Learning Hierarchical Teaching Policies for Cooperative Agents

Dong-Ki Kim
MIT-LIDS
MIT-IBM Watson AI Lab
dkkim93@mit.edu

Miao Liu
IBM Research
MIT-IBM Watson AI Lab
miao.liu1@ibm.com

Shayegan Omidshafiei*
MIT-LIDS
MIT-IBM Watson AI Lab
shayegan@mit.edu

Sebastian Lopez-Cot
MIT-LIDS
MIT-IBM Watson AI Lab
slcot@mit.edu

Matthew Riemer
IBM Research
MIT-IBM Watson AI Lab
mdriemer@us.ibm.com

Golnaz Habibi
MIT-LIDS
MIT-IBM Watson AI Lab
golnaz@mit.edu

Gerald Tesauro
IBM Research
MIT-IBM Watson AI Lab
gtesauro@us.ibm.com

Sami Mourad
IBM Research
MIT-IBM Watson AI Lab
Sami.Mourad@ibm.com

Murray Campbell
IBM Research
MIT-IBM Watson AI Lab
mcam@us.ibm.com

Jonathan P. How
MIT-LIDS
MIT-IBM Watson AI Lab
jhow@mit.edu

ABSTRACT

Collective learning can be greatly enhanced when agents effectively exchange knowledge with their peers. In particular, recent work studying agents that learn to teach other teammates has demonstrated that action advising accelerates team-wide learning. However, the prior work has simplified the learning of advising policies by using simple function approximations and only considering advising with primitive (low-level) actions, limiting the scalability of learning and teaching to complex domains. This paper introduces a novel learning-to-teach framework, called hierarchical multiagent teaching (HMAT), that improves scalability to complex environments by using the deep representation for student policies and by advising with more expressive extended action sequences over multiple levels of temporal abstraction. Our empirical evaluations demonstrate that HMAT improves team-wide learning progress in large, complex domains where previous approaches fail. HMAT also learns teaching policies that can effectively transfer knowledge to different teammates with knowledge of different tasks, even when the teammates have heterogeneous action spaces.

KEYWORDS

Learning agent-to-agent interactions; Deep reinforcement learning

1 INTRODUCTION

The history of human social groups provides evidence that the collective intelligence of multiagent populations may be greatly boosted if agents share their learned behaviors with others [29]. With this motivation in mind, we explore a new methodology for

---

*Work was completed while at MIT

allowing agents to effectively share their knowledge and learn from other peers while maximizing their collective team reward. Recently proposed frameworks allow for various types of knowledge transfer between agents [34]. In this paper, we focus on transfer based on *action advising*, where an experienced "teacher" agent helps a less experienced "student" agent by suggesting which action to take next. Action advising allows a student to learn effectively from a teacher by directly executing suggested actions without incurring much computation overhead.

Recent works on action advising include the Learning to Coordinate and Teach Reinforcement (LeCTR) framework [26], in which agents learn when and what actions to advise. While LeCTR improves upon prior advising methods based on heuristics [1, 4, 5, 36], it faces limitations in scaling to more complicated tasks with high-dimension state-action spaces, long time horizons, and delayed rewards. The key difficulty is *teacher credit assignment*: learning teacher policies requires estimates of the impact of each piece of advice on the student agentâĂŹs learning progress, but these estimates are difficult to obtain [26]. For example, if a student policy is represented by deep neural networks (DNN), the student requires mini-batches of experiences to stabilize its learning [12], and these experiences can be randomly selected from a replay memory [20, 23]. Hence, the student's learning progress is affected by a batch of advice suggested at varying times, and identifying the extent to which each piece of advice contributes to the student's learning is challenging (see Section 3.2 for details). As a result, many prior techniques use a simple function approximation, such as tile coding, to simplify the learning of teaching policies [26], but this approach does not scale well.

Contribution. This paper proposes a new learning-to-teach framework, called *hierarchical multiagent teaching (HMAT)*. The main contribution of this work is an algorithm that enables learning and teaching in problems with larger domains and higher complexity

than possible with previous advising approaches. Specifically, our approach: 1) handles large state-action spaces by using deep representation for student policies, 2) provides a method based on the extended advice sequences and the use of temporary policies (see Section 3.3 for details) that correctly estimates the teacher credit assignment and improves the teacher's learning process, and 3) addresses long-horizons and delayed rewards by advising temporally extended sequences of primitive actions (i.e., sub-goal) via hierarchical reinforcement learning (HRL) [24]. We note that the learning of high-level teaching policies in HMAT contrasts with prior work on action advising where teachers advise primitive (low-level) actions to students [1, 4, 5, 26, 36]. Empirical evaluations also demonstrate that agents in HMAT can learn high-level teaching policies that are transferable to different types of students and/or tasks, even when teammates have heterogeneous action spaces.

2 PRELIMINARIES

We consider a cooperative multiagent reinforcement learning (MA-RL) setting in which n agents jointly interact in an environment, then receive feedback via local observations and a shared team reward. This setting can be formalized as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), defined as a tuple $\{\mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \Omega, O, \mathcal{R}, \gamma\}$ [25]; $\mathcal{I} = \{1, \ldots, n\}$ is the set of agents, $\mathcal{S}$ is the set of states, $\mathcal{A} = \times_{i \in \mathcal{I}} \mathcal{A}^i$ is the set of joint actions, $\mathcal{T}$ is the transition probability function, $\Omega = \times_{i \in \mathcal{I}} \Omega^i$ is the set of joint observations, O is the observation probability function, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. At each timestep t, each agent i executes an action according to its policy $a_t^i \sim \pi^i(h_t^i; \theta^i)$ parameterized by θ^i, where $h_t^i = (o_1^i, \ldots, o_t^i)$ is the agent i's observation history at timestep t. A joint action $a_t = \{a_t^1, \ldots, a_t^n\}$ yields a transition from a current state $s_t \in \mathcal{S}$ to next state $s_{t+1} \in \mathcal{S}$ with probability $\mathcal{T}(s_{t+1} | s_t, a_t)$. Then, a joint observation $o_{t+1} = \{o_{t+1}^1, \ldots, o_{t+1}^n\}$ is obtained and the team receives a shared reward $r_t = \mathcal{R}(s_t, a_t)$. Each agent's objective is to maximize the expected cumulative team reward $\mathbb{E}[\sum_t \gamma^t r_t]$. Note that the policy parameter will often be omitted and reactive policies are considered (i.e., $\pi^i(h_t^i; \theta^i) \equiv \pi^i(o_t^i)$).

2.1 Learning to Teach in Cooperative MARL

In this section, we review key concepts and notations in the learning-to-teach framework (LeCTR) [26].

Task-level learning problem. LeCTR considers a cooperative MARL setting with two agents i and j in a shared environment. At each learning iteration, agents interact in the environment, collect experiences, and update their policies, π^i and π^j, with learning algorithms, $\mathbb{L}^i$ and $\mathbb{L}^j$. The resulting policies aim to coordinate and optimize final task performance. The problem of learning task-related policies is referred to as the task-level learning problem $\mathcal{P}_{\text{task}}$.

