Learning Hierarchical Teaching Policies for Cooperative Agents

Dong-Ki Kim
MIT-LIDS
MIT-IBM Watson AI Lab
ddkim93@mit.edu

Miao Liu
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 considered 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

With this motivation in mind, we explore a new methodology for 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 [5, 33]. In this paper, we focus on transfer based on action advising [1, 4, 6, 35, 40], 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 learns peer-to-peer teaching policies that accelerate team-wide learning progress, 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 agent’s 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 [13], and these experiences can be randomly selected from a replay memory [20, 24]. 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 advising 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 that

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].
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) [25]. 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, 6, 26, 35, 40]. 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 (MARL) 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 partially observable Markov game, defined as a tuple \( \{I, S, A, T, O, R, \gamma, \pi\} \) [21]. \( I = \{1, \ldots, n\} \) is the set of agents, \( S \) is the set of states, \( A = \times_{i \in I} A_i \) is the set of joint actions, \( T \) is the transition probability function, \( O = \times_{i \in I} O_i \) is the set of joint observations, \( R = \times_{i \in I} R_i \) is the set of joint reward functions, and \( \gamma \in [0, 1) \) is the discount factor. The cooperative setting is a specialized case of the Markov games with a shared team reward function (i.e., \( R^t = \ldots = R^n \)). At each timestep \( t \), each agent \( i \) executes an action according to its policy \( \pi^t_i \sim \pi_i(a^t_i; \theta^i) \) parameterized by \( \theta^i \), where \( a^t_i \in O^t_i \) is the agent \( i \)'s observation at timestep \( t \). A joint action \( a_t = [a^t_1, \ldots, a^n_t] \) yields a transition from a current state \( s_t \in S \) to next state \( s_{t+1} \in S \) with probability \( T(s_t|s_{t+1}, a_t) \). Then, a joint observation \( o_{t+1} = [o^t_{t+1}, \ldots, o^n_{t+1}] \) is obtained and the team receives a shared reward \( r_t \). Each agent’s objective is to maximize the expected cumulative team reward \( \mathbb{E}[\sum_t y^t r_t] \). Note that for simplicity the policy parameter will often be omitted (i.e., \( \pi^t_i(o^t_i; \theta) \equiv \pi^t_i(o^t_i) \)).

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, \( \pi^t_i \) and \( \pi^t_j \), with learning algorithms, \( L^j \) and \( L^i \). 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 what and when to advise is referred to as the advice-level learning problem \( \mathcal{P}_{\text{advice}} \), where (•) denotes teacher property.

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 \( \mathcal{P}_{\text{advice}} \). 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 maximum horizon \( T \), but, for \( \mathcal{P}_{\text{advice}} \), an episode ends when task-level policies have converged, forming one “episode” for learning teacher policies. To distinguish these two concepts, we refer to an episode as one episode for \( \mathcal{P}_{\text{task}} \) and a session as one episode for \( \mathcal{P}_{\text{advice}} \) (see Figure 1).

Learning complexity. Another major difference between \( \mathcal{P}_{\text{task}} \) and \( \mathcal{P}_{\text{advice}} \) is in their learning objectives. \( \mathcal{P}_{\text{task}} \) aims to maximize cumulative reward per episode, whereas \( \mathcal{P}_{\text{advice}} \) 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 \( \mathcal{P}_{\text{advice}} \) 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\( T \) timesteps). In addition, teachers must consider heterogeneous knowledge (compactly represented by task-level observations, actions, and Q-values; see Section 4.1) 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, 8, 9, 16, 27, 28, 32, 37]. 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. [25] with a two-layer hierarchical structure: the higher-level manager policy \( \pi_M \) and the lower-level worker policy \( \pi_W \). The manager policy obtains an observation \( o_t \) and plans a high-level sub-goal \( g_t \sim \pi_M(o_t) \) for the worker policy. The worker policy attempts 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.

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).
by the worker. The manager learns to accomplish a task by optimizing the cumulative environment reward and stores an experience \(o_t, r_t, \sum_{t+1}^{t+H-1} r_j, o_{t+H}\) every \(H\) timesteps. By contrast, the worker learns to reach the sub-goal by maximizing the cumulative intrinsic reward \(r_{\text{intrinsic}}^{i}\) and stores an experience \(o_t, a_t, r_{\text{intrinsic}}^{i}, o_{t+1}\) at each timestep. Without loss of generality, we also denote the next observation with the prime symbol \(\dot{o}\).

