Heterogeneous Knowledge Transfer via Hierarchical Teaching in Cooperative Multiagent Reinforcement Learning

Dong-Ki Kim[1,2] dkkim93@mit.edu
Miao Liu[2,3] miao.liu1@ibm.com
Shayegan Omidshafiei[1,2] shayegan@mit.edu
Sebastian Lopez-Cot[1,2] slcot@mit.edu
Matthew Riemer[2,3] mdriemer@us.ibm.com
Gerald Tesauro[2,3] gtesauro@us.ibm.com
Murray Campbell[2,3] mcam@us.ibm.com
Sami Mourad[2,3] Sami.Mourad@ibm.com
Golnaz Habibi[1,2] golnaz@mit.edu
Jonathan P. How[1,2] jhow@mit.edu

[1]LIDS, MIT [2]MIT-IBM Watson AI Lab [3]IBM Research

Abstract

Heterogeneous knowledge naturally arises among different agents in cooperative multiagent reinforcement learning (MARL). Existing works have demonstrated that peer-to-peer knowledge transfer, a process referred to as *teaching*, accelerates individual learning speed and improves team-wide performance. Similar to recent learning to teach frameworks, we aim to learn teaching policies that decide *when* and *what* to advise to a teammate. In this work, we introduce a new learning to teach framework, called Hierarchical Multiagent Teaching (HMAT). The proposed framework solves difficulties faced by existing works when operating in domains with long horizons, large state spaces, and continuous actions. HMAT transfers heterogeneous knowledge by taking advantage of temporal abstractions of hierarchical reinforcement learning and representations of a deep neural network. Our empirical evaluations show that HMAT accelerates team-wide learning progress in complex environments.

Introduction

Peer-to-peer knowledge transfer among different agents in cooperative multiagent reinforcement learning (MARL) can accelerate individual learning speed and improve team-wide performance (Vrancx, De Hauwere, and Nowe 2011; Taylor and Galvan-Lopez 2013; Garant, da Silva, and Lesser 2015; da Silva, Glatt, and Costa 2017; Omidshafiei et al. 2018). This improvement primarily results from the fact that agents can obtain specialized skill knowledge by visiting local states and then pass on that knowledge to uninformed teammates, a process referred to here as *teaching*.

How can we obtain an effective teaching in MARL? Indeed this is an open question, for even in the context of humans, determining what makes a teacher effective is subjective (Rockoff and Speroni 2011). We define effective teaching as solving the problem of *action advising*: learning *when* and *what* actions to advise to a teammate. Optimizing for these two objectives is a complicated problem. An ideal teaching framework requires an accurate estimate of a student's instantaneous learning progress. As noted in recent works on learning to teach (Omidshafiei et al. 2018), this estimate is difficult to obtain given that students can follow many different learning trajectories in arriving at final converged policies. In addition, developing ground-truth data for approximating student learning progress requires either an oracle for the environment or a pre-trained expert, neither of which may be available in practical settings. Similarly, deciding advice is difficult because effective teachers must not only advise their best action choices but also guide their students to explore when necessary. Despite these difficulties, significant progress has been made in the MARL community on designing effective multiagent teaching frameworks that encourage agents to learn from each others' unique experiences to accelerate the overall learning.

Previous work on action advising can be classified into three categories: optimal-advising-, heuristics-, and learning-based. The optimal-advising-based methods (Taylor et al. 2014; Fachantidis, Taylor, and Vlahavas 2017) require the existence of an expert teacher (ground-truth advising knowledge). The heuristic-based teaching methods (da Silva, Glatt, and Costa 2017) use various rules, such as the state visit frequency to determine the advice to give. However, assuming ground-truth advising knowledge is often not practical and developing heuristics for effective teaching is rather subjective.

Learning-based approaches develop a more general and reliable way to teach. A noticeable example is the LeCTR framework (Omidshafiei et al. 2018), which allows each agent to learn when and what to advise, enabling significantly faster learning progress. However, LeCTR was designed to handle primitive action level advising and thus could only be applied to domains with discrete action and state spaces (similar to most other works on teaching for MARL). With delayed reward assignment, primitive action-based methods are usually difficult to scale for domains with long horizons and can cost a significant amount of expenses for advising. Moreover, LeCTR only considered tasks for which shallow policy representations (tabular, tile-coding) are efficient. However, it is well known that these shallow policy representations do not scale to continuous action spaces due to the curse of dimensionality (Sutton and Barto 1998). Although one might conjecture that the second issue can be addressed by using deep neural network (DNN) policy representations, training DNNs introduces a more difficult credit assignment dilemma: identifying which portions of teacher advice led to successful student learning is difficult when randomly sampling experiences from a replay memory and performing batch updates. Hence, LeCTR cannot be easily adapted to domains requiring DNN policy representations.

To solve difficulties faced by existing methods for learning to teach in MARL when operating in domains with long horizons, large state spaces, and continuous actions, we propose a new framework called Hierarchical Multi-agent Teaching (HMAT). The new framework is created by using an improved teacher reward function that enables HMAT to employ DNN based policy representations and provides a more accurate estimate of a student's learning progress than teacher reward functions in previous frameworks (Omidshafiei et al. 2018). HMAT also utilizes a hierarchical learning formulation (Kulkarni et al. 2016; Bacon, Harb, and Precup 2017; Vezhnevets et al. 2017; Nachum et al. 2018) in order to efficiently learn and transfer knowledge in more complex domains. To our knowledge, HMAT is the first framework to take advantage of temporal abstractions to transfer high-level actions (or sub-goals) amongst cooperative agents in a learning-to-teach setting. Our experiments show that HMAT accelerates student learning in multiple environments as compared to related baselines.