Advice-level learning problem. Throughout the task-level learning, agents may develop unique skills from their experiences. As such, it is potentially beneficial for agents to advise one another using their specialized knowledge to improve the final performance and accelerate the team-wide learning. The problem of learning teacher policies that decide *when* and *what* to advise is referred to as the advice-level learning problem $\widetilde{\mathcal{P}}_{\text{advice}}$, where $\widetilde{(\cdot)}$ denotes teacher property.
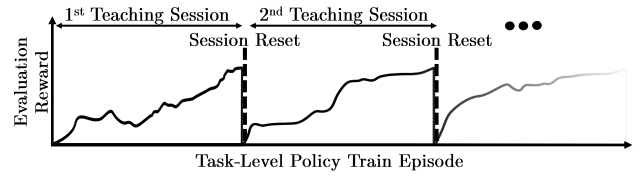

Figure 1: Illustration of task-level learning progress for each session. After completing a pre-determined episode count, task-level policies are re-initialized (teaching session reset). With each new teaching session, teacher policies are better at advising students, leading to faster learning progress for task-level policies (i.e., larger area under the learning curve).

Episode and session. Learning task-level and advice-level policies are both RL problems that are interleaved within the learning-to-teach framework, but there are important differences between $\mathcal{P}_{\text{task}}$ and $\widetilde{\mathcal{P}}_{\text{advice}}$. One difference lies in the definition of learning episodes because rewards about the success of advice are naturally delayed relative to typical task-level rewards. For $\mathcal{P}_{\text{task}}$, an episode terminates either when agents arrive at a terminal state or t exceeds the horizon T, but, for $\widetilde{\mathcal{P}}_{\text{advice}}$, an episode ends when task-level policies have converged, forming one "episode" for learning teacher policies. To distinguish these, we refer to an *episode* as one episode for $\mathcal{P}_{\text{task}}$ and a *session* as one episode for $\widetilde{\mathcal{P}}_{\text{advice}}$ (see Figure 1).

Learning complexity. Another major difference between $\mathcal{P}_{\text{task}}$ and $\widetilde{\mathcal{P}}_{\text{advice}}$ is in their learning objectives. $\mathcal{P}_{\text{task}}$ aims to maximize cumulative reward per episode, whereas $\widetilde{\mathcal{P}}_{\text{advice}}$ aims to maximize cumulative teacher reward per session, corresponding to accelerating the team-wide learning progress (i.e., a maximum area under the learning curve in a session). Consequently, the learning complexity involved in $\widetilde{\mathcal{P}}_{\text{advice}}$ is significantly higher than in $\mathcal{P}_{\text{task}}$ due to the much longer time horizons associated with advice-level policies than task-level policies. For example, task-level policies need to only consider actions during one episode (i.e., the horizon of T timesteps), but if the task-level policies converge after 500 episodes, then teachers need to learn how to advise during one session (i.e., the horizon of $500 \times T$ timesteps). In addition, teachers must consider the heterogeneous knowledge (compactly represented by task-level observations, actions, and Q-values; see Appendix A.3) between agents to advise, which inherently results in a higher dimension of the input size than for task-level policies and induces harder learning.

2.2 Hierarchical Reinforcement Learning

HRL is a structured framework with multi-level reasoning and extended temporal abstraction [2, 7, 8, 16, 27, 28, 33, 38]. HRL efficiently decomposes a complex problem into simpler sub-problems, which offers a benefit over non-HRL in solving difficult tasks with long horizons and delayed reward assignments. The closest HRL framework that we use in this paper is that of Nachum et al. [24] with a two-layer hierarchical structure: the higher-level manager policy $\pi_{\mathcal{M}}$ and the lower-level worker policy $\pi_{\mathcal{W}}$. The manager policy obtains an observation o_t and plans a high-level sub-goal $g_t \sim \pi_{\mathcal{M}}(o_t)$ for the worker policy. The worker policy attempts
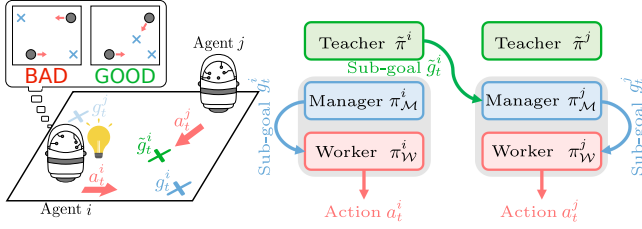
Figure 2: Agents teach each other according to an advising protocol (e.g., knowledgeable agent i evaluates the action that agent j intends to take and advises if needed).
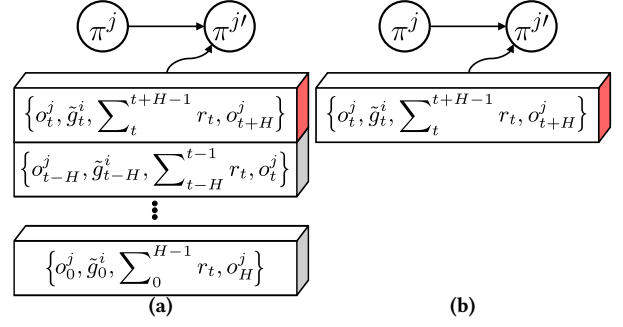

Figure 3: (a) Student j with the deep representation. In general, a mini-batch of experiences is required for the stable update. Note that the mini-batch includes the experiences affected by both current advice (denoted by color red) and previous advice (denoted by color gray). (b) Student j with the tile-coding representation. Tile-coding can be trained online and uses only the current experience.

to reach this sub-goal from the current state by executing a primitive action $a_t \sim \pi_W(o_t, g_t)$ in the environment. Following this framework, an updated sub-goal is generated by the manager every H timesteps and a sequence of primitive actions are executed by the worker. The manager learns to accomplish a task by optimizing the cumulative environment reward and stores an experience $\{o_t, g_t, \sum_t^{t+H-1} r_t, o_{t+H}\}$ every H timesteps. By contrast, the worker learns to reach the sub-goal by maximizing the cumulative intrinsic reward $r_t^{\text{intrinsic}}$ and stores an experience $\{o_t, a_t, r_t^{\text{intrinsic}}, o_{t+1}\}$ at each timestep. Without loss of generality, we also denote the next observation with the prime symbol o'.

3 HMAT OVERVIEW

HMAT improves the scalability of the learning and teaching in problems with larger domains and higher complexity by the deep representation of student policies and learning of high-level teacher policies to decide what high-level actions to advise fellow agents and when advice should be given. We begin by introducing our deep hierarchical task-level policy structure (Section 3.1). We then demonstrate why identifying which portions of the advice led to successful student learning is difficult to accomplish with deep task-level policies (Section 3.2). Finally, we explain how our algorithm addresses the teacher credit assignment issue in Section 3.3 and Section 4.

3.1 Deep Hierarchical Task-Level Policy

We extend task-level policies with DNN and hierarchical representations. Specifically, we replace π^i and π^j with deep hierarchical policies consisting of manager policies, π_M^i and π_M^j, and worker policies, π_W^i and π_W^j (see Figure 2). Note that the manager and worker policies in HMAT are trained with different objectives. Managers learn to accomplish a task together (i.e., solving $\mathcal{P}_{\text{task}}$) by optimizing cumulative reward, while workers are trained to reach sub-goals suggested by their managers. In this paper, we focus on transferring knowledge at the manager-level instead of the worker-level, since manager policies represent abstract knowledge, which is more relevant to fellow agents. Therefore, hereafter when we discuss task-level polices, we are implicitly only discussing the manager policies. The manager subscript $\mathcal{M}$ is often omitted when discussing these task-level policies to simplify notation (i.e., $\pi^i \equiv \pi_M^i$).