### 3 HMAT OVERVIEW

HMAT improves the scalability of the learning and teaching in problems with larger domains and higher complexity by employing deep student policies and learning of high-level teacher policies that 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 in 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 in 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 \(\pi^t\) and \(\pi^{i}\) with deep hierarchical policies consisting of manager policies, \(\pi^M_t\) and \(\pi^M_{t'}\), and worker policies, \(\pi^W_t\) and \(\pi^W_{t'}\) (see Figure 2). Note that the manager and worker policies are trained with different objectives. Managers learn to accomplish a task together (i.e., solving \(P_{\text{task}}\)) 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. Hereafter, when we discuss task-level policies, we are implicitly only discussing the manager policies. The manager subscript \(M\) is often omitted when discussing these task-level policies to simplify notation (i.e., \(\pi^t \equiv \pi^M_t\)).

#### 3.2 Teacher Credit Assignment Issue

Learning advising policies requires estimates of the impact of the advice on the student agents’ learning progress [26], but these estimates are difficult to obtain especially with deep student policies. For example, consider an example, in which a teacher agent

\[\pi^j\text{ advises a student } i \text{ with advice } \tilde{g}^j_t.\] Following the advice, \(j\) obtains an experience of \(\{o^j_t, \tilde{g}^j_t, \sum_{t+1}^{t+H-1} r_j, o_{t+H}\}\). The agent \(i\) then requires feedback, reflecting the extent to which its advice \(\tilde{g}^j_t\) contributed to \(j\)’s learning progress, to be able to improve its teaching strategy. Note that, in general, a mini-batch of experiences is used to stably update the student’s deep task-level policy from \(\pi^t\) to \(\pi^{t'}\) [13, 20, 24], and this mini-batch consists of many experiences including the ones affected by both current advice \(\tilde{g}^j_t\) and previous advice \(\tilde{g}^j_0, \ldots, \tilde{g}^j_{t-1}\) (see Figure 3a). Consequently, \(j\)’s learning progress at timestep \(t\) is jointly affected by a batch of advice suggested at various times, and identifying the amount of current advice \(\tilde{g}^j_t\) contributes to \(j\)’s learning is challenging.

LeCTR simplifies the teacher credit assignment problem by assuming tile-coding task-level policies with online updates [26]. As such, the impact of \(\tilde{g}^j_t\) on \(j\)’s learning progress at timestep \(t\) can be easily observed as the update is directly correlated by the experience affected by current advice only (see Figure 3b). However, the assumption breaks with deep students due to the offline experiences, leading to incorrect credit assignment and learning of poor teachers (see Section 5.3), which is the core challenge that our contribution addresses.

#### 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 \(P_{\text{task}}\)) and how to provide advice using the teacher policies (i.e., solving \(P_{\text{advice}}\)). 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 [39], 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 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 1: 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 $\tilde{g}_i = (\tilde{g}_0_i, \tilde{g}_1_i, \ldots, \tilde{g}_{L_i}^t)$, where $\tilde{g}_0_i^t = (\tilde{g}_0, \tilde{g}_1^t, \tilde{g}_2^t, \ldots, \tilde{g}_{T_i}^t)$ denotes multiple advice by agent $i$ in one episode, instead of just providing one piece of advice $\tilde{g}_i = (\tilde{g}_0_i, \tilde{g}_1_i)$, 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, 6, 26, 35]. This extension is important: by following advice $[1, 4, 6, 26, 35]$. This extension is important: by following advice that the task-level policies were updated based on a single piece of advice, instead of just providing one piece of advice