Background

This work considers a cooperative MARL setting in which agents jointly interact in the environment and receive a shared team reward. This setting can be formalized as a partially observable Markov game (Littman 1994), defined by $\langle \mathcal{I}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \mathcal{O}, \mathcal{T}, \gamma \rangle$, where $\mathcal{I}$ is a set of N agents, $\mathcal{S}$ is a set of states, $\boldsymbol{\mathcal{A}} = \langle A^1, A^2, \ldots, A^N \rangle$[1] is a set of action sets for all N agents, and $\boldsymbol{\mathcal{O}} = \langle O^1, O^2, \ldots, O^N \rangle$ is a set of observations for all N agents. Each agent i executes an action according to its stochastic policy $\pi_{\theta^i} : O^i \times A^i \mapsto [0,1]$, which yields the next state by the state transition function $\mathcal{T} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \mapsto \mathcal{S}$. Each agent i obtains a reward $r^i : \mathcal{S} \times A^i \mapsto \mathbb{R}$, and receives an observation $o^i : \mathcal{S} \mapsto O^i$. Each agent i has an objective to maximize its total expected return $R^i = \mathbb{E}[\sum_t \gamma^t r^i_t]$, where $\gamma \in [0,1)$ is the discount factor. The cooperative setting is a specialized case of the Markov games with a shared team reward.

Off-Policy Deterministic Policy Gradient

The off-policy deterministic policy gradient (DPG) (Silver et al. 2014) is an RL framework for continuous action spaces. Deep deterministic policy gradient (DDPG) (Lillicrap et al. 2015) expands DPG by using DNNs to approximate the actor and the critic. DDPG also samples experiences from a replay memory $\mathcal{D}$ and uses a target network, as in DQN (Mnih et al. 2015a). In our work, we utilize TD3 (Fujimoto, van Hoof, and Meger 2018), a variant of DDPG that addresses the overestimation issue of DPG. TD3 involves a deterministic policy μ_θ (i.e., $\mu_\theta : \mathcal{S} \mapsto \mathcal{A}$) and two critics, Q^{θ_1} and Q^{θ_2}. Critics minimize the following loss:

$$\begin{aligned} \mathcal{L} &= \sum\nolimits_{i=1}^{2} \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left(y - Q^{\theta_i}(s,a) \right)^2 \\ y &= r + \gamma \min_{j=1,2} Q^{\theta'_j}(s', \mu_{\theta'}(s') + \epsilon) \end{aligned} \tag{1}$$

where θ', θ'_1, and θ'_2 are the target network parameters; and $\epsilon \sim \mathcal{N}(0, \sigma)$. The policy μ_θ is updated by:

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{s \sim \mathcal{D}} \left[\nabla_\theta \mu_\theta(a|s) \nabla_a Q^{\theta_1}(s,a)|_{a=\mu_\theta(s)} \right] \tag{2}$$

In a MARL setting, the environment appears non-stationary from the perspective of any agent because all of the agents are simultaneously learning together (Omidshafiei et al. 2017). To address non-stationary issues in MARL, recent works (Lowe et al. 2017; Foerster et al. 2017) introduce the framework of centralized training (e.g., centralized critics) with decentralized execution (e.g., decentralized actors). This paper adopts the same approach (centralized critics and the decentralized actors).

Hierarchical Reinforcement Learning

Hierarchical reinforcement learning (HRL) provides a structured framework with multi-level reasoning and extended temporal abstraction (Kulkarni et al. 2016; Bacon, Harb, and Precup 2017; Vezhnevets et al. 2017; Nachum et al. 2018). HRL offers a benefit over non-hierarchical RL in solving complex tasks. The closest HRL framework that we use in this paper is that of (Nachum et al. 2018) with a two-layer hierarchical structure: the higher-level policy (manager policy $\mu_{\mathcal{M}}$) and the lower-level policy (worker policy $\mu_{\mathcal{W}}$). The manager policy obtains an observation o_t and plans a high-level sub-goal $g_t \sim \mu_{\mathcal{M}}(o_t)$ for the worker policy. The worker policy attempts to reach this sub-goal from current state by executing a primitive action $a_t \sim \mu_{\mathcal{W}}(o_t, g_t)$ in the environment. Following this, an updated sub-goal is generated by the manager every horizon h time steps[2] and a sequence of primitive actions are executed by the worker. The manager policy learns to accomplish a task by optimizing the cumulative environment reward. By contrast, the worker policy learns to reach the sub-goal by maximizing the cumulative intrinsic reward. An example intrinsic reward can be the negative distance between the current observation and the sub-goal: $r_t^{\text{intrinsic}} = -||o_t - g_t||_2^2$, where o_t and g_t can be (x,y) coordinate in a 2D environment. Lastly, the manager stores an experience $\langle o_t, g_t, r_{t:t+h-1}, o_{t+h} \rangle$ every h time steps and the worker stores an experience $\langle o_t, a_t, r_t^{\text{intrinsic}}, o_{t+1} \rangle$ every step.

Hierarchical Learning to Teach Overview

The previous work, LeCTR (Omidshafiei et al. 2018) established a general learning to teach in a MARL framework, enabling the transfer of knowledge among cooperative, heterogeneous agents to accelerate team-wide learning progress. Agents share knowledge via "teacher" policies that learn *when* and *what* to advise fellow agents. This proposed work explores a novel algorithm for learning high-level teacher

[1]Superscript denotes a property for an agent (e.g., an action set for agent 1: A^1), Additionally, a bold symbol represents a set including N agents' property.