3.2 Teacher Credit Assignment Issue

Learning teacher policies requires estimates of the impact of the advice on the student agents' learning progress (i.e., teacher credit assignment). However, these estimates are difficult to obtain, especially when students' policies are represented by DNN [26].

To understand this challenge in more detail, consider an example, in which the teacher agent i decides to advise the student agent j with advice $\tilde{g}_t^i$. Agent i then requires feedback (reflecting the extent to which its advice contributed to the student's learning progress) to be able to improve its teaching strategy. Assuming j has a deep task-level policy, the student uses a mini-batch of experiences to stably update its task-level policy from π^j to $\pi^{j'}$ [12, 20, 23], and this mini-batch consists of many experiences, including the experiences affected by both current advice $\tilde{g}_t^i$ and previous advice (see Figure 3a). Consequently, each learning update of the student j is jointly affected by a batch of advice suggested at various times (i.e., $\{\tilde{g}_0^i, \ldots, \tilde{g}_{t-H}^i, \tilde{g}_t^i\}$), and identifying the amount of each piece of advice contributes to the student's learning is challenging.

Note that the tile-coding representation can be trained online [32] and the student agent uses only the experience affected by the teacher's current advice $\tilde{g}_t^i$ (see Figure 3b). As such, the impact of advice on task-level policies can be easily observed, and the teacher credit assignment problem is greatly simplified. That representation approach is, however, greatly limited in scalability to more complex environment, as our evaluations in Section 6.3 demonstrate.

3.3 HMAT Algorithm

HMAT iterates over the following three phases to learn how to coordinate with deep hierarchical task-level policies (i.e., solving $\mathcal{P}_{\text{task}}$) and how to provide advice using the teacher policies (i.e., solving $\widetilde{\mathcal{P}}_{\text{advice}}$). These phases are designed to address the teacher credit assignment issue with deep task-level policies (see Section 3.2). The issue is addressed by adapting ideas developed for learning an exploration policy for a single agent [40], which includes an extended view of actions and the use of a temporary policy for measuring a reward for the exploration policy. We adapt and extend these

Table 1: Summary of considered teacher reward functions. $\hat{R}$ denotes the sum of rewards in the self-practice experiences.

Teacher Reward Function Name	Description	Teacher Reward
Value Estimation Gain (VEG)	Student's Q-value above a threshold τ [26]	$\mathbb{1}(Q_{\text{student}} > \tau)$
Difference Rollout (DR)	Difference in rollout reward before/after advising [40]	$\hat{R} - \hat{R}_{\text{before}}$
Current Rollout (CR)	Rollout reward after advising phase	$\hat{R}$

ideas from learning-to-explore in a single-agent setting into our learning-to-teach in a multiagent setting. The pseudocode of HMAT is presented in Algorithm 1.

Phase I: Advising. Agents advise each other using their teacher policies according to the advising protocol (see Section 4.1) during one episode. This process generates a batch of task-level experiences influenced by the teaching policy's behavior. Specifically, we *extend* the concept of teacher policies by providing advice in the form of a sequence of multiple sub-goals $\widetilde{g}_{0:T} = \{\widetilde{g}_{0:T}^i, \widetilde{g}_{0:T}^j\}$, where $\widetilde{g}_{0:T}^i = \{\widetilde{g}_0^i, \widetilde{g}_H^i, \widetilde{g}_{2H}^i, \ldots, \widetilde{g}_T^i\}$ denotes multiple advice by agent i in one episode, instead of just providing one piece of advice $\widetilde{g}_t = \{\widetilde{g}_t^i, \widetilde{g}_t^j\}$, before updating task-level policies. One teacher action in this extended view corresponds to providing multiple pieces of advice during one episode, which contrasts with previous teaching approaches that the task-level policies were updated based on a single piece of advice [1, 4, 5, 26, 36]. This extension is important: by following advice $\widetilde{g}_{0:T}$ in an episode, a batch of task-level experiences for agents i and j, $E_{0:T}^{\text{advice}} = \{\{o_{0:T}^i, \widetilde{g}_{0:T}^i, r_{0:T}, o_{0:T}^{i\prime}\}, \{o_{0:T}^j, \widetilde{g}_{0:T}^j, r_{0:T}, o_{0:T}^{j\prime}\}\}$, are generated to allow the stable mini-batch updates of deep policies.

Phase II: Advice evaluation. Learning teacher policies requires reward feedback from the advice suggested in Phase I. Phase II evaluates and estimates the impact of the advice on improving team-wide learning progress, yielding the teacher policies' rewards. We use *temporary* task-level policies to estimate the teacher reward for $\widetilde{g}_{0:T}$, where the temporary policies denote the copy of their current task-level policies (i.e., $\pi_{\text{temp}} \leftarrow \pi$) and $\pi = \{\pi^i, \pi^j\}$. To determine the teacher reward for $\widetilde{g}_{0:T}$, π_{temp} are updated for a small number of iterations using $E_{0:T}^{\text{advice}}$ (i.e., $\pi_{\text{temp}}' \leftarrow \mathbb{L}(\pi_{\text{temp}}, E_{0:T}^{\text{advice}})$), where $\mathbb{L} = \{\mathbb{L}^i, \mathbb{L}^j\}$. Then, the updated temporary policies generate a batch of self-practice experiences $E_{0:\overline{T}}^{\text{self-prac.}}$ by rolling out a total of $\overline{T}$ timesteps without involving teacher policies. These self-practice experiences based on π_{temp}' reflect how agents on their own would perform after the advising phase and can be used to estimate the impact of $\widetilde{g}_{0:T}$ on team-wide learning. The teacher reward function $\widetilde{\mathcal{R}}$ (see Section 4.1) uses the self-practice experiences to compute the teacher reward for $\widetilde{g}_{0:T}$ (i.e., $\widetilde{r} = \widetilde{\mathcal{R}}(E_{0:\overline{T}}^{\text{self-prac.}})$). The key point is that π_{temp}', which are used to compute $\widetilde{r}$, are updated based on $E_{0:T}^{\text{advice}}$ only, so experiences from the past iterations do not affect the teacher reward estimation for $\widetilde{g}_{0:T}$. Consequently, the use of temporary policies addresses the teacher credit assignment issue.

Phase III: Policy update. Task-level policies are updated to solve $\mathcal{P}_{\text{task}}$ by using $\mathbb{L}^i$ and $\mathbb{L}^j$, and advice-level policies are updated to solve $\widetilde{\mathcal{P}}_{\text{advice}}$ by using teacher learning algorithms $\widetilde{\mathbb{L}}^i$ and $\widetilde{\mathbb{L}}^j$. In particular, task-level policies, π^i and π^j, are updated for the next

iteration by randomly sampling experiences from task-level experience memories, $\mathcal{D}^i$ and $\mathcal{D}^j$. As in Xu et al. [40], both $E_{0:T}^{\text{advice}}$ and $E_{0:\overline{T}}^{\text{self-prac.}}$ are added to the task-level memories. Similarly, after adding the teacher experience, which is collected from the advising (Phase I) and advice evaluation phases (Phase II), to teacher experience memories, $\widetilde{\mathcal{D}}^i$ and $\widetilde{\mathcal{D}}^j$, teacher policies are updated by randomly selecting samples from their replay memories.

