(\pi_{\theta_i}\) is the policy of agent \(i\) over options, rather than primitive actions, and takes the local observation \(z_i\) as input. \(\theta_i\) is the policy parameters (e.g. neural network weights). Such \(N\) policies can support either homogeneous with identical observation and option space or heterogeneous agents with different observation and option spaces.

Due to varying temporal length of options, it is unlikely for all agents to naturally synchronize and always choose options at the same lower-level time steps. Instead, it is more realistic that a subset of agents complete their own options and terminate, while other agents do not and continue executing their ongoing option. At time step \(t\), we have the option selection probability:

\[
\pi_{\theta_i}(o_i^t | z_i^t) = \begin{cases} 
1, & \text{if } i \notin U^t \\
\pi_{\theta_i}(o_i^t | z_i^t), & \text{otherwise.}
\end{cases}
\]
This is analogous to the action probability in a standard RL setting, except that the agents which are continuing to execute their options have an option selection probability of 1. Moreover, this mode has variants depending on the use of one centralized value function \( V_\phi(z_t) \) (centralized training decentralized execution, CTDE \[22\]) or multiple decentralized value functions \( V_\phi_i(z_t^i) \) in training. \( z_t \) is joint-observation over all agents and \( \phi \) is function parameters. But Eq. \[2\] holds for both variants as value functions are independent from options.

4.2 Fully-Centralized

A centralized policy takes as input the joint local observations of all agents, and selects a joint option for all agents. Under this setting, we assume to have one centralized policy \( \pi(o_t | z_t) \) where \( z_t \) is the joint observation over all agents’ local observation at time step \( t: z_t = (z^t_0, z^t_1, \ldots, z^t_{N-1}) \); and \( o_t \) is the joint option: \( o_t = (o^t_0, o^t_1, \ldots, o^t_{N-1}) \).

Unlike the decentralized case where we hold the assumption that the options are independently selected as the options among agents, here we formulate the conditional option selection. That is, we condition the option distribution of the agents in \( U_t \) on the currently executing options of agents in \( \overline{U}^t \):

\[
\pi_\theta \left( o^t_U \mid o^t_{\overline{U}}, z^t_t \right) = \frac{\pi_\theta \left( o^t_U, o^t_{\overline{U}} \mid z^t_t \right)}{\pi_\theta \left( o^t_{\overline{U}} \mid z^t_t \right)}, \tag{3}
\]

where \( o^t_U = [o^t_{u_0}, o^t_{u_1}, ..., o^t_{u|U|}] \) is the joint option at time step \( t \) for a subset of agents in \( U \), and \( o^t_{\overline{U}} \) is the joint option for agents in \( \overline{U}^t \). Eq. \[3\] can be easily calculated since we have access to the joint option distribution \( \pi_\theta \left( o^t_U, o^t_{\overline{U}} \mid z^t_t \right) \) (e.g. direct output of \( \pi_\theta \)), and \( \pi_\theta \left( o^t_{\overline{U}} \mid z^t_t \right) \) can be obtained via marginalization.

4.3 Partially-Decentralized

Inspired from fully-centralized learning where a joint-option is selected in a conditional way based on joint-observations, we propose partially-decentralized cooperation which describes agents choose their options in a conditional way, but only based on their local observations. This mode is especially useful in a realistic situations when multi-agent options are highly dependent yet each agent has limited access to the global information (or the joint-observations is too highly-dimensional to be used as policy input). In this mode, we use multiple policies \( \pi_i \), one for each agent \( i \) for option selection. At time step \( t \), similar to fully-centralized learning, we formulate the conditional option selection by conditioning the option distribution of the agents in \( U^t \) on the currently executing options of agents in \( \overline{U}^t \):

\[
\pi_{\theta_i} \left( o^t_i \mid o^t_{\overline{U_t}}, z^t_t \right) = \begin{cases} 1, & \text{if } i \in \overline{U}^t \\ \frac{\pi_{o_i} \left( o^t_i, o^t_{\overline{U}} \mid z^t_t \right)}{\pi_{\theta_i} \left( o^t_{\overline{U_t}} \mid z^t_t \right)}, & \text{otherwise}. \end{cases} \tag{4}
\]
In contrast to fully-centralized mode, here we conditionally select a local option for each agent instead of a joint-option for all agents.