[2]Manager samples a sub-goal when $t = nh$ with n=0,1,2,...

policies by introducing HRL and nonlinear function approximators (e.g., deep neural networks) that scale up the LeCTR approach to more difficult tasks involving continuous action spaces. Before describing our framework in detail, we provide a high-level overview of the approach in this section.

Task-Level Learning Problem

Following the conventions from LeCTR (Omidshafiei et al. 2018), we consider a standard cooperative MARL setting with two agents i and j in a shared environment. At each learning iteration, agents interact with the environment, collect experiences, and update their policies, μ^i and μ^j, with learning algorithms, $\mathbb{L}^i$ and $\mathbb{L}^j$. The resulting policies learn to coordinate and maximize final performance in the task. This problem of learning task-related policies to coordinate is referred to as the task-level learning problem $\mathcal{P}_{\text{Task}}$.

Hierarchical Task-Level Policy: In this work, we extend task-level policies with HRL. To this end, we replace μ^i and μ^j with hierarchical policies consisting of manager policies, $\mu^i_{\mathcal{M}}$ and $\mu^j_{\mathcal{M}}$, and worker policies, $\mu^i_{\mathcal{W}}$ and $\mu^j_{\mathcal{W}}$ as shown in Figure 1. Note that manager and worker policies have different objectives. Manager policies learn to accomplish a task together by optimizing the cumulative environment reward. By contrast, worker policies are trained to successfully reach sub-goals suggested by managers.

Here, we focus on transferring knowledge at the manager policy level instead of the worker policy level since manager policies represent abstract knowledge which is more relevant to fellow agents. Therefore, hereafter we refer to managers as the task-level polices. To simplify notation, the manager subscript $\mathcal{M}$ will often be omitted when discussing task-level policies (e.g., $\mu^i = \mu^i_{\mathcal{M}}, \mu^j = \mu^j_{\mathcal{M}}$).

Knowledge Heterogeneity: During the task-level policy learning, knowledge heterogeneity among agents is likely to arise. This, for instance, occurs when an agent j enters a foreign state its teammate i has already visited and mastered. There are several likely causes for the existence of knowledge heterogeneity in MARL. For one, the underlying stochastic transition process naturally causes each agent to explore a different section of the environment. As a result, agents would contain unique experiences, develop different skills, and accumulate heterogeneous knowledge. Additionally, varying prior experiences among agents may cause knowledge heterogeneity.

Advice-Level Learning Problem

Throughout task-level policy learning, the ability for agents to exchange their heterogeneous knowledge among teammates is likely to accelerate team-wide learning progress. In this work, we consider sharing knowledge between task-level policies via teacher policies which advise task-level policies. Instead of using hand-crafted heuristics to advise, we train teacher policies that learn *when* and *what* to advise. Ideally, teacher policies should learn how to transform local knowledge into appropriate advice such that team-wide learning progress improves by following suggested advice. A desirable teacher policy is also one which only instructs when necessary and in a way to promote organic exploration

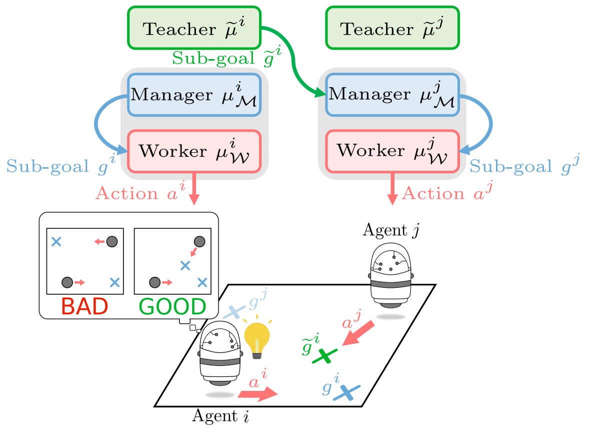


Figure 1: Agents teach each other according to the advising protocol. Knowledgeable agent i will evaluate the actions agent j intended to take and will advise if necessary. Advising occurs between the teacher policy of one agent (e.g., $\tilde{\mu}^i$) and the manager policy of another (e.g., $\mu^j_{\mathcal{M}}$). The common object they exchange is a sub-goal, which is fed to the worker policy to follow.

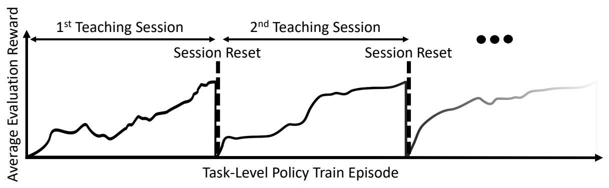


Figure 2: An example of task-level learning progress for each session. Upon reaching completion of a pre-determined episode count, task-level policies are re-initialized (teaching session reset). With each new teaching session, the teacher policies become better at advising students and thus resulting in task-level policies attaining faster learning progress.

so that task-level policies are able to succeed in the teacher's absence. Lastly, a teacher reward should accurately reflect the amount that advice contributed to improving team-wide learning progress. In (Omidshafiei et al. 2018), this problem of learning teacher policies is referred to as the advice-level learning problem, $\tilde{\mathcal{P}}_{\text{Advise}}$[3].