4 HMAT DETAILS

We explain important details of the teacher policy (focusing on teacher i for clarity). See appendix for the remaining details.

4.1 Advice-Level Policy Details

Teacher observation and action. Teacher-level observations $\widetilde{o}_t = \{\widetilde{o}_t^i, \widetilde{o}_t^j\}$ compactly provide information about the nature of the heterogeneous knowledge between the two agents. Given $\widetilde{o}_t^i$, teacher i decides when and what to advise, with one action for deciding whether or not it should provide advice and another action for selecting the sub-goal to give as advice. If no advice is provided, student j executes its originally intended sub-goal.

Teacher reward function. Teacher policies aim to maximize their cumulative teacher rewards in a session that should result in faster task-level learning progress. Recall in Phase II that the batch of self-practice experiences reflect how agents by themselves perform after one advising phase. Then, the question is what is an appropriate teacher reward function $\widetilde{\mathcal{R}}$ that can map self-practice experiences into better task-level learning performance. Intuitively, maximizing the teacher rewards $\widetilde{r}$ returned by an appropriate teacher reward function means that teachers should advise so that the task-level learning performance is maximized after one advising phase. In this work, we consider a new reward function, called current rollout (CR), which returns the sum of rewards in the self-practice experiences $E_{0:\overline{T}}^{\text{self-prac.}}$. We also evaluate different choices of teacher reward functions, including the ones in Omidshafiei et al. [26] and Xu et al. [40], as described in Table 1.

Teacher experience. One teacher experience corresponds to $\widetilde{E}_{0:T}^i = \{\widetilde{o}_{0:T}^i, \widetilde{g}_{0:T}^i, \widetilde{r}, \widetilde{o}_{0:T}^{i\prime}\}$; where $\widetilde{o}_{0:T}^i = \{\widetilde{o}_0^i, \widetilde{o}_H^i, \widetilde{o}_{2H}^i, \ldots, \widetilde{o}_T^i\}$ is the teacher observation; $\widetilde{g}_{0:T}^i = \{\widetilde{g}_0^i, \widetilde{g}_H^i, \widetilde{g}_{2H}^i, \ldots, \widetilde{g}_T^i\}$ is the teacher action; $\widetilde{r}$ is the estimated teacher reward with $\widetilde{\mathcal{R}}$; and $\widetilde{o}_{0:T}^{i\prime} = \{\widetilde{o}_0^{i\prime}, \widetilde{o}_H^{i\prime}, \widetilde{o}_{2H}^{i\prime}, \ldots, \widetilde{o}_T^{i\prime}\}$ is the next teacher observation, obtained by updating $\widetilde{o}_{0:T}^i$ with the updated temporary policy $\pi_{\text{temp}}^{j\prime}$ (i.e., representing the change in student j's knowledge due to advice $\widetilde{g}_{0:T}^i$).

Advice protocol. Consider Figure 2, where there are two roles: that of a student agent j (i.e., an agent whose manager policy receives advice) and that of a teacher agent i (i.e., an agent whose

teacher policy gives advice). Note that agents i and j can simultaneously teach each other, but, for clarity, Figure 2 only shows a one-way interaction. Here, student j has decided that it is appropriate to strive for a sub-goal g_t^j by querying its manager policy. Before j passes the sub-goal g_t^j to its worker, i's teaching policy checks j's intended sub-goal and decides whether to advise or not. Having decided to advise, i transforms its task-level knowledge into desirable sub-goal advice $\widetilde{g}_t^i$ via its teacher policy and suggests it to j. After j accepts the advice from the teacher, the updated sub-goal g_t^j is passed to j's worker policy, which then generates a primitive action a_t^j.

4.2 Training Protocol

Task-level training. We use TD3 [11] to train the worker and manager task-level policies. TD3 is an actor-critic algorithm which introduces two critics, Q_1 and Q_2, to reduce overestimation of Q-value estimates in DDPG [20] and yields more robust learning performance. Originally, TD3 is a single-agent deep RL algorithm accommodating continuous spaces/actions. Here, we extend TD3 to multiagent settings with a resulting algorithm termed MATD3, and non-stationarity in MARL is addressed by applying centralized critics/decentralized actors [10, 21]. Another algorithm termed HMATD3 further extends MATD3 with HRL. In HMATD3, agent i's task policy critics, Q_1^i and Q_2^i, minimize the following critic loss:

$$\mathcal{L} = \sum_{\alpha=1}^{2} \mathbb{E}_{\{o,g,r,o'\} \sim \mathcal{D}^i} \left[y - Q_\alpha^i(o,g) \right]^2,$$
$$\text{s.t.} \quad y = r + \gamma \min_{\beta=1,2} Q_{\beta,\text{target}}^i(o', \pi_{\text{target}}(o') + \epsilon), \tag{1}$$

where $o = \{o^i, o^j\}$, $g = \{g^i, g^j\}$, $o' = \{o^{i\prime}, o^{j\prime}\}$, $\pi_{\text{target}} = \{\pi_{\text{target}}^i, \pi_{\text{target}}^j\}$, the subscript "target" denotes the target network, and $\epsilon \sim \mathcal{N}(0,\sigma)$. The agent i's actor policy π^i with parameter θ^i is updated by:

$$\nabla_{\theta^i} J(\theta^i) = \mathbb{E}_{o \sim \mathcal{D}^i} \left[\nabla_{\theta^i} \pi^i(g^i|o^i) \nabla_{g^i} Q_1^i(o,g)|_{g=\pi(o)} \right]. \tag{2}$$

Advice-level training. TD3 is also used for updating teacher policies. We modify Eqn (1) and (2) to account for the teacher's *extended* view. Considering agent i for clarity, agent i's teacher policy critics, $\widetilde{Q}_1^i$ and $\widetilde{Q}_2^i$, minimize the following critic loss:

$$\widetilde{\mathcal{L}} = \sum_{\alpha=1}^{2} \mathbb{E}_{\widetilde{E}_{0:T} \sim \widetilde{\mathcal{D}}^i} \left[\mathbb{E}_{\widetilde{E} \sim \widetilde{E}_{0:T}} \left(y - \widetilde{Q}_\alpha^i(\widetilde{o}, \widetilde{g}) \right) \right]^2,$$
$$\text{s.t.} \quad y = \widetilde{r} + \gamma \min_{\beta=1,2} \widetilde{Q}_{\beta,\text{target}}^i(\widetilde{o}', \widetilde{\pi}_{\text{target}}(\widetilde{o}') + \epsilon), \tag{3}$$

$$\widetilde{E}_{0:T} = \{\widetilde{o}_{0:T}, \widetilde{g}_{0:T}, \widetilde{r}, \widetilde{o}_{0:T}'\}, \widetilde{E} = \{\widetilde{o}, \widetilde{g}, \widetilde{r}, \widetilde{o}'\},$$

where $\widetilde{o} = \{\widetilde{o}^i, \widetilde{o}^j\}$; $\widetilde{g} = \{\widetilde{g}^i, \widetilde{g}^j\}$; $\widetilde{o}' = \{\widetilde{o}^{i\prime}, \widetilde{o}^{j\prime}\}$; $\widetilde{\pi}_{\text{target}} = \{\widetilde{\pi}_{\text{target}}^i, \widetilde{\pi}_{\text{target}}^j\}$. The agent i's actor policy $\widetilde{\pi}^i$ with parameter $\widetilde{\theta}^i$ is updated by:

$$\nabla_{\widetilde{\theta}^i} J(\widetilde{\theta}^i) = \mathbb{E}_{\widetilde{o}_{0:T} \sim \widetilde{\mathcal{D}}^i} \left[\mathbb{E}_{\widetilde{o} \sim \widetilde{o}_{0:T}} (z) \right], \tag{4}$$

where $z = \nabla_{\widetilde{\theta}^i} \widetilde{\pi}^i(\widetilde{g}^i|\widetilde{o}^i) \nabla_{\widetilde{g}^i} \widetilde{Q}_1^i(\widetilde{o}, \widetilde{g})|_{\widetilde{g}=\widetilde{\pi}(\widetilde{o})}$.

