Explainable Action Advising for Multi-Agent Reinforcement Learning

Yue Guo[1], Joseph Campbell[1], Simon Stepputtis[1], Ruiyu Li[1],
Dana Hughes[1], Fei Fang[1], Katia Sycara[1], *Fellow, IEEE*

Abstract— Action advising is a knowledge transfer technique for reinforcement learning based on the teacher-student paradigm. An expert teacher provides advice to a student during training in order to improve the student's sample efficiency and policy performance. Such advice is commonly given in the form of state-action pairs. However, it makes it difficult for the student to reason with and apply to novel states. We introduce Explainable Action Advising, in which the teacher provides action advice as well as associated explanations indicating why the action was chosen. This allows the student to self-reflect on what it has learned, enabling advice generalization and leading to improved sample efficiency and learning performance – even in environments where the teacher is sub-optimal. We empirically show that our framework is effective in both single-agent and multi-agent scenarios, yielding improved policy returns and convergence rates when compared to state-of-the-art methods.

I. INTRODUCTION

As robots are deployed in real-world settings, opportunities may arise for agents to impart their knowledge to other agents. When considering the space of all possible observations that an agent might encounter in the wild, it is unlikely a single agent can learn to react to all possible situations. In such cases, it is beneficial for robots with a diversity of experiences to share this knowledge with others so they know how to act. Recent works have explored teacher-student methods [1], [2], [3], [4] in which a teacher agent transfers its knowledge to a student agent with the goal of accelerating the student's learning. Such techniques can be applied in robotics, where experienced agents can transfer knowledge on how to handle as-of-yet unseen situations to inexperienced students. Teacher-student methods are applicable in many scenarios, such as when a) directly copying the teacher's knowledge is infeasible, e.g., different policy models or a human student/teacher; b) the teacher is non-optimal and we wish to only transfer a subset of its knowledge, e.g., transfer learning to novel environments; or c) we want to pursue incremental learning as in human curricula. A well-studied approach is that of *action advising* in which a teacher selectively provides advice in the form of state-action pairs to the student, representing a suggested course of action. Reinforcement learning (RL) settings in particular benefit from advising, given the high sample complexity associated with many algorithms [5] which leads to time-consuming training. Unlike related techniques such

[1]Carnegie Mellon University, USA {yueguo, feif, katia}
@cs.cmu.edu, {jcampbell, stepputtis }@cmu.edu,
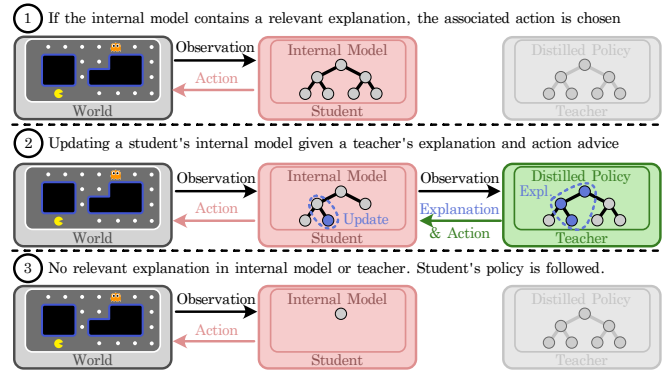{ruiyul, danahugh}@andrew.cmu.edu

Fig. 1: Overview of EAA. As the student makes observations during training, one of three cases may occur. 1) The student has previously received advice for a similar state and is able to identify it through an explanation in its internal model, then follow the associated action. 2) The student has not previously received advice, but is now given advice by the teacher and stores the explanation in its internal model, then follows the advised action. 3) The student has not previously received advice and the teacher is not currently providing advice; the student follows its own policy. In cases where the teacher is trained in a different environment than the student, the student may choose to reject unhelpful advice.

as inverse reinforcement learning [6], intention inference [7], and offline reinforcement learning [8] which assume an offline dataset of expert actions, action advising techniques leverage an intelligent teacher capable of deciding when and how to issue advice.

Despite some notable successes in simple environments [5], action advising methods often perform poorly in complex environments with large state spaces. This is especially true for multi-agent reinforcement learning (MARL) where environments exhibit complex team dynamics, non-stationarity, and high-dimensional state spaces. Memory-based approaches address these difficulties to a certain extent by reusing previous advice [4]; however, they are inherently limited as advice can only be re-used when the exact same state is encountered again. To overcome this, we propose a novel action advising algorithm in which the teacher agent issues action advice as well as an explanation for *why* that advice was issued. We show that incrementally incorporating these explanations into an internal model enables a form of *learning through reflection* [9], where an agent is able to reason about why a specific action was recommended and when it would apply in the future. This greatly improves advice generalization and improved sample efficiency. Reflection also allows an agent to identify unhelpful advice under certain conditions, allowing advice transfer to novel environments.

In our proposed method, Explainable Action Advising

(EAA), we assume access to a teacher's "black box" policy, i.e., neural network, from which we can sample demonstration trajectories. These trajectories are used to distill the policy into a decision tree in which each leaf represents an action, and the corresponding path from root to leaf represents the explanation for why an action was taken. When the teacher issues advice, both the decision path (explanation) as well as the leaf (action) are given to the student, who incrementally re-assembles the decision paths together in an attempt to reconstruct the teacher's decision tree in its internal model. This internal model serves two purposes: 1) it acts as a memory and allows a student to apply prior advice to same or similar future states, and 2) it allows a student to identify unhelpful advice and selectively ignore it, which is useful for a sub-optimal teacher and/or novel environments. This is analogous to the self-explanations [10] employed by human students in order to improve learning effectiveness. Importantly, EAA is agnostic to the student's underlying reinforcement learning algorithm – intuitively the teacher is guiding policy exploration.