Advice Protocol: We describe how agents in HMAT interact together with teacher policies, $\tilde{\mu}^i$ and $\tilde{\mu}^j$. Consider Figure 1, where agents are learning to coordinate with hierarchical task-level policies (i.e., solving $\mathcal{P}_{\text{Task}}$) while advising one another via teacher policies (i.e., solving $\tilde{\mathcal{P}}_{\text{Advise}}$). In our framework, there are two roles: the role of the student agent j (i.e., an agent who's manager policy receives advice) and the role of the teacher agent i (i.e., an agent whose teacher policy gives advice). Agents i and j are able to teach each other, but we only consider a one-way interaction in Figure 1 for clarity. Here, the student j has decided that it is appropriate to strive for sub-goal g^j by querying its manager policy. Before j passes the sub-goal g^j to its worker, i's teaching policy checks j's intended sub-goal and considers i and j's heterogeneous knowledge at the task-level in

[3] $\tilde{(\cdot)}$ refers to a teacher policy property.

order to decide whether to advise or not. Having decided to advise, i transforms its local knowledge into desirable sub-goal advice via its teacher policy and suggests it to j. After student j accepts the suggested advice from the teacher, the updated sub-goal g^j is now passed to j's worker policy and the worker policy executes an action a^j.

Task-Level Problem vs Advice-Level Problem: We summarize a few important differences between the task-level problem $\mathcal{P}_{\text{Task}}$ and the advice-level problem $\widetilde{\mathcal{P}}_{\text{Advise}}$.

Firstly, the length of episodes in each are different. For the task-level learning problem, an episode terminates either when agents arrive at terminal states or a time step t exceeds a maximum episode time step T. However, for the advice-level problem, an episode ends when task-level policies have converged, forming one "episode" for learning teaching policies. Upon completion of the advising-level episode, task-level policies are re-initialized and training proceeds for another advising-level episode. To avoid confusion, we refer to an *episode* as one task-level problem episode and a *session* as one advice-level problem episode (Figure 2).

Secondly, the objectives of these two are also different. The goal of the task-level learning is to coordinate and maximize the cumulative environment reward in an episode. On the contrary, the goal of the advice-level learning is to maximize the cumulative teacher reward in a session, which corresponds to accelerating team-wide learning progress (i.e., a maximum area under the learning curve in one session).

Lastly, task-level policies are inherently off-policy while teacher policies are not necessarily off-policy. This is because task-level policies are updated by experiences affected by teacher policies, instead of experiences generated by task-level policies alone (i.e., behavior policy differs from the target policy). Thus, we use an off-policy learning algorithm (e.g., TD3 (Fujimoto, van Hoof, and Meger 2018)) to learn task-level policies.

Limitation of Existing Teaching Works: A key limitation of previous learning-to-teach frameworks, such as LeCTR (Omidshafiei et al. 2018) is the difficulty to scale to more complicated tasks (e.g., a task involving continuous action spaces). Previous frameworks have developed with task-level policies represented by look-up tables or tile-codings in order to ease training stability of teacher policies. Stability of teachers is less of a concern in these cases because the immediate teaching reward is more correlated with TD errors obtained from the online update as opposed to sampling from a replay memory or a batch update. As a result, teacher reward assignment is accurate since the task policy update uses the agent's experience at t affected by primitive action advice $\tilde{a}_t$, which then returns the teacher reward $\tilde{r}_t$ that reflects how much $\tilde{a}_t$ contributed to improving the learning progress at t. However, this stability is achieved with a consequence: tabular and tile-coding task policies are vulnerable to the curse of dimensionality and thus can be inefficient for solving tasks involving continuous actions.

Recently, RL with nonlinear function approximators (e.g., DNNs) has been applied to solve challenging tasks including those with continuous actions (Lillicrap et al. 2016; Duan et al. 2016). Although DNN policies can scale better than tabular or tile-coding policies, it is not trivial to directly apply DNNs to existing teaching frameworks, especially because of the teacher reward assignment issue. DNN policies often require a batch of experiences with random samples for each update to stabilize learning (Mnih et al. 2015b). As a result, given primitive action advice $\tilde{a}_t$, the DNN task policies will update using a batch of experiences not only affected by $\tilde{a}_t$ but also advice suggested at previous times. As a result, the teacher reward $\tilde{r}_t$ returned by task policies after the update would not reflect well how much $\tilde{a}_t$ was responsible in improving the student's learning progress at time t. This teacher reward assignment issue is significant because noisy or wrong estimate of teacher reward will lead to wrong or diverging teacher policies. This challenge is not addressed in previous teaching frameworks. We address the issue in detail in the next section.

Algorithm 1 HMAT Pseudocode

```
Require: A maximum number of episodes in a session $E$
 1: Initialize advice-level policy $\widetilde{\boldsymbol{\mu}}$
 2: for Teaching session $\widetilde{s} = 1$ do
 3:     Re-initialize task-level policy parameters $\boldsymbol{\mu}$
 4:     Re-initialize train episode count: $e = 1$
 5:     while $e \leq E$ do
 6:         $\boldsymbol{E}_{0:T}, \langle \widetilde{\boldsymbol{o}}_{0:T}, \widetilde{\boldsymbol{g}}_{0:T} \rangle \leftarrow$ Agents advise one another during one episode (Phase I)
 7:         Update e: $e \leftarrow e + 1$
 8:         Copy temporary task-level policies: $\boldsymbol{\mu}_{\text{temp}} \leftarrow \boldsymbol{\mu}$
 9:         Update $\boldsymbol{\mu}_{\text{temp}}$: $\boldsymbol{\mu}'_{\text{temp}} \leftarrow \mathbb{L}(\boldsymbol{\mu}_{\text{temp}}, \boldsymbol{E}_{0:T})$
10:         $\widetilde{r}, \boldsymbol{E}_{0:T_{\text{eval}}} \leftarrow$ Get teacher reward by rolling out for $n_{\text{eval}}$ episodes (Phase II)
11:         Add $\boldsymbol{E}_{0:T}$ and $\boldsymbol{E}_{0:T_{\text{eval}}}$ to task-level memories $\boldsymbol{D}$
12:         Update e: $e \leftarrow e + n_{\text{eval}}$
13:         Update $\boldsymbol{\mu}$ by using samples in $\boldsymbol{D}$ (Phase III)
14:         Add a teacher experience $\langle \widetilde{\boldsymbol{o}}_{0:T}, \widetilde{\boldsymbol{g}}_{0:T}, \widetilde{r}, \widetilde{\boldsymbol{o}}'_{0:T} \rangle$ to teacher memories $\widetilde{\boldsymbol{D}}$
15:         Update $\widetilde{\boldsymbol{\mu}}$ by using samples in $\widetilde{\boldsymbol{D}}$ (Phase IV)
16:     end while
17: end for
```

HMAT: Hierarchical Multiagent Teaching

This section summarizes the overall algorithm of our learning to teach framework, Hierarchical Multiagent Teaching (HMAT), and explains remaining details about learning teacher policies.