5 RELATED WORK

Action advising is not the only possible approach to transfer knowledge. Works on imitation learning studies on how to learn a policy

---

Algorithm 1 HMAT Pseudocode

Require: Maximum number of episodes in session S
Require: Teacher update frequency f_{teacher}
1: Initialize advice-level policies $\widetilde{\pi}$ and memories $\widetilde{D}$
2: **for** teaching session **do**
3:     Re-initialize task-level policy parameters π
4:     Re-initialize task-level memories D
5:     Re-initialize train episode count: $e = 0$
6:     **while** $e \leq S$ **do**
7: Phase $\Big\{$ $E_{0:T}^{\text{advice}}, \{\widetilde{o}_{0:T}, \widetilde{g}_{0:T}\} \leftarrow$ Teacher's advice
8:   I $\Big\{$ Update episode count: $e \leftarrow e + 1$
9:     $\Big($ Copy temporary task-level policies: $\pi_{\text{temp}} \leftarrow \pi$
10:    $\Big|$ Update to π_{temp}' using Eqn (1)–(2) with $E_{0:T}^{\text{advice}}$
11: Phase $\Big\{$ $E_{0:\overline{T}}^{\text{self-prac.}} \leftarrow \pi_{\text{temp}}'$ perform self-practice
12:   II $\Big|$ Update episode count: $e \leftarrow e + \overline{T}/T$
13:    $\Big($ $\widetilde{r} \leftarrow$ Get teacher reward with $\widetilde{\mathcal{R}}$
14:    $\Big($ Add $E_{0:T}^{\text{advice}}$ and $E_{0:\overline{T}}^{\text{self-prac.}}$ to D
15:    $\Big|$ Add a teacher experience $\widetilde{E}_{0:T}$ to $\widetilde{D}$
16: Phase $\Big|$ Update π using Eqn (1)–(2) with D
17:   III $\Big|$ **if** e mod f_{teacher} $== 0$ **then**
18:    $\Big|$ Update $\widetilde{\pi}$ using Eqn (3)–(4) with $\widetilde{D}$
19:    $\Big($ **end if**
20:    **end while**
21: **end for**

---

from expert demonstrations [6, 30, 31]. Recent work applied imitation learning for multiagent coordination [18] and explored effective combinations of imitation learning and HRL [17]. Curriculum learning [3, 13, 37], which progressively increases task difficulty, is also relevant. Approaches in curriculum learning measure or learn the hardness of tasks and design a curriculum for a learning agent to follow. Curriculum learning has many applications, including recent works that learn a training data curriculum for an image classifier [9, 15]. Another work of Thomaz and Breazeal [35] studies transferring human knowledge to an RL agent via providing the reward signals. While most works on these topics focus on learning and/or transferring knowledge for solving single-agent problems, this paper investigates peer-to-peer knowledge transfer in cooperative MARL. The related work by Xu et al. [40] learns an exploration policy, which relates to our approach of learning teaching policies. However, their approach led to unstable learning in our setting, motivating our policy update rule in Eqn (3) and (4). We also consider various teacher reward functions, including the one in Xu et al. [40], and a new reward function of CR that is empirically shown to perform better in our domains.

6 EVALUATION

We demonstrate HMAT's performance in increasingly challenging domains that involve continuous states/actions, long horizons, and delayed rewards. See appendix for additional experiment details.
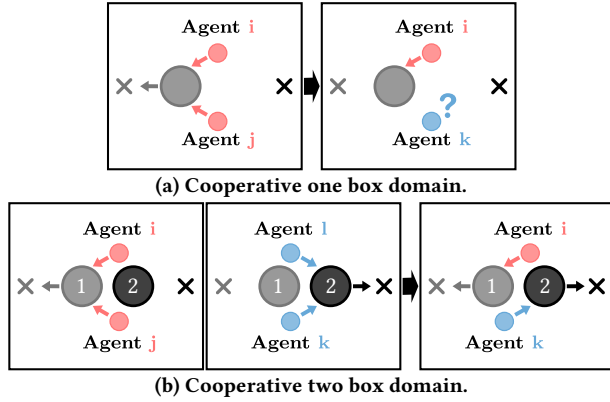
(a) Cooperative one box domain.

(b) Cooperative two box domain.

Figure 4: Two scenarios used for evaluating HMAT.

6.1 Experiment Setup

Our domains are based on OpenAI's multiagent particle environment. We modify the environment and propose new domains, called cooperative one and two box push:

Cooperative one box push (COBP). The domain consists of one round box and two agents (see Figure 4a). The objective is to move the box to the target on the left side as soon as possible. The box can be moved iff two agents act on it together. This unique property requires agents to coordinate. The domain also has a delayed reward because there is no change in reward until the box is moved by the two agents.

Cooperative two box push (CTBP). This domain is similar to COBP but with increased complexity. There are two round boxes in the domain (see Figure 4b). The objective is to move the left box (box1) to the left target (target1) and the right box (box2) to the right target (target2). In addition, the boxes have different mass – box2 is 3x heavier than box1.

Heterogeneous knowledge. For each domain, we provide each agent with a different set of priors to ensure heterogeneous knowledge between them and motivate interesting teaching scenarios. For COBP, agents i and j are first trained to move the box to the left target. Then, agents i and k are teamed up, where k has no knowledge about the domain. Agent i, which understands how to move the box to the left, should teach agent k by giving good advice to improve k's learning progress. For CTBP, agents i and j have received prior training about how to move box1 to target1, and agents k and l understand how to move box2 to target2. However, these two teams have different skills as the tasks involve moving boxes with different weights (light vs heavy) and also in different directions (left vs right). Then agents i and k are teamed up, and in this scenario, agent i should transfer its knowledge about moving box1 to k. Meanwhile, agent k should teach i how to move box2, so that there is a two-way transfer of knowledge where each agent is the primary teacher at one point and primary student at another.

Domain complexity. COBP and CTBP are comparable to other recent MARL scenarios [19, 39] but are more difficult because of the inherent delayed rewards associated with the fact that the box moves iff both agents learn to push it together. These domains are also considerably more challenging than those investigated

in the prior work on learning to teach [26], which considered domains (repeated game and gridworld) with short horizons and low-dimension, discrete space-action spaces.

6.2 Baselines

We compare several baselines in order to provide context for the performance of HMAT.

No-teaching. MATD3 and HMATD3 (Section 4.2) are baselines for a primitive and hierarchical MARL without teaching, respectively.