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 $\tilde{g}_0^t$, where the temporary policies denote the copy of their current task-level policies (i.e., $\pi^{\text{temp}} \leftarrow \pi$) and $\pi = \{\pi^t, \pi^j\}$. To determine the teacher reward for $\tilde{g}_0^t$, $\pi^{\text{temp}}$ are updated for a small number of iterations using $E^{\text{advice}}$ (i.e., $\pi^{\text{temp}} \leftarrow \pi^{\text{advice}}$), where $\pi^{\text{advice}} \leftarrow \pi^{\text{advice}}$ is used to generate the allowed mini-batch updates of deep policies. These self-practice experiences based on $\pi^{\text{temp}}$ reflect how agents on their own would perform after the advising phase and can be used to estimate the impact of $\tilde{g}_0^t$ on team-wide learning. The teacher reward function $\hat{R}$ (see Section 4.1) uses the self-practice experiences to compute the teacher reward for $\tilde{g}_0^t$ (i.e., $\hat{R} = \hat{R}(E^{\text{self-prac}})$). The key point is that $\pi^{\text{temp}}$, which are used to compute $\hat{R}$, are updated based $E^{\text{advice}}$ only, so experiences from the past iterations do not affect the teacher reward estimation for $\tilde{g}_0^t$. Consequently, the use of temporary policies addresses the teacher credit assignment issue.

Phase III: Policy update. Task-level policies are updated by solving $P_{\text{task}}$ by using $L^j$ and $L^t$, and advice-level policies are updated to solve $P_{\text{advice}}$ by using learning algorithms $L_i^j$ and $L_i^t$. In particular, task-level policies, $\pi^t$ and $\pi^j$, are updated for the next iteration by randomly sampling experiences from task-level experience memories, $D^j$ and $D^t$. As in Xu et al. [39], both $E^{\text{advice}}(\pi^{\text{advice}})$ and $E^{\text{self-prac}}(\pi^{\text{temp}})$ 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, $D^j$ and $D^t$, teacher policies are updated by randomly selecting samples from their replay memories.

4 IMAT DETAILS

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

4.1 Advice-Level Policy Details

Teacher observation and action. Teacher-level observations $\tilde{o}_i = (\tilde{o}_i^t, \tilde{o}_i^j)$ compactly provide information about the nature of the heterogeneous knowledge between the two agents. Specifically, for agent $i$’s teacher policy, its observation $\tilde{o}_i^t$ consists of:

$\tilde{o}_i^t = \{o_i, g_i^j, Q^i(o_i, g_i^j, \tilde{g}_i^{(i)}), Q^t(o_i, g_i^j, \tilde{g}_i^{(j)})\}$

where $o_i = \{o_i^t, o_i^j\}$, $g_i^j \sim \pi^j(o_i^j)$, $g_i^j \sim \pi^i(o_i^j)$, $g_i^j \sim \pi^i(o_i^j)$, $Q^i$ 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 $R_{\text{remain}}$ is the remaining time in the session. Given $\tilde{o}_i^t$, 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 I 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 $R$ that can map self-practice experiences into better task-level learning performance. Intuitively, maximizing the teacher rewards $R$ 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 $E^{\text{self-prac}}$.

We also evaluate different choices of teacher reward functions, including the ones in Omidshafiei et al. [26] and Xu et al. [39], as described in Table 1.

Teacher experience. One teacher experience corresponds to $E_{\text{task}}^i = \{\tilde{o}_0^i, \tilde{g}_0^i, \tilde{g}_1^i, \ldots, \tilde{g}_T^i\}$, where $\tilde{o}_0^i = (\tilde{o}_0^i, \tilde{g}_0^i, \tilde{g}_1^i, \ldots, \tilde{g}_T^i)$ is the teacher observation, $\tilde{g}_0^i = (\tilde{g}_0^i, \tilde{g}_1^i, \ldots, \tilde{g}_T^i)$ is the teacher action, $\tilde{R}$ is

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

<table>
<thead>
<tr>
<th>Teacher Reward Function Name</th>
<th>Description</th>
<th>Teacher Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Estimation Gain (VEG) [26]</td>
<td>Student’s $Q$-value above a threshold $\tau$</td>
<td>$1 (Q_{\text{student}} &gt; \tau)$</td>
</tr>
<tr>
<td>Difference Rollout (DR) [39]</td>
<td>Difference in rollout reward before/after advising</td>
<td>$\hat{R} - R_{\text{before}}$</td>
</tr>
<tr>
<td>Current Rollout (CR)</td>
<td>Rollout reward after advising phase</td>
<td>$\hat{R}$</td>
</tr>
</tbody>
</table>
Algorithm 1 HMAT Pseudocode