Overview of Algorithm

Agents in HMAT iterate over four phases to simultaneously learn how to coordinate (i.e., solving $\mathcal{P}_{\text{Task}}$) and how to advise one another (i.e., solving $\widetilde{\mathcal{P}}_{\text{Advise}}$):

- **Phase I (Advising Phase)**: Agents advise one another using the advising protocol during one episode. This process generates a batch of task-level experiences influenced by the teachers' advice.
- **Phases II (Evaluation Phase)**: This phase evaluates and estimates the amount that the teachers' advice will con-

tribute to improving team-wide learning progress by using a teacher reward function. As a result, this process yields the teacher reward.

- **Phase III (Task-Level Policy Update Phase)**: Task-level policies are updated by randomly sampling experiences from the task-level experience memories.
- **Phase IV (Advice-Level Policy Update Phase)**: A teacher experience that includes the teacher reward in Phase II is added to the teacher-level experience memories. Then, by using experiences randomly sampled from teacher memories, teacher policies are updated.

We explain each phase by demonstrating how issues specific to learning teacher policies are addressed.

Teacher Policy Details

Resolving Teacher Reward Assignment Issue: Our teacher policies require feedback (i.e., teacher reward) that accurately reflects learning progress improvement by given advice. However, with task-level policies represented by DNNs, estimating a correct teacher reward when using batch updates is difficult. To resolve this reward issue, we adopt methods developed for learning a single agent exploration policy (Xu et al. 2018) into our multiagent learning to teach setting.

Specifically, we address the issue with the following ideas. Firstly, a view of teacher policies is enlarged by providing multiple advice $\widetilde{\boldsymbol{g}}_{0:T}$ [4] before updating task-level policies. This contrasts to previous learning-to-teach approaches, where task-level policies are updated based on a single advice. As a result of providing multiple advice $\widetilde{\boldsymbol{g}}_{0:T}$, a batch of task-level experiences $\boldsymbol{E}_{0:T} = \langle \boldsymbol{o}_{0:T}, \widetilde{\boldsymbol{g}}_{0:T}, \boldsymbol{r}_{0:T}, \boldsymbol{o}'_{0:T}\rangle$ are generated in the **phase I** and satisfy the desirable batch update for DNNs.

Secondly, HMAT utilizes two separate task-level updates: one update to accurately estimate teacher reward $\widetilde{r}$ (**phase II**) and the other update to efficiently train task-level policies (**phase III**). The former update includes copying current task-level policies $\boldsymbol{\mu}$ to temporary task-level policies, $\boldsymbol{\mu}_{\text{temp}}$. These temporary policies are updated a small number of steps by using $\boldsymbol{E}_{0:T}$: $\boldsymbol{\mu}'_{\text{temp}} \leftarrow \mathbb{L}(\boldsymbol{\mu}_{\text{temp}}, \boldsymbol{E}_{0:T})$. Then a teacher reward function $\widetilde{\mathcal{R}}$ utilizes the updated temporary policies and returns a teacher reward $\widetilde{r} = \widetilde{\mathcal{R}}(\boldsymbol{\mu}'_{\text{temp}})$. Note the temporary policies are used only to measure $\widetilde{r}$ and then discarded (i.e., copied again at next train iteration). On the other hand, the latter update trains current task-level policies by randomly sampling experiences from the task-level experience memories $\boldsymbol{\mathcal{D}}$. As a result, we can estimate the contribution of $\widetilde{\boldsymbol{g}}_{0:T}$ to the learning progress improvement but also use the advantages of the replay memory to stabilize learning of task-level policies (Liu and Zou 2017; Xu et al. 2018). In the analysis section, we empirically show that these ideas result in the teacher reward, which closely estimates the true learning progress.

Teacher Reward Function: Given the updated temporary policies $\boldsymbol{\mu}'_{\text{temp}}$, the teacher reward function $\widetilde{\mathcal{R}}$ estimates $\widetilde{r}$ by executing the updated temporary policies for n_{eval} episodes that equals to T_{eval} steps (i.e., rollout data). This process generates a batch of evaluation experiences $\boldsymbol{E}_{0:T_{\text{eval}}}$, where $\widetilde{r}$ is obtained by summing up rewards in the evaluation batch. Note that the experiences are generated without the teacher policies being involved, so $\boldsymbol{E}_{0:T_{\text{eval}}}$ evaluates how students perform by themselves after one advising phase.

Teacher Policy Input and Output: A teacher policy makes its decision, i.e., when/what to advise, based on a teacher's and a student's heterogeneous knowledge. An agent's knowledge is represented by the policy parameters. However, as task-level policies in our framework are DNNs possibly with millions of parameters, it is not feasible to learn over the parameter space. Instead, teacher policies learn over teacher-level observations $\widetilde{\boldsymbol{o}}_t = \langle \widetilde{o}^i_t, \widetilde{o}^j_t\rangle$, selected to compactly provide information about agents' knowledge.