### 4.4 Partially-Centralized

Compared to the fully-centralized case, the partially-centralized case describes a specific multi-agent cooperation situation where each agent’s option selection is not only dependent upon its own local observation, but also dependent upon the global, joint observations $z_t$ from all other agents. This models a real-world multi-robot cooperation scenario that local observations can be broadcast among agents. This way, every robot has access to the observations of others. In this case, we use multiple policies $\pi_i$, one for each agent $i$ for option selection. Similar to fully-decentralized mode, $o^t_i$ here is independent from the option selected by the other agents. Therefore, the option selection probability at time step $t$ is the same as the fully-decentralized case, except that the option is conditioned on the joint observation, as shown below. As before, $U^t$ is a set of agents executing ongoing options at time step $t$.

$$\pi_{\theta_i}(o^t_i \mid z^t) = \begin{cases} 1, & \text{if } i \notin U^t \\ \pi_{\theta_i}(o^t_i \mid z^t), & \text{otherwise}. \end{cases}$$

### 5 Option-Based Multi-Agent Policy Gradient

In Sec. 4, we study the policy distribution adjustment for asynchronous option selection at a single, option-level time step $t$. These results can be naturally extended to trajectory-wise by considering a sequence of time steps. As a result, it can be seamlessly adapted into standard policy gradient calculation in Eq. (1).

![Fig. 2: An example option-based trajectory for two agents in our approach. $k$ is lower-level time step while $t$ is higher-level time step. We adjust policy distribution and collect training samples for both agents only at each $t$.](image)

Our method uses asynchronous option-based trajectory for multiple agents. As is shown in Fig. 2 for agent $i$, we collect training trajectory at option-level: $\tau_i = (z^{t_0}_i, o^{t_0}_i, R(z^{t_0}_i), ..., z^{t_T}_i)$ where $t_T$ is the last option-level time step in an episode. Note that this is unlike to standard MARL where samples are synchronously collected at lowest action-level.

In the following, we generalize the single-step policy distribution adjustment as well as option-level training trajectories to establish option-Based multi-agent policy gradient towards four multi-agent cooperation modes.
5.1 Decentralized

**Fully-Decentralized.** For decentralized case with \( N \) agents and \( N \) associated policies, consider we have a trajectory \( \tau_i = (z_{i0}^t, o_{i0}^t, R(z_{i0}^t), ..., z_{iT}^t) \). Each \( \tau_i \) is generated by option-level local policy \( \pi_{\theta_i} \). Following the standard Eq. (1), we have an adapted version as shown in Eq. (6), where \( \pi_{\theta_i}(o_{it} | z_{it}) \) refers to Eq. (2).

\[
\nabla_{\theta_i} J(\pi_{\theta_i}) = E_{\tau_i \sim \pi_{\theta_i}} \left[ \sum_{t=0}^{T} (\nabla_{\theta_i} \log \pi_{\theta_i}(o_{it}^t | z_{it}^t)) \left( G_{i}^t - V_{\phi_i}(z_{it}^t) \right) \right]. \tag{6}
\]

**Partially-Decentralized.** For partially-decentralized case, with the same type of option-level trajectories as fully-decentralized case, we have the corresponding policy gradient formula as Eq. (7), where \( \pi_{\theta_i}(o_{it}^t | o_{it}^{\mathcal{U}}, z_{it}^t) \) refers to Eq. (4).

\[
\nabla_{\theta_i} J(\pi_{\theta_i}) = E_{\tau_i \sim \pi_{\theta_i}} \left[ \sum_{t=0}^{T} (\nabla_{\theta_i} \log \pi_{\theta_i}(o_{it}^t | o_{it}^{\mathcal{U}}, z_{it}^t)) \left( G_{i}^t - V_{\phi_i}(z_{it}^t) \right) \right]. \tag{7}
\]