Require: Maximum number of episodes in session $S$
Require: Teacher update frequency $f_{teacher}$
1. Initialize advice-level policies $\pi$ and memories $D$
2. for teaching session $t$ do
3. Re-initialize task-level policy parameters $\theta$
4. Re-initialize task-level memories $D$
5. Re-initialize train episode count: $e = 0$
6. while $e \leq S$ do
7. Phase I
6. Update episode count: $e \leftarrow e + 1$
9. Copy temporary task-level policies: $\pi^t_{temp} \leftarrow \pi$
10. Update to $\pi^t_{temp}$ using Eqn (1)-(2) with $E^{adv}_{0:T}$
11. Phase II
12. Update episode count: $e \leftarrow e + T/T$
13. $T \leftarrow$ Get teacher reward with $R$
14. Add $E^{adv}_{0:T}$ and $E^{self-prac.}_{0:T}$ to $D$
15. Add a teacher experience $E^{H}_{0:T}$ to $\tilde{D}$
16. Phase III
17. if $e \mod f_{teacher} = 0$ then
18. Update $\pi$ using Eqn (3)-(4) with $\tilde{D}$
19. end if
20. end while
21. end for

Figure 4: Two scenarios used for evaluating HMAT.

HMATD3 further extends MATD3 with HRL. In HMATD3, agent $i$’s task policy critics, $Q_i^1$ and $Q_i^2$, minimize the following critic loss:

$$L = \sum_{a=1}^{2} E_{(o,a,r,o')} [y - Q_i^{(\alpha)}(o,a)]^2,$$

s.t. $y = r + \gamma \min_{\hat{a}} Q_i^{(\beta)}(o',\hat{a}), \pi_{target}(o') + \epsilon),$

where $o = (o',\hat{o}), g = (g',\hat{g}), \alpha' = (\alpha',\hat{\alpha}'), \pi_{target} = (\pi_{target}'),$ the subscript “target” denotes the target network, and $\epsilon \sim N(0, \sigma).$ The agent $i$’s actor policy $\pi_i$ with parameter $\theta_i$ is updated by:

$$\nabla_{\theta_i} J(\theta_i) = E_{o \sim D^r} \left[ \nabla_{\theta_i} \pi_i(o) \nabla_{\theta_i} Q_i(o,a) | g = \pi_i(o) \right].$$

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, $Q_i^1$ and $Q_i^2$, minimize the following critic loss:

$$L = \sum_{a=1}^{2} E_{(o,a,r,o')} [y - Q_i^{(\alpha)}(o,a)]^2,$$

s.t. $y = r + \gamma \min_{\hat{a}} Q_i^{(\beta)}(o',\hat{a}), \pi_{target}(o') + \epsilon),$

where $o = (o',\hat{o}), g = (g',\hat{g}), \alpha' = (\alpha',\hat{\alpha}'), \pi_{target} = (\pi_{target}'),$ the subscript “target” denotes the target network, and $\epsilon \sim N(0, \sigma).$ The agent $i$’s actor policy $\pi_i$ with parameter $\theta_i$ is updated by:

$$\nabla_{\theta_i} J(\theta_i) = E_{o \sim D^r} \left[ \nabla_{\theta_i} \pi_i(o) \nabla_{\theta_i} Q_i(o,a) | g = \pi_i(o) \right].$$

5 EVALUATION
We demonstrate HMAT’s performance in increasingly challenging domains that involve continuous states/actions, long horizons, and delayed rewards.

5.1 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, 38] 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.

5.2 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.

5.3 Results on One Box and Two Box Push
Table 2 compares HMAT and its baselines. The results show both final task-level performance (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 MATD3 and with MAT compared to MATD3, as indicated by the higher final performance (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 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 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 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 compared to VEG and DR. 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.