Our agents in HMAT can simultaneously advise one another. For clarity, we detail a teacher policy's input when agents i and j are a teacher and student agent, respectively (Figure 1). For agent i's teacher policy, its observation $\widetilde{o}^i_t$ consists of:

$$\widetilde{o}^i_t = \langle \underbrace{\boldsymbol{o}_t, g^i_t, g^{ij}_t, Q^i(\boldsymbol{o}_t, g^i_t, g^j_t), Q^i(\boldsymbol{o}_t, g^i_t, g^{ij}_t),}_{\text{Teacher Knowledge}}$$
$$\underbrace{g^j_t, Q^j(\boldsymbol{o}_t, g^i_t, g^j_t), Q^j(\boldsymbol{o}_t, g^i_t, g^{ij}_t)}_{\text{Student Knowledge}}, t_{\text{remain}}\rangle \tag{3}$$

where $t = nh$ with $n = 0, 1, 2, \ldots$ and the horizon h; $\boldsymbol{o}_t = \langle o^i_t, o^j_t\rangle$; $g^i_t \sim \mu^i(o^i_t)$; $g^j_t \sim \mu^i(o^j_t)$; $g^{ij}_t \sim \mu^i(o^j_t)$; Q^i and Q^j are the centralized critic for agent i and j, respectively; and t_{remain} is the remaining iteration until current session ends. Given an input $\widetilde{o}^i_t$, teacher i decides when/what to advise: one action for deciding whether to or not to advise; the other action for sub-goal advice. Given no advice, student j executes its originally intended sub-goal g^j_t.

Teacher Experience: With the enlarged teacher policy perspective, we explain *one* teacher experience considering teacher i for clarity. *One* teacher observation corresponds to $\widetilde{o}^i_{0:T}$, which consists of multiple $\widetilde{o}^i_t$. *One* teacher action is advice $\widetilde{g}^i_{0:T}$, and *one* next teacher observation can be obtained by updating each $\widetilde{o}^i_t$ in $\widetilde{o}^i_{0:T}$ with the updated temporary policy μ'^j_{temp} (thus, representing change in student knowledge due to advice $\widetilde{g}^i_{0:T}$). Finally, with the measured teacher reward $\widetilde{r}$, *one* teacher experience corresponds to $\langle \widetilde{o}^i_{0:T}, \widetilde{g}^i_{0:T}, \widetilde{r}, \widetilde{o}'^i_{0:T}\rangle$. This experience can be stored in the teacher replay memory $\widetilde{\mathcal{D}}^i$ and sampled to update teacher policies (**Phase IV**).

Training Protocol

We utilize TD3 as our underlying RL algorithm for the worker, manager, and teacher policies. We extend TD3 to MARL settings and incorporate the multiagent actor-critic approach (Lowe et al. 2017) into (1) and (2) to reduce the non-stationary for manager and teacher policies. Note that the manager critics and the teacher critics use experiences sampled from different replay memories to update their parameters: manager critics sample from $\boldsymbol{D}$ and teacher critics sample from $\widetilde{\boldsymbol{D}}$. Pseudocode is presented in Algorithm 1.

[4] $\widetilde{\boldsymbol{g}}_{0:T} = \langle \widetilde{\boldsymbol{g}}_0, \widetilde{\boldsymbol{g}}_h, \widetilde{\boldsymbol{g}}_{2h}, \ldots, \widetilde{\boldsymbol{g}}_T\rangle$, where h is the manager horizon and T is the maximum episode time step.

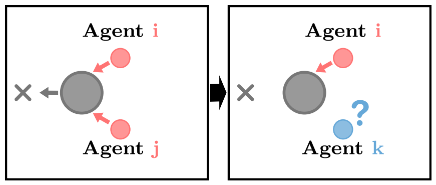


Figure 3: Visualization of the cooperative one box domain. The objective is to push the box O to the target $\times$. Agents i and j have prior knowledge about how to attain the objective. After learning to coordinate, agent i is teamed up with an agent k that has no knowledge about the domain. Agent i teaches agent k to accelerate team-wide learning progress.

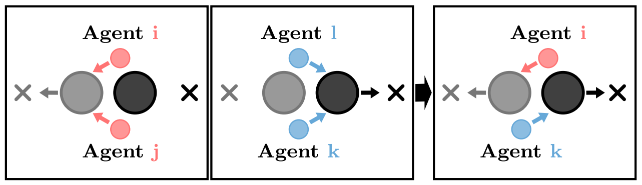


Figure 4: Visualization of the cooperative two box domain. The goal is to move box1 (the left box) to the left target and move box2 (the right box) to the right target. Agent pairs i–j and $k - l$ have prior knowledge about how to push box1 and box2 to corresponding targets, respectively. Then, agents i and k are teamed up. They teach one another to accelerate team-wide learning progress.

Evaluation

In this section, we demonstrate the performance of our proposed learning to teach framework, HMAT on a sequence of increasingly challenging domains. Agents in these domains have a heterogeneous knowledge and benefit from the peer-to-peer knowledge transfer via teaching. Through empirical evaluations, we show accelerated team-wide learning progress with HMAT.

Evaluation Domain

Our domains are based on the OpenAI's multiagent particle environment, which supports continuous observation and action space with basic simulated physics. We modify the environment and propose new domains, "cooperative one box push" and a "cooperative two box push", which exhibit hierarchical structure tasks and require coordinating agents.