5.2 Centralized

**Fully-Centralized.** For centralized learning with \( N \) agents and a shared policy, if we have a joint trajectory \( \tau \), and use Eq. (8) as adapted version for policy gradient calculation over asynchronous options, where \( \pi_{\theta}(o_{t}^{\mathcal{U}}, o_{t}^{\bar{\mathcal{U}}}, z_{t}) \) refers to Eq. (3). Note that \( o_{t}^{\mathcal{U}} = (o_{t_0}^{\mathcal{U}}, o_{t_1}^{\mathcal{U}}, ..., o_{t_N}^{\mathcal{U}} | u_0, u_1, ..., u_N \in \mathcal{U}) \) which is concatenation of options for multiple agents in set \( \mathcal{U} \). And similarly \( o_{t}^{\bar{\mathcal{U}}} = [o_{u_0}^{\bar{\mathcal{U}}}, o_{u_1}^{\bar{\mathcal{U}}}, ..., o_{u_N}^{\bar{\mathcal{U}}}] \) represents a concatenation of ongoing options from agents in set \( \bar{\mathcal{U}} \).

\[
\nabla_{\theta} J(\pi_{\theta}) = E_{\tau \sim \pi_{\theta}} \left[ \sum_{t=0}^{T} (\nabla_{\theta} \log \pi_{\theta}(o_{t}^{\mathcal{U}}, o_{t}^{\bar{\mathcal{U}}}, z_{t})) \left( G_{i}^t - V_{\phi}(z_{t}) \right) \right]. \tag{8}
\]

**Partially-Centralized.** For partially-centralized training setup, the adapted version of trajectory-based policy gradient calculation is similar to that of fully-decentralized case, as shown in Eq. (9). This is because they share same assumption that each agent’s option selection is independent from the others’ options. \( \pi_{\theta_i}(o_{it}^t | z_{it}^t) \) refers to Eq. (5).

\[
\nabla_{\theta_i} J(\pi_{\theta_i}) = E_{\tau_i \sim \pi_{\theta_i}} \left[ \sum_{t=0}^{T} (\nabla_{\theta_i} \log \pi_{\theta_i}(o_{it}^t | z_{it}^t)) \left( G_{i}^t - V_{\phi_i}(z_{it}^t) \right) \right]. \tag{9}
\]

6 Experimental Results and Discussion

We run simulated experiments on typical cooperative multi-robot tasks, and provides a variety of qualitative and quantitative results to evaluate our method. In these experiments, we mainly answer the following questions:

1. Does our approach work well in general with multi-agent policy gradient optimization using asynchronous options?
2. What effect will it bring on policy learning given the use of different cooperative training paradigms?
3. Does the training using options boost the learning efficiency and performance compared to using actions?
4. Does the asynchronous training in our approach has advantages over the synchronous counterpart?

Notations: In the following, we use abbreviation forms to name all paradigms (e.g. fully-dec corresponds to fully-decentralized learning, etc.). Especially fully-dec-CTDE represents a variant of fully-dec that uses centralized value function.

6.1 Task Specifications

Water Filling. We experiment with an large indoor environment with two agents and three water containers. Two agents are heterogeneous with one of them a slower ground vehicle capable of navigate, visualize and fill a container with water, while the other a faster drone and only capable of navigate and visualize but cannot fill water. The key to finish this task requires two agents collaborate accordingly and leverage their respective ability (e.g. drone for quick exploration; vehicle for manipulation) to refill the water timely. The water levels in containers decrease randomly, unpredictably at each time step. This is to make sure that two agents will end up with “real” multi-agent cooperation rather any trivial behavior repetition. We simulate it using Ai2thor simulator [6]. We design this task as it can be a miniature for many potential scenarios for indoor or outdoor service robots. An illustration of this task setup is shown in Fig. 3.