5.4 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.
Table 2: $\bar{V}$ and AUC for different algorithms. Results show a mean and standard deviation computed for 10 sessions. Best results in bold (computed via a t-test with $p < 0.05$).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Hierarchical?</th>
<th>Teaching?</th>
<th>One Box Push</th>
<th>Two Box Push</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATD3</td>
<td></td>
<td></td>
<td>$-12.64 \pm 3.42$</td>
<td>$-44.21 \pm 5.01$</td>
</tr>
<tr>
<td>LeCTR-Tile</td>
<td></td>
<td></td>
<td>$-18.00 \pm 0.01$</td>
<td>$-60.50 \pm 0.05$</td>
</tr>
<tr>
<td>LeCTR-D</td>
<td></td>
<td></td>
<td>$-14.85 \pm 2.67$</td>
<td>$-46.07 \pm 3.00$</td>
</tr>
<tr>
<td>LeCTR-OHD</td>
<td></td>
<td></td>
<td>$-17.92 \pm 1.01$</td>
<td>$-60.43 \pm 0.08$</td>
</tr>
<tr>
<td>AI</td>
<td></td>
<td></td>
<td>$-11.33 \pm 2.46$</td>
<td>$-41.09 \pm 3.50$</td>
</tr>
<tr>
<td>AICI</td>
<td></td>
<td></td>
<td>$-10.60 \pm 0.85$</td>
<td>$-38.23 \pm 0.45$</td>
</tr>
<tr>
<td>MAT (with CR)</td>
<td></td>
<td></td>
<td>$-10.04 \pm 0.38$</td>
<td>$-39.49 \pm 2.93$</td>
</tr>
<tr>
<td>HMA3D</td>
<td>✓</td>
<td></td>
<td>$-10.24 \pm 0.20$</td>
<td>$-31.55 \pm 3.51$</td>
</tr>
<tr>
<td>LeCTR-HD</td>
<td>✓</td>
<td></td>
<td>$-12.10 \pm 0.94$</td>
<td>$-38.87 \pm 2.94$</td>
</tr>
<tr>
<td>LeCTR-OHD</td>
<td>✓</td>
<td></td>
<td>$-16.77 \pm 0.66$</td>
<td>$-60.23 \pm 0.33$</td>
</tr>
<tr>
<td>HAI</td>
<td>✓</td>
<td></td>
<td>$-10.23 \pm 0.19$</td>
<td>$-31.37 \pm 3.71$</td>
</tr>
<tr>
<td>HAICI</td>
<td>✓</td>
<td></td>
<td>$-10.25 \pm 0.26$</td>
<td>$-29.32 \pm 1.19$</td>
</tr>
<tr>
<td>HMAT (with VEG)</td>
<td>✓</td>
<td></td>
<td>$-10.38 \pm 0.25$</td>
<td>$-29.73 \pm 2.89$</td>
</tr>
<tr>
<td>HMAT (with DR)</td>
<td>✓</td>
<td></td>
<td>$-10.36 \pm 0.32$</td>
<td>$-28.12 \pm 1.58$</td>
</tr>
<tr>
<td>HMAT (with CR)</td>
<td>✓</td>
<td></td>
<td>$-10.10 \pm 0.19$</td>
<td>$-27.49 \pm 0.96$</td>
</tr>
</tbody>
</table>

Heterogeneous Action Result

Figure 5: (a) and (b) Task-level learning progress in COBP and CTBP, respectively. The oracles in (a) and (b) refer to the performance of converged HMA3D. 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.

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 before, it achieves an AUC of $400 \pm 10$, compared to no-teaching baseline (HMA3D) AUC of $369 \pm 27$.

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 $363 \pm 43$, task-level learning with teaching achieves AUC of $414 \pm 11$. 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., $180^\circ$ 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.
5.5 HMAT Analyses

Teacher reward accuracy. Developing ground-truth learning progress of task-level policies often requires an expert policy and could be computationally undesirable [14]. 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 [7, 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 [24], 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) [23]. 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).

6 RELATED WORK

Action advising is not the only possible approach to transfer knowledge. Works on imitation learning studies on how to learn a policy from expert demonstrations [7, 30, 31]. Recent work applied imitation learning for multiagent coordination [18] and explored effective combinations of imitation learning and HRL [17]. Curriculum learning [3, 14, 36], 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 [10, 15]. Another work of Thomaz and Breazeal [34] 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. [39] 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. [39], and a new reward function of CR that is empirically shown to perform better in our domains.

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. In the future, we will extend this work along with the following directions: 1) investigating ways to expand HMAT and associated learning to teach methods to problems involving more than two agents, 2) studying theoretical properties of HMAT, and 3) applying HMAT to more complex domains that may require more than two levels of hierarchies.

ACKNOWLEDGEMENTS

Research funded by IBM (as part of the MIT-IBM Watson AI Lab initiative) and computational support through Amazon Web Services. Dong-Ki Kim was also supported by a Kwanjeong Educational Foundation Fellowship.