Learn to teach. The original LeCTR framework with tile-coding task-level policies (LeCTR–Tile) is compared. We also consider modifications of that framework, in which task-level policies are learned with deep RL, MATD3 (LeCTR–D) and HMATD3 (LeCTR–HD). Lastly, we additionally compare two LeCTR-based baselines using online MATD3 (LeCTR–OD) and online HMATD3 (LeCTR–OHD), where the online update denotes the task-level policy update with the most recent experience only (i.e., no mini-batch).

Heuristic teaching. Two heuristic-based primitive teaching baselines, Ask Important (AI) and Ask Important–Correct Important (AICI) [1], are compared. In AI, each student asks for advice based on the importance of a state using its Q-values. When asked, the teacher agent always advises with its best action at a student state. Students in AICI also ask for advice, but teachers can decide whether to advise with their best action or not to advise. Each teacher decides based on the state importance using its Q-values and the difference between the student's intended action and teacher's intended action at a student state. Note that AICI is one of the best performing heuristic algorithms in Amir et al. [1]. Hierarchical AI (HAI) and hierarchical AICI (HAICI) are similar to AI and AICI, but teach in the hierarchical setting (i.e., managers teach each other).

HMAT variant. A non-hierarchical variant of HMAT, called MAT, is compared.

6.3 Results on One Box and Two Box Push

Table 2 compares HMAT and its baselines. The results show both final task-level performance ($\bar{V}$) (i.e., final episodic average reward measured at the end of the session) and area under the task-level learning curve (AUC) – higher values are better for both metrics.

Comparisons to no-teaching. The results demonstrate improved task-level learning performance with HMAT compared to HMATD3 and with MAT compared to MATD3, as indicated by the higher final performance ($\bar{V}$) and the larger rate of learning (AUC) in Figures 5a and 5b. These results demonstrate the main benefit of teaching that it accelerates the task-level learning progress.

Comparisons to learning to teach. HMAT also achieves better performance than the LeCTR baselines. LeCTR–Tile shows the smallest $\bar{V}$ and AUC due to the limitations of the tile-coding representation of the policies in these complex domains. Teacher policies in LeCTR–D and LeCTR–HD have poor estimates of the teacher credit assignment with deep task-level policies, which result in unstable learning of the advising policies and worse performance than the no-teaching baselines (LeCTR–D vs MATD3, LeCTR–HD vs HMATD3). In contrast, both LeCTR–OD and LeCTR–OHD have good estimates of the teacher credit assignment as the task-level policies are updated online. However, these two approaches suffer

Table 2: $\bar{V}$ and AUC for different algorithms. Results show a mean and standard deviation (std) computed for 10 sessions. Best results in bold (computed via a t-test with $p < 0.05$).

Algorithm	Hierarchical?	Teaching?	One Box Push		Two Box Push	
			$\bar{V}$	AUC	$\bar{V}$	AUC
MATD3	✗	✗	-12.64 ± 3.42	148 ± 105	-44.21 ± 5.01	1833 ± 400
LeCTR–Tile	✗	✓	-18.00 ± 0.01	6 ± 0	-60.50 ± 0.05	246 ± 3
LeCTR–D	✗	✓	-14.85 ± 2.67	81 ± 80	-46.07 ± 3.00	1739 ± 223
LeCTR–OD	✗	✓	-17.92 ± 0.10	17 ± 3	-60.43 ± 0.08	272 ± 26
AI	✗	✓	$\mathbf{-11.33 \pm 2.46}$	162 ± 86	-41.09 ± 3.50	2296 ± 393
AICI	✗	✓	$\mathbf{-10.60 \pm 0.85}$	200 ± 74	-38.23 ± 0.45	2742 ± 317
MAT (with CR)	✗	✓	$\mathbf{-10.04 \pm 0.38}$	274 ± 38	-39.49 ± 2.93	2608 ± 256
HMATD3	✓	✗	-10.24 ± 0.20	427 ± 18	-31.55 ± 3.51	4288 ± 335
LeCTR–HD	✓	✓	-12.10 ± 0.94	265 ± 59	-38.87 ± 2.94	2820 ± 404
LeCTR–OHD	✓	✓	-16.77 ± 0.66	70 ± 22	-60.23 ± 0.33	306 ± 28
HAI	✓	✓	$\mathbf{-10.23 \pm 0.19}$	427 ± 9	-31.37 ± 3.71	4400 ± 444
HAICI	✓	✓	$\mathbf{-10.25 \pm 0.26}$	433 ± 5	-29.32 ± 1.19	4691 ± 261
HMAT (with VEG)	✓	✓	-10.38 ± 0.25	424 ± 28	-29.73 ± 2.89	4694 ± 366
HMAT (with DR)	✓	✓	-10.36 ± 0.32	417 ± 31	$\mathbf{-28.12 \pm 1.58}$	4758 ± 286
HMAT (with CR)	✓	✓	$\mathbf{-10.10 \pm 0.19}$	$\mathbf{458 \pm 8}$	$\mathbf{-27.49 \pm 0.96}$	$\mathbf{5032 \pm 186}$
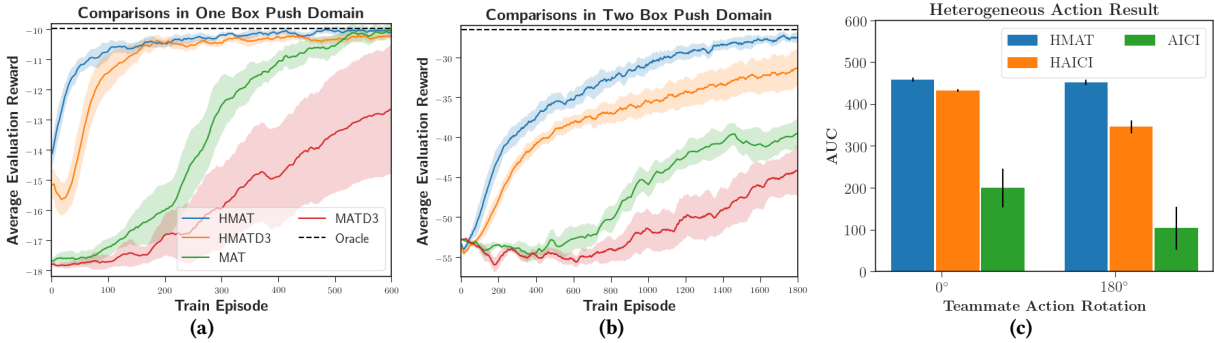
Figure 5: (a) and (b) Task-level learning progress in the one box push domain and two box push domain, respectively. The oracles in (a) and (b) refer to performance of converged HMATD3. For fair comparisons, HMAT and MAT include both the number of episodes used in Phase I and II when counting the number of train episodes. (c) Heterogeneous action AUC based on the action rotation. Mean and 95% confidence interval computed for 10 sessions are shown in all figures.

from the instability caused by the absence of a mini-batch update for the DNN policies.

Other comparisons. HMAT attains the best performance in terms of $\bar{V}$ and AUC compared to the heuristics-based baselines. Furthermore, HMAT also shows better performance than MAT, demonstrating the benefit of the high-level advising that helps address the delayed rewards and long-time horizons in these two domains. Lastly, our empirical experiments show that CR performs the best with HMAT. Consistent with this observation, the learning progress estimated with CR has a high correlation with the true learning progress (see Figure 6). These combined results demonstrate the key advantage of HMAT in that it accelerates the learning progress for complex tasks with continuous states/actions, long horizons, and delayed rewards.