We take local RGB image and transform it to a low-dimensional visual embedding as part of the observation. Observation also includes important auxiliary information: the water levels for all containers that has been explored previously; and the total time steps elapsed until the water levels get updated. Available options for the mobile and Fetch robot can be found in Table. 1. The reward function used in this task is shown in Eq. (10), where \( w(s; c) \) is the water level of the \( c \)-th container by knowing the true state \( s \). We use the multi-agent PPO algorithm [22] but adapt it to the option-based policy gradient according to our method described in Sec. 5.

\[
    r(s) = \sum_{c=0}^{\mid C \mid} \left( -\frac{1.0}{w(s; c) + 0.001} + 1 \right), c = 0, 1, 2. \tag{10}
\]

Tool Delivery. Tool delivery [19] is a three agents, human-robot interaction task requiring more complicated collaborations and long-term reasoning, as shown in Figure. 3b. It involves one human working at the workshop and needs assistance from a Fetch robot and two mobile robots to search, pass and deliver particular tools in time to finish a multi-stage assembly task.

Each mobile robot has access to local observations including its location; observable human working step; index of carried tools and the number of the tools at the waiting spots while the Fetch robot observes the number of tools waiting to be passed and which mobile robot is waiting beside. Note that neither
Fetch nor mobile robots is aware of the correct tool required by human at each step, such that the robots have to reason about this information via training. Available options for the mobile and Fetch robot can be found in Table 1. The agents receive $-1$ as step cost, $-10$ when the fetch pass a tool to an empty waiting spot and $+100$ for a successful tool delivery to human.

$$r(s) = -1 + 100 \times \mathbb{I}_{\text{GoodDelivery}} + (-10) \times \mathbb{I}_{\text{BadPass}}.$$  \hspace{1cm} (11)

Fig. 3: (a) Water-Fill task. Left: top view; right: front view. Yellow colors represent water levels; white, pink and green circles are locations of three containers. Two robots are located in the middle space (gray color). (b) Tool delivery task.

| Task   | Agent   | Options                                      |
|--------|---------|----------------------------------------------|
| WF     | Drone   | Wait; Up; Down; Left; Right; NavTo(i);       |
| WF     | Vehicle | Wait; Up; Down; Left; Right; NavTo(i); NavToFill(i) |
| TD     | MobileBot | GoToWS; GoToTR; GetTool                      |
| TD     | Fetch   | Wait; SearchTool(j); PassTo(k)               |

Table 1: Available agent options for Water Filling (WF) and Tool Delivery (TD) tasks. $i \in \{\text{Cup, Bottle, Mug}\}$; $j \in \{0,1,2\}$ represents the tool index and $k \in \{0,1\}$ denotes the index of waiting mobile robot. Options could have varying time lengths depending on the termination condition and task property.

6.2 Results: Overall Performance

To justify the effectiveness and wide applicability of our method, we use it to train the policy under different paradigms and evaluate the policy on two option-based multi-agent cooperation tasks. These training paradigms involve varying levels of distributedness and interdependence over multiple agents and can be a good indicator to verify the generalisability of our approach.

Fig. 4 shows the quantitative results in terms of average step reward versus training steps on two tasks. Fig. 4a records the actual training progress for
| Methods       | Initial Perf  | Evaluated Perf | Empty Duration(%) |
|---------------|---------------|----------------|-------------------|
| fully-dec    | $-1224 \pm 87.3$ | $-316.4 \pm 29.3$ | 85.3 → 29.3       |
| fully-dec (CTDE) | $-1029.6 \pm 232.5$ | $-245.3 \pm 53.8$ | 78.2 → 27.6       |
| fully-cen    | $-698.2 \pm 75.1$ | $-247.5 \pm 9.5$  | 76.2 → 27.1       |
| partially-cen| $-1267.5 \pm 64.5$ | $-345.5 \pm 33.5$ | 86.5 → 26.9       |
| partially-dec| $-778.1 \pm 102.8$ | $-259.1 \pm 28.8$ | 79.7 → 24.2       |

Table 2: Policy evaluation using our approach with five training paradigms on water-filling task. Data is averaged over five runs with different seeds.

the water-filling task. Our approach achieves constantly increasing reward until convergence regardless of training paradigms. Similar conclusion can be drawn from table. 2 where we evaluate the trained policy as well as the total time steps that any container staying empty as a task-specific characteristics. With all cooperative setups, our method finally learns a policy receiving much higher reward and less empty duration indicating the effectiveness of our method on two-agent water-filling task.