Cooperative One Box Push Domain: The domain consists of one round box and two agents (Figure 3). The objective is to move the box to the target on the left side as soon as possible. The box can be moved if and only if two agents act on it together. This unique property requires the two agents to coordinate. Note that the box has a round shape. Therefore, agents are required to coordinate properly to move the box straightly to the target. If one agent pushes harder than the other agent, the box will follow a curved path taking a longer time to reach to the target. Each agent can observe its position, its speed, the position of the box, the position of the target, and the other agent's position. Each agent outputs two forces between -1 and 1 and moves in the environment. The domain returns a team reward: $-||\text{loc(Target)} - \text{loc(Box)}||_2^2$. The box and target reset at the same initial location but agents reset at random locations.

Cooperative Two Box Push Domain: This domain is similar to the one box domain but with increased complexity. There are two round boxes in the domain Figure 4. The objective is to move the left box (box1) to the left target (target1) and the right box (box2) to the right target (target2). The boxes still require two agents to act together. Additionally, box1 and box2 have different mass: box2 is three times heavier than box1. Each agent can observe its position, its speed, the positions of the boxes, the positions of the targets, and the other agent's position. The domain returns a team reward: $-\{||\text{loc(Target1)} - \text{loc(Box1)}||_2^2 + ||\text{loc(Target2)} - \text{loc(Box2)}||_2^2\}$. The boxes and targets reset at the same initial locations but agents reset at random locations.

Heterogeneous Knowledge and Teach Task

For each domain, we provide each agent with different prior knowledge, ensure a heterogeneous knowledge between agents, and motivate interesting teaching scenarios.

Cooperative One Box Push Domain: Consider Figure 3. Agent i and j have prior knowledge about how to move the box to the target. Then, agents i and k are teamed up, and agent k has no knowledge about the domain. Agent i, which understands how to move the box, should teach agent k. By giving desirable advice, it will improve k's learning progress and results in moving the box together quickly.

Cooperative Two Box Push Domain: Consider Figure 4. Agents i and j have prior knowledge about how to move box1 to target1. Agents k and l understand how to move box2 to target2. Note that the former team and the latter team have two different skills as the skills 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. In this scenario, agent i should transfer its knowledge about moving box1 to agent k. At the same time, agent k should teach agent i how to move box2. Thus, there is knowledge transfer both ways.

Implementation and Algorithm Details

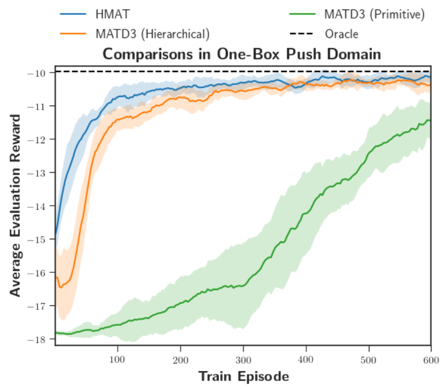
Each policy's actor and critic are two-layer feedforward neural networks, consisting of rectified linear unit (ReLU) activations and 400 nodes per layer. A final layer of tanh activation is used at the output of the actor policy. Since we focus on teaching at the manager level, not at the worker level, we pre-train and fix the worker policies by giving randomly generated sub-goals, similar to (Kulkarni et al. 2016).

Baselines

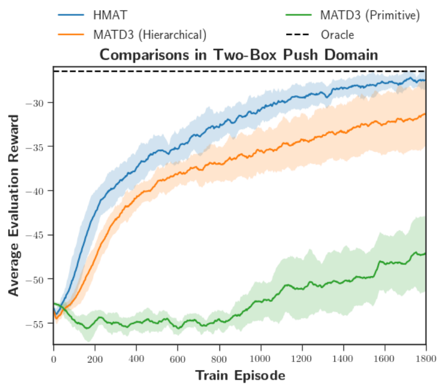
We compare our framework to the following baselines, which don't include teaching:

MATD3 (Primitive): TD3 is extended to MARL setting and the multi-agent actor-critic of (Lowe et al. 2017) is added to reduce the non-stationary.

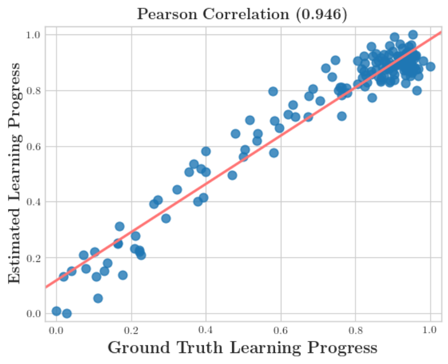
MATD3 (Hierarchical): The MATD3 (Primitive) baseline is extended with HRL with the two-hierarchical structure, which consists of manager policies and worker policies. The manager policies' actors are decentralized, but



(a)



(b)



(c)

Figure 5: **(a)** Task-level learning progress in the one-box push domain. Agents in HMAT show accelerated team-wide learning progress by agent i teaching agent k. **(b)** Task-level learning progress in the two-box push domain. Agents in HMAT show accelerated team-wide learning progress by teaching one another. **(c)** Ground-truth learning progress vs estimated learning progress. Our teacher reward function well estimates the true learning progress with a high correlation.

their critics are centralized to reduce the non-stationary. The worker policies are pre-trained.

Results

The task-level learning progress on both domains are summarized in Figure 5a and Figure 5b. The plots show a mean and a standard deviation for 10 sessions. The oracle in the figures refers to a maximum reward achieved by training MATD3 (Hierarchical) baseline until it converges. There are two consistent observations between the two results. One observation is that agents in HMAT accelerate in learning progress. This empirically demonstrates the strength of transferring heterogeneous knowledge between agents via teaching. The other observation is that HRL-based methods outperform the primitive method. This could have resulted from the fact that the domains pose delayed reward assignment. The one-box domain, for instance, returns the same reward until the box is moved by two agents. Therefore, the extended temporal abstraction provided by HRL would have helped in learning.