6.4 Transferability and Heterogeneity

We evaluate the transferability of HMAT in advising different types of students and/or tasks and teaching with heterogeneous action spaces. The numerical results are mean/standard deviation for 10 sessions and a t-test with $p < 0.05$ is performed for checking statistical significance.

Transfer across different student types. We first create a small population of students, each having different knowledge. Specifically, we create 7 students that can push one box to distinct areas in the one-box push domain: top-left, top, top-right, right, bottom-right, bottom, and bottom-left. This population is divided into train {top-left, top, bottom-right, bottom, and bottom-left}, validation {top-right}, and test {right} groups. After the teacher policy has converged, we fix the policy and transfer it to a different setting in which the teacher advises a student in the test group. Although the teacher has never interacted with the student in the test group
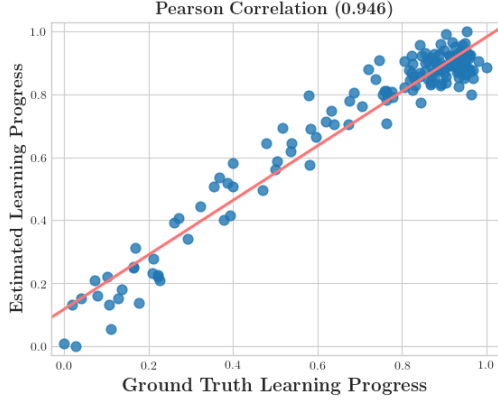
Figure 6: Ground-truth vs estimated learning progress with CR. The CR teacher reward function estimates the true learning progress with the high Pearson correlation of 0.946.

Figure 7: Teacher actor loss between the different numbers of threads. With an increasing number of threads, a teacher policy converges faster.

before, it achieves an AUC of 400 ± 10, compared to no-teaching baseline (HMATD3) AUC of 369 ± 27.

Transfer across different tasks. We first train the teacher that learns to transfer knowledge to agent k about how to move the box to the left in the one box push domain. Then we fix the converged teacher policy and evaluate on a different task of moving the box to the right. While task-level learning without teaching achieves AUC of 363 ± 43, task-level learning with teaching achieves AUC of 414 ± 11. Thus, learning is faster, even when using pre-trained teacher policies from different tasks.

Teaching with heterogeneous action spaces. We consider heterogeneous action space variants in COBP, where agent k has both remapped manager and worker action spaces (e.g., 180° rotation) compared to its teammate i. Note that teachers in the heuristic-based approaches advise with their best action assuming the action space homogeneity between the teachers and students. Consequently, when the action space is flipped, the heuristic advising can be an inefficient teaching strategy and lead teammates to no-reward regions. As Figure 5c shows, both HAICI and AICI show decreased AUC when the action space is remapped. In contrast, the capability of learning to advise in HMAT can understand the heterogeneity in the action space, so HMAT achieves the best performance regardless of action rotation.

6.5 HMAT Analyses

Teacher reward accuracy. Developing ground-truth learning progress of task-level policies often requires an expert policy and could be computationally undesirable [13]. Thus, HMAT uses an estimation of the learning progress as a teacher reward. However, it is important to understand how close the estimation is to true learning progress. The goal of teacher policies is to maximize the cumulative teacher reward, so a wrong estimate of teaching reward would result in learning undesirable teacher behaviors. In this section, we aim to measure the differences between the true and estimated learning progress and analyze the CR teacher reward function, which performed the best.
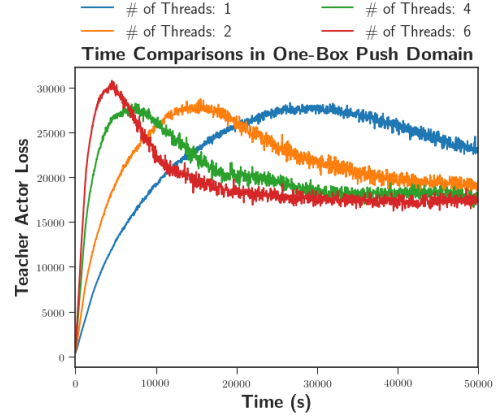
In imitation learning, with an assumption of a given expert, one standard method to measure the true learning progress is by measuring the distance between an action of a learning agent and an optimal action of an expert [6, 30]. Similarly, we pre-train expert policies using HMATD3 and measure the true learning progress by the action differences. The comparison between the true and estimated learning progress using the CR teacher reward function is shown in Figure 6. The Pearson correlation is 0.946, which empirically shows that CR well estimates the learning progress.

Asynchronous HMAT. Similar to a deep RL algorithm requiring millions of episodes to learn a useful policy [23], our teacher policies would require many *sessions* to learn. As one session consists of many episodes, much time might be needed until teacher policies converge. We address this potential issue with an asynchronous policy update with multi-threading as in asynchronous advantage actor-critic (A3C) [22]. A3C demonstrated a reduction in training time that is roughly linear in the number of threads. We also show that our HMAT variant, asynchronous HMAT, achieves a roughly linear reduction in training time as a function of the number of threads (see Figure 7).

7 CONCLUSION

The paper presents HMAT, which utilizes the deep representation and HRL to transfer knowledge between agents in cooperative MARL. We propose a method to overcome the teacher credit assignment issue and show accelerated learning progress in challenging domains. Ongoing work is expanding to more than two agents.

A APPENDIX

A.1 Implementation Details

Each policy's actor and critic are two-layer feed-forward neural networks consisting of the rectified linear unit (ReLU) activations. In terms of the task-level actor policies, a final layer of the tanh activation, which has the range of (-1, 1), is used at the output. The actor policies for both the worker and primitive task-level policies

output two actions that correspond to x–y forces to move in the one box and two box domains. Similarly, the actor policies for the manager task-level policies output two actions, but they correspond to sub-goals of (x, y) coordinate. Regarding the advice-level actor policies, they output four actions, where the first two outputs correspond to *what* to advise (i.e., continuous sub-goal advice of (x, y) coordinate) and the last two outputs correspond to *when* to advise (i.e., discrete actions converted to the one-hot encoding). Two separate final layers of the tanh activation and the linear activation are applied to what and when to advise actions, respectively. The Gumbel-Softmax estimator [14] is used to compute gradients for the discrete actions of when to advise. Regarding the hierarchical methods, since we focus teaching at the manager-level, not at the worker-level, we pre-train the worker policies by giving randomly generated sub-goals and then fix the policies. The intrinsic reward function to pre-train the worker policy is the negative distance between the current position and sub-goal. All hierarchical methods presented in this work use the pre-trained workers.

A.2 COBP/CTBP Experiment Details

Each agent's observation includes its position/speed, the positions of the box(es), targets, and its peer. In terms of initialization, the left and right box initialize at $(−0.25, 0.0)$ and $(0.25, 0.0)$, respectively, the two targets initialize at $(−0.85, 0.0)$ and $(0.85, 0.0)$, respectively, and agents reset at random locations. Regarding the maximum timestep T, COBP and CTBP have 50 and 100 timesteps, respectively. The maximum number of episodes in a session S is 600 episodes for COBP and 1800 episodes for CTBP. Also, for the self-practice rollout size $\overline{T}$, COBP and CTBP use 100 and 200 timesteps, respectively. Managers in all hierarchical methods generate the sub-goal every $H = 5$ timesteps. Lastly, we use Adam optimizer with the actor learning rate of 0.0001 and the critic learning rate of 0.001.