Fig. 4b shows periodic policy evaluation along the training process for the three-agent tool-delivery task. Our method acquires policy improvement under all cooperative training paradigms, but with varying boosting levels compared to the results from two-agent task. This is potentially due to additional task complexity brought by more agents and task stages. In particular, centralized-based learning in general achieves much better results than decentralized learning. This conforms to the conclusions from previous literature [22], as centralized learning has access to more global information to help agents make a better decision.

(a) Water-filling task

(b) Tool-delivery task

Fig. 4: Overall policy learning performance using our approach with multiple cooperative training paradigms on two typical tasks. Averaged step rewards are calculated over a batch of samples and training steps indicate the total number of samples collected from the environment as training proceeds.
6.3 Results: Cross Comparison

To further investigate what effect a particular training paradigm will impose on option-based MARL, we conduct a cross comparison between the fully and partially learning setup from the training result of tool-delivery task. Fig. 5 compares fully-dec with partially-cen and partially-dec respectively. As is shown in Fig. 5a, fully-dec (CTDE) shares similar learning result with partially-cen, but performs much better than fully-dec. This suggests that when multiple agents use decentralized policies and choose their local options, having access to more global information (e.g. fully-dec CTDE has centralized value function; partially-cen policies has joint observations as input) is vital and helpful to better cooperation. When it comes to Fig. 5b, partially-dec performs worse than fully-dec (CTDE) even though they both use centralized value function. Such result implies that only having local observations may be not sufficient to select good options for the other agents, which is what partially-dec does.

In Fig. 6, we compare fully-cen with partially-cen and partially-dec respectively. Fully-cen and partially-cen shares the property of taking joint observations into account for multi-agent option selection. However, fully-cen encodes one shared, centralized policy for all agents while partially-cen uses decentralized policy to select local options. Fig. 6a indicates that regarding option-based multi-agent cooperation, using centralized policy for joint-option selection is advantageous over using decentralized policies. In Fig. 6b, the lack of global information especially the observations from other agents as well as the use of decentralized policies give rise to the inferior learning performance of partially-dec paradigm.

6.4 Results: Option v.s. Primitive Action

The use of primitive actions is common for MARL. In this work, we use the name “end-to-end learning (end2end)” to represent the multi-agent policy learning with actions instead of options. We conduct the training on water-filling task...
using ours and end-to-end approach. Our method aims to learn a hierarchical policy over asynchronous options while the end-to-end training learns a regular policy over one-step action. For both methods, we use them under the “fully-dec” training paradigm. As shown in Fig. 7a, our method constantly attains much higher averaged reward than “end2end” and ends up with substantial improvement on learning efficiency and peak performance. Note that this is probably not a strictly fair comparison as the options may encode more useful, task-specific knowledge than primitive actions. However, we emphasize that instead learning from scratch over actions, our method is an efficient alternative if appropriate options can be easily derived or pre-defined.

6.5 Results: Asynchronous v.s. Synchronous

Asynchronous and synchronous terminations are two main strategies used for option-based hierarchical policy learning. Specifically, we consider two types of synchronous methods: “sync-wait” and “sync-cut”, and compare their training performances with our method. “Sync-wait” synchronizes multi-agent options by waiting for the latest completed option while “sync-cut” interrupts the other unfinished options when one of the options finishes. For all methods, we use them under the “fully-dec” training paradigm. We evaluate the averaged reward collected by final policy and measure the training time steps required to reach same performance level. As depicted in Fig. 7b, our asynchronous approach acquires the highest reward at convergence (left-axis) and best learning efficiency (right-axis) in both tasks. On the other hand, “sync-wait” method eventually performs moderately but is five times slower than our method in term of learning speed. “sync-cut” method can sometimes learn a good policy (tool-delivery) but can fail catastrophically in task with long-range options (water-filling).
7 Conclusion