We also have attempted to compare against LeCTR's Value Estimation Gain (VEG) reward, $\mathbb{1}(\hat{V}(s) > \tau)$, where a teacher receives reward $+1$ if a student's value estimate is above a threshold τ and 0 otherwise. However, our preliminary experiments did not show a significant task-level learning progress using this reward. We conjecture that this is due to the VEG being a binary teacher reward function, where teachers may have to wait a long time before they receive positive rewards and teacher policies may diverge as a result, especially if a task is difficult. Thus, a different choice of teacher reward function might be more appropriate for LeCTR to operate in our domains.

Analysis of Teacher Reward Function

As noted earlier, developing ground-truth learning progress of task-level policies often requires an expert policy or is computationally undesirable (Graves et al. 2017). Thus, existing learning to teach frameworks and our framework use an estimation of the learning progress as a teacher reward. However, it is important to understand how close the estimation is to the true learning progress. The goal of teacher policies is to maximize the cumulative teacher reward, so a wrong teacher reward that doesn't estimate closely would result in learning undesirable teachers. In this section, we aim to measure the differences between the true and estimated learning progress and analyze our teacher reward function.

In imitation learning, with an assumption of a given expert, the true learning progress is measured by the distance between an action by a learning agent and an optimal action by an expert (Ross and Bagnell 2014; Daswani, Sunehag, and Hutter 2015). Similarly, we pre-train expert policies (MATD3 (Hierarchical)) and measure the true learning progress by the action differences. The comparison between the true and the estimated learning progress using our teacher reward function is shown in Figure 5c. The Pearson correlation is 0.946, which empirically shows that our teacher reward function well estimates the true signal.

Conclusion

We introduce Hierarchical Multiagent Teaching (HMAT), a learning to teach framework, which utilizes hierarchical reinforcement learning and deep neural network representation, to transfer heterogeneous knowledge in cooperative MARL. We show agents in HMAT accelerate learning progress in challenging domains. Future works include expanding the framework to support the peer-to-peer teaching with more than two agents involved.

Acknowledgements

This work was supported by IBM (as part of the MIT-IBM Watson AI Lab initiative) and AWS Machine Learning Research Awards program.

References

Bacon, P.; Harb, J.; and Precup, D. 2017. The option-critic architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 1726–1734.

da Silva, F. L.; Glatt, R.; and Costa, A. H. R. 2017. Simultaneously learning and advising in multiagent reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, 1100–1108.

Daswani, M.; Sunehag, P.; and Hutter, M. 2015. Reinforcement learning with value advice. In Phung, D., and Li, H., eds., *Proceedings of the Sixth Asian Conference on Machine Learning*, volume 39 of *Proceedings of Machine Learning Research*, 299–314. Nha Trang City, Vietnam: PMLR.

Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; and Abbeel, P. 2016. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, 1329–1338. JMLR.org.

Fachantidis, A.; Taylor, M. E.; and Vlahavas, I. P. 2017. Learning to teach reinforcement learning agents. *CoRR* abs/1707.09079.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2017. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926.*

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 1587–1596.

Garant, D.; da Silva, B. C.; and Lesser, V. R. 2015. Accelerating multi-agent reinforcement learning with dynamic co-learning.

Graves, A.; Bellemare, M. G.; Menick, J.; Munos, R.; and Kavukcuoglu, K. 2017. Automated curriculum learning for neural networks. *CoRR* abs/1704.03003.

Kulkarni, T. D.; Narasimhan, K. R.; Saeedi, A.; and Tenenbaum, J. B. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, 3682–3690. USA: Curran Associates Inc.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *International Conference on Representation Learning (ICRL).*

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, 157–163. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Liu, R., and Zou, J. 2017. The effects of memory replay in reinforcement learning. *CoRR* abs/1710.06574.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, O. P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 6382–6393.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015a. Human-level control through deep reinforcement learning. *Nature* 529–533.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015b. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Nachum, O.; Gu, S.; Lee, H.; and Levine, S. 2018. Data-efficient hierarchical reinforcement learning. *CoRR* abs/1805.08296.

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

Omidshafiei, S.; Kim, D.; Liu, M.; Tesauro, G.; Riemer, M.; Amato, C.; Campbell, M.; and How, J. P. 2018. Learning to teach in cooperative multiagent reinforcement learning. *CoRR* abs/1805.07830.

Rockoff, J. E., and Speroni, C. 2011. Subjective and objective evaluations of teacher effectiveness: Evidence from new york city. *Labour Economics* 18(5):687 – 696.

Ross, S., and Bagnell, J. A. 2014. Reinforcement and imitation learning via interactive no-regret learning. *CoRR* abs/1406.5979.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 387–395.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to Reinforcement Learning.* Cambridge, MA, USA: MIT Press, 1st edition.

Taylor, A., and Galvan-Lopez, E. 2013. Transfer learning in multi-agent systems through parallel transfer.

Taylor, M. E.; Carboni, N.; Fachantidis, A.; Vlahavas, I.; and Torrey, L. 2014. Reinforcement learning agents providing advice in complex video games. *Connection Science* 26(1):45–63.

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K. 2017. FeUdal networks for hierarchical reinforcement learning. In Precup, D., and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 3540–3549. International Convention Centre, Sydney, Australia: PMLR.

Vrancx, P.; De Hauwere, Y.-M.; and Nowe, A. 2011. Transfer learning for multi-agent coordination.

Xu, T.; Liu, Q.; Zhao, L.; and Peng, J. 2018. Learning to explore with meta-policy gradient. *CoRR* abs/1803.05044.