A.3 Teacher Observation

For agent i's teacher policy, its observation $\widetilde{o}_t^i$ consists of:

$$\widetilde{o}_t^i = \big\{ \underbrace{o_t, g_t^i, g_t^{ij}, Q^i(o_t, g_t^i, g_t^j), Q^i(o_t, g_t^i, g_t^{ij})}_{\text{Teacher Knowledge}},$$

$$\underbrace{g_t^j, Q^j(o_t, g_t^i, g_t^j), Q^j(o_t, g_t^i, g_t^{ij})}_{\text{Student Knowledge}},$$

$$\underbrace{R_{\text{phase I}}, R_{\text{phase II}}, t_{\text{remain}}}_{\text{Misc.}} \big\},$$

where $o_t = \{o_t^i, o_t^j\}$, $g_t^i \sim \pi^i(o_t^i)$, $g_t^j \sim \pi^j(o_t^j)$, $g_t^{ij} \sim \pi^i(o_t^j)$, Q^i and Q^j are the centralized critics for agents i and j, respectively, $R_{\text{phase I}}$ and $R_{\text{phase II}}$ are average rewards in the last few iterations in Phase I and II, respectively, and t_{remain} is the remaining time in the session.

REFERENCES

[1] Ofra Amir, Ece Kamar, Andrey Kolobov, and Barbara J Grosz. 2016. Interactive teaching strategies for agent training. In *International Joint Conferences on Artificial Intelligence (IJCAI)*.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The Option-Critic Architecture. In *Association for the Advancement of Artificial Intelligence (AAAI)*. 1726–1734. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14858

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *International Conference on Machine Learning (ICML)*. 41–48.

[4] Jeffery Allen Clouse. 1996. On integrating apprentice learning and reinforcement learning. (1996).

[5] Felipe Leno da Silva, Ruben Glatt, and Anna Helena Reali Costa. 2017. Simultaneously Learning and Advising in Multiagent Reinforcement Learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1100–1108.

[6] Mayank Daswani, Peter Sunehag, and Marcus Hutter. 2015. Reinforcement learning with value advice. In *Asian Conference on Machine Learning (ACML)*, Vol. 39. 299–314. http://proceedings.mlr.press/v39/daswani14.html

[7] Peter Dayan and Geoffrey E Hinton. 1993. Feudal reinforcement learning. In *Neural Information Processing Systems (NIPS)*.

[8] Thomas G. Dietterich. 2000. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *J. Artif. Int. Res.* 13, 1 (Nov. 2000), 227–303. http://dl.acm.org/citation.cfm?id=1622262.1622268

[9] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2018. Learning to Teach. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=HJewuJWCZ

[10] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual Multi-Agent Policy Gradients. *CoRR* abs/1705.08926 (2017). arXiv:1705.08926 http://arxiv.org/abs/1705.08926

[11] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning (ICML)*.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[13] Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. 2017. Automated Curriculum Learning for Neural Networks. *CoRR* abs/1704.03003 (2017). arXiv:1704.03003 http://arxiv.org/abs/1704.03003

[14] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[15] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *International Conference on Machine Learning (ICML)*, Vol. 80. 2304–2313.

[16] Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. 2016. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In *Neural Information Processing Systems (NIPS)*. 3682–3690. http://dl.acm.org/citation.cfm?id=3157382.3157509

[17] Hoang Minh Le, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé III. 2018. Hierarchical Imitation and Reinforcement Learning. *CoRR* abs/1803.00590 (2018). arXiv:1803.00590 http://arxiv.org/abs/1803.00590

[18] Hoang M Le, Yisong Yue, Peter Carr, and Patrick Lucey. 2017. Coordinated Multi-Agent Imitation Learning. In *International Conference on Machine Learning (ICML)*. 1995–2003.

[19] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. 2019. Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient. *Association for the Advancement of Artificial Intelligence (AAAI)* 33, 01 (Jul. 2019), 4213–4220. https://doi.org/10.1609/aaai.v33i01.33014213

[20] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2015). arXiv:1509.02971 http://arxiv.org/abs/1509.02971

[21] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Neural Information Processing Systems (NIPS)*. 6382–6393.

[22] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. http://proceedings.mlr.press/v48/mniha16.html

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* (2015), 529–533.

[24] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. 2018. Data-Efficient Hierarchical Reinforcement Learning. *CoRR* abs/1805.08296 (2018). arXiv:1805.08296 http://arxiv.org/abs/1805.08296

[25] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer. https://doi.org/10.1007/978-3-319-28929-8

[26] Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P. How. 2019. Learning to Teach in Cooperative Multiagent Reinforcement Learning. *Association for the Advancement of Artificial Intelligence (AAAI)* 33, 01 (Jul. 2019), 6128–6136. https://doi.org/10.1609/aaai.v33i01.33016128

[27] Ronald Parr and Stuart Russell. 1998. Reinforcement Learning with Hierarchies of Machines. In *Neural Information Processing Systems (NIPS)*. 1043–1049. http://dl.acm.org/citation.cfm?id=302528.302894

[28] Matthew Riemer, Miao Liu, and Gerald Tesauro. 2018. Learning abstract options. In *Neural Information Processing Systems (NIPS)*.

[29] Everett M Rogers. 2010. *Diffusion of innovations*. Simon and Schuster.

[30] Stéphane Ross and J. Andrew Bagnell. 2014. Reinforcement and Imitation Learning via Interactive No-Regret Learning. *CoRR* abs/1406.5979 (2014). arXiv:1406.5979 http://arxiv.org/abs/1406.5979

[31] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics (AIStat)*, Vol. 15. 627–635. http://proceedings.mlr.press/v15/ross11a.html

[32] Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning* (1st ed.). MIT Press, Cambridge, MA, USA.

[33] Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* (1999). https://doi.org/10.1016/S0004-3702(99)00052-1

[34] Matthew E. Taylor and Peter Stone. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *J. Mach. Learn. Res.* 10 (Dec. 2009), 1633–1685. http://dl.acm.org/citation.cfm?id=1577069.1755839

[35] Andrea L. Thomaz and Cynthia Breazeal. 2006. Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance. In *Association for the Advancement of Artificial Intelligence (AAAI)*. 1000–1005. http://dl.acm.org/citation.cfm?id=1597538.1597696

[36] Lisa Torrey and Matthew Taylor. 2013. Teaching on a budget: Agents advising agents in reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1053–1060.

[37] Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. 2016. Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning. In *Association for Computational Linguistics (ACL)*. https://doi.org/10.18653/v1/P16-1013

[38] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. FeUdal Networks for Hierarchical Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. http://proceedings.mlr.press/v70/vezhnevets17a.html

[39] Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. 2019. Probabilistic Recursive Reasoning for Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=rkl6As0cF7

[40] Tianbing Xu, Qiang Liu, Liang Zhao, and Jian Peng. 2018. Learning to Explore with Meta-Policy Gradient. *CoRR* abs/1803.05044 (2018). arXiv:1803.05044 http://arxiv.org/abs/1803.05044