In this work, we propose a mathematical framework to generalize policy gradient-based hierarchical policy learning with asynchronous options. We do this by first adjusting option-based policy distribution as well as trajectory probability according to different multi-agent cooperative modes, and then utilizing this trajectory probability along with option-wise samples to conduct standard policy gradient optimization. Our method is highly adaptive and can work with a variety of multi-agent cooperative learning schemes. It seamlessly transforming off-the-shelf multi-agent policy gradient methods from action-based to option-based without affecting the asynchronous robot-environment interactions. With our method, we not only preserve the learning efficiency of using temporally-extended actions, but also inherits the benefits of policy gradient methods on more stable and flexible learning.
References

1. Amato, C., Konidaris, G., Kaelbling, L.P., How, J.P.: Modeling and planning with macro-actions in decentralized pomdps. Journal of Artificial Intelligence Research 64, 817–859 (2019)
2. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
3. Frans, K., Ho, J., Chen, X., Abbeel, P., Schulman, J.: Meta learning shared hierarchies. In: International Conference on Learning Representations (2018)
4. Han, D., Boehmer, W., Wooldridge, M., Rogers, A.: Multi-agent hierarchical reinforcement learning with dynamic termination. In: Pacific Rim International Conference on Artificial Intelligence. pp. 80–92. Springer (2019)
5. Jain, U., Weihs, L., Kolve, E., Farhadi, A., Lazebnik, S., Kembhavi, A., Schwing, A.: A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 471–490. Springer International Publishing, Cham (2020)
6. Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., Farhadi, A.: Ai2-thor: An interactive 3d environment for visual ai. arXiv preprint arXiv:1712.05474 (2017)
7. Konidaris, G.D., Barto, A.G.: Building portable options: Skill transfer in reinforcement learning. In: IJCAI. vol. 7, pp. 895–900 (2007)
8. Kulkarni, T.D., Narasimhan, K., Saeedi, A., Tenenbaum, J.: Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. Advances in neural information processing systems 29, 3675–3683 (2016)
9. Li, P., Huang, H., Hu, Y., Wu, X., He, R., Sun, Z.: Hierarchical face aging through disentangled latent characteristics. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 86–101. Springer International Publishing, Cham (2020)
10. Liang, X., Wang, T., Yang, L., Xing, E.: CirL: Controllable imitative reinforcement learning for vision-based self-driving. In: The European Conference on Computer Vision (ECCV) (September 2018)
11. Makar, R., Mahadevan, S., Ghavamzadeh, M.: Hierarchical multi-agent reinforcement learning. In: Proceedings of the fifth international conference on Autonomous agents. pp. 246–253 (2001)
12. Oliehoek, F.A., Amato, C.: A concise introduction to decentralized POMDPs. Springer (2016)
13. Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., Whiteson, S.: Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: International Conference on Machine Learning. pp. 4295–4304. PMLR (2018)
14. Ren, L., Lu, J., Wang, Z., Tian, Q., Zhou, J.: Collaborative deep reinforcement learning for multi-object tracking. In: The European Conference on Computer Vision (ECCV) (September 2018)
15. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
16. Stone, P., Sutton, R.S., Kuhlmann, G.: Reinforcement learning for robocup soccer keepaway. Adaptive Behavior 13(3), 165–188 (2005)
17. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine learning 3(1), 9–44 (1988)
18. Sutton, R.S., Precup, D., Singh, S.: Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial intelligence 112(1-2), 181–211 (1999)

19. Xiao, Y., Hoffman, J., Amato, C.: Macro-action-based deep multi-agent reinforcement learning. In: Kaelbling, L.P., Kragic, D., Sugiyama, K. (eds.) Proceedings of the Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 100, pp. 1146–1161. PMLR (30 Oct–01 Nov 2020), https://proceedings.mlr.press/v100/xiao20a.html

20. Xiao, Y., Tan, W., Amato, C.: Asynchronous multi-agent actor-critic with macro-actions (2022), https://openreview.net/forum?id=wQStfB93RZZ

21. Yang, J., Borovikov, I., Zha, H.: Hierarchical cooperative multi-agent reinforcement learning with skill discovery. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. pp. 1566–1574 (2020)

22. Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., Wu, Y.: The surprising effectiveness of ppo in cooperative, multi-agent games. arXiv preprint arXiv:2103.01955 (2021)
8 Supplementary Materials

8.1 Code and Video Demos

Please find our code in two folders: code and code_wf. The README.md file contains the instructions to reproduce our experiments results.

We also include a few video demos showing the behaviors of trained policies to demonstrate the effectiveness of our approach.

1. wf_frames_firstperson.mp4; wf_frames_topdown.mp4: policy evaluation on water-filling task (first-person and top-down view). Two agents cooperate to timely explore and fill water for three containers before they become empty.
2. td_frames.mp4: policy evaluation on tool-delivery task. Three agents cooperate to search, pass, delivery tools to human to accomplish long-term, multi-stage assembly task.
3. ct_frames.mp4: policy evaluation on capture-target task. Two agents cooperate to achieve simultaneous target arrival.

8.2 More Results and Discussions

Fig. 8: A simulated sequence generate by a trained hierarchical policy on water-filling task. The faster robot (a drone, blue diamond) explores more quickly and informs the slower robot (a vehicle, red triangle) to reach and refill the green container since it is almost empty.

Visualization Analysis We record typical roll-outs in the simulated environment with a trained policy. From agents’ behaviors, the trained hierarchical policy does a good job on coordinating each agent to play their own role depending on their abilities. For example, in Fig. 8, the faster robot explores larger area more quickly in order to be aware of the global water levels timely. However, due to its lack of manipulation ability, it needs to communicate with the slower robot so that it can move the container that requires water-refilling.

Similarly in Fig. 9, the faster agent finds out which container needs to be refilled and lets the slower manipulator know. Then the slower agent goes to help refill water, and at the same time the faster agent leaves to explore the other locations to have an “ahead of time” awareness of the environment.

Figure 10 demonstrates a option-based policy controls three heterogeneous agents to collaborate for tool delivery. The Fetch robot learns to successively
Fig. 9: A simulated sequence generated by a trained hierarchical policy from different initial positions on a water-filling task. The drone explores and finds out the pink container it is almost empty. Thus it informs the vehicle to refill it as the drone cannot refill. Then the drone leaves and explores the other containers with its faster speed in case of any of them are getting empty.

Fig. 10: A simulated sequence generated by a trained hierarchical policy on a tool-delivery task. Fetch robot successively searches and passes tools to two mobile robots to be delivered to the human workshop.

search and pass available tools when there are awaiting agents. As for mobile robots, they learn to navigate to the human workshop instantly after they obtain the tools, and come back to the waiting spot for receiving more tools for the next round delivery. By properly coordinating the underlying timing in this task, three robots successfully accomplish the multi-stages task using appropriate options.

Degree of Partially-Decentralized Learning In this section, we further discuss a particular multi-agent cooperation mode: partially-decentralized learning. We study it as it potentially includes different levels of inter-dependency in terms of multi-agent option selection, which can result in different levels of training performance. We particularly study two cases: (1) higher-degree and (2) lower-degree partially-decentralized (Partially-Dec) modes. Higher-degree Partially-Dec requires conditional option selection for every agent while lower-degree Partially-Dec allows some of the agents to have independent option selection.

An agent conditionally selects an option if it uses a policy distribution conditioned by the on-going options from the other agents, as what they do in Fully-Cen mode. Instead, an agent independently selects an option if it does not depend on the other agents’ on-going options, as what they do in the Fully-Dec mode. By allowing various numbers of agents using independent option selection, we have various degrees of Partially-Dec. For example, in the tool-delivery task, we have three agents taking asynchronous options to accomplish the task. We denote “higher-degree” as all three agents select their own options conditionally, and denote “lower-degree” as two agents (MobileBots) select options independently but one the other agent (Fetch) selects option conditionally.
Fig. 11: (a) Performance comparison in terms of averaged reward on Tool-Delivery task between higher-degree and lower-degree Partially-Dec. (b) A view of Capture-Target task. (c) Performance of our approach under Fully-Cen mode on Capture-Target task with continuous options

Fig. 11a shows the training performance comparison between “higher-degree” and “lower-degree” Partially-Dec. The “lower-degree” Partially-Dec achieves much better averaged reward than “higher-degree” along the training process. The result indicates that too much conditional option selection probably is not good for multi-agent partially-decentralized learning. This is potentially because it is really difficult for each agent to provide a good policy distribution conditioned by the others’ options without knowing any information from the other agents (in Partially-Dec, each agent only has access to its own observation). Therefore, by decreasing the level of conditional option selection via allowing more independent agents, the “lower-degree” Partially-Dec performs better.

Continuous Option Space We further study if our framework works with continuous option space. Continuous option is rarely investigated in prior literature and it is a type of option that can be sampled from a continuous distribution (e.g. Gaussian distribution). For example, “go to a place with constant speed $v$” is a continuous option depending on the choice of a continuous variable $v$.

We design an experiment called “Capture Target” as shown in Fig. 11b. It describes an environment with two agents and one target landmark. The target landmark is randomly positioned and kept static in each training episode, and the two agents are expected to reach the target simultaneously by adjusting their moving speed. Each agent has access to their own location, speed, the other agent’s location as well as the relative location of the target. For agent $i$ ($i \in \{1, 2\}$), it chooses a continuous option “go to target with speed $v_i$” where $v_i$ is a continuous variable and stays constant during the option execution. To reinforce the simultaneous arrival of two agents towards the target, we use combinational distance from two agents to the target as reward shown in Eq. (12), where $d(s, i)$
is the distance between agent $i$ and target landmark under the true state $s$:

$$r(s) = \sum_{i=1}^{2} d(s; i). \quad (12)$$

For this task, we in particular use fully-centralized (Fully-Cen) training mode, and model the option selection of both agents with a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. We utilize one shared policy network $\pi_\theta$ which receives the joint-observations from both agents and predicts $\mu = (\mu_1, \mu_2)$. We use a fixed co-variance matrix $\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} = \begin{bmatrix} 1 & \sigma \\ \sigma & 1 \end{bmatrix}$, where the choices of $\sigma$ describe the intrinsic conditional dependency between agents’ option selections (we assume symmetric $\Sigma$ here). When we consider Fully-Cen mode and model a two-agent policy distribution with a multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$, we specify the closed form of Eq. (3) to adjust the joint-option selection distribution conditioned by on-going asynchronous options at option-level time $t$. In particular, we consider

$$\pi_\theta(o^t_1, o^t_2 | z^t) \triangleq \mathcal{N}(\mu, \Sigma), \quad (13)$$

if both agents choose new options. Otherwise, if agent 1 chooses a new option while agent 2 continues its on-going option, we have

$$\pi_\theta(o^t_1 | o^t_2, z^t) \triangleq \mathcal{N}(\mu, \Sigma), \quad (14)$$

where $\mu = \mu_1 + \sigma_{12}\sigma_{22}^{-1}(a - \mu_2)$ and $\Sigma = \sigma_{11} - \sigma_{12}\sigma_{22}^{-1}\sigma_{21}\sigma_{12}$.2

As shown in Fig. 11c, we experimented with a few different choices of $\sigma \in \{0, 0.5, 0.8\}$ and they all constantly improve the policy performance until convergence. This indicates that our approach works well with continuous option space. Moreover, we also observe that by setting a stronger dependency (larger value of $\sigma$) of option selection between agents, we achieve higher a training performance eventually. This is rather an interesting finding and it implies that if the option selections of both agents are largely correlated, they can cooperate better to accomplish the task. In this example, a strong correlation between the moving speeds of two agents can potentially help them adjust their relative speed in order to achieve simultaneous target arrival.

---

2 This follows the properties of multi-variate conditional Gaussian distribution.