Our work makes three main contributions, we: a) introduce an RL algorithm-agnostic action advising method which uses explanations to improve advice generalization; b) utilize explanations to make use of sub-optimal advice for use in transfer learning; c) empirically show that our approach[1]. improves policy returns and sample efficiency in single-agent and multi-agent environments, when compared to multiple state-of-the-art baselines.

II. Related Work

Action Advising (AA): Action advising in reinforcement learning is an approach for transferring knowledge from a teacher to a student by giving advice in the form of state-action pairs [11]. Important early work introduced the concept of teaching on a budget, and described heuristic methods for when to give advice in single-agent missions [2]. Since then, additional work has examined different forms of teacher-student relationships defined by who initiates the advice, e.g. (a) student-initiated [5], (b) teacher-initiated [12], and (c) jointly-initiated [13]. Most of the work focuses on addressing *when* and *which* action to give, among some of which advising is formed as a reinforcement task that the teacher agent takes the student's performance as its reward [12], [3], [14]. Other works instead build off of previously introduced heuristic methods for deep reinforcement learning [15], [16].

Action advising in MARL: In many MARL-based action advising approaches, each agent is simultaneously both a student and a teacher and they take alternate teaching roles [16], [3], [17]. The agents are not required to possess expert knowledge a priori, and agents are capable of probabilistically rejecting teammates' advice [16]. To improve efficiency, option learning is utilized to tell which agent's policy is the best to learn [18], while distillation and value-matching are used to combine knowledge from

homogeneous agents to decompose the multi-agent task [19]. Teaming in partially observable environments in decentralized settings [20] has been explored, in which heuristic teaching methods are employed to send advice to a decentralized team. Conversely, centralized settings have also received attention [21], where a single centralized teacher observes the entire space and communicates advice to student agents. None of the above works consider explainability or transparency with respect to the teacher's advice. While sub-optimal teachers are considered [22], there is no mechanism for a student to identify the reasoning of sub-optimality unlike in our approach.

Reusing Action Advice: Recent work has explored the reuse of advice in the form of state-action pairs so as to maximize communication efficiency. In one approach [4], state-action pairs are directly stored in memory and later selectively re-used. An imitation learning module has been proposed to imitate the teacher's advice, such that advice can be generated after the budget is exhausted [15]. Prior work has also investigated a teacher capable of providing a qualitative assessment of a given state along with a piece of advice, such that students are aware of how favorable a given state is [23]. However, these approaches all struggle to generalize action advice to novel states and environments.

III. Background

Multi-Agent Reinforcement Learning: We consider the problem of transferring knowledge from a teacher to a student in a multi-agent RL setting. The Markov Decision Process (MDP) for this is a tuple of $(\mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T})$, where $\mathcal{N} = \{0...n\}$ is a set of agents, $\mathcal{S}$ is a set of states, $\mathcal{A} = \Pi_i \mathcal{A}_i$ is a set of joint actions , $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ is a reward function, $\mathcal{T}(s, a_0, a_1...a_n, s') = P(s'|s, a_0, a_1...a_n)$ is a transition function. There is a set of policies $\pi_i(s, a_i) = P(a_i|s)$ for each agent i, where the single-agent setting is a special case. The goal is to learn a policy which maximizes the discounted cumulative reward.

Decision Trees and Policy Distillation: A precondition of our method is the distillation of a black-box teacher agent policy into an interpretable representation, i.e., a decision tree which is commonly employed as an interpretable proxy model [24]. Given a pretrained neural network policy, we extract it into an equivalent decision tree model utilizing the distillation method from the *Verifiability via Iterative Policy ExtRaction* (VIPER) [25] framework. VIPER extracts a suitable decision tree policy $\hat{\pi}^*$ from a policy network π^* through an iterative process of sampling trajectories and training a decision tree with standard algorithms, in a DAGGER-based [26] setup. Over the course of N iterations, VIPER samples trajectories of state-action pairs $(s, \pi^*(s))$ into a running data set D_t. This data set is re-sampled according to $(s,a) \sim P((s,a)) \propto \hat{l}(s)\mathbb{1}[(s,a) \in D]$ where $\mathbb{1}()$ is the indicator function and the loss is computed by:

$$\hat{l}_t(s) = V_t^{(\pi^*)}(s) - min_{a \in A}Q_t^{(\pi^*)}(s,a)$$

After re-sampling to get the current data set D_t, a single decision tree policy is trained. After N iterations, the best
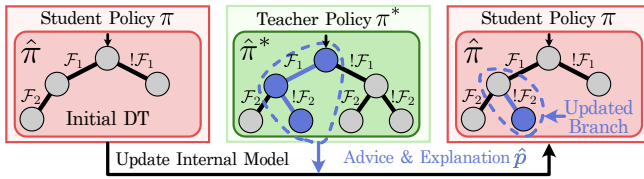
Fig. 2: The decision tree reconstruction process that is performed when updating a student's internal model $\hat{\pi}$. The student receives an explanation decision path $\mathcal{F}_1 \wedge !\mathcal{F}_2$ from the teacher's distilled policy $\hat{\pi}^*$, finds the last common node $\mathcal{F}_1$ currently in its model, and adds all subsequent missing nodes – in this case only $!\mathcal{F}_2$.

tree policy $\hat{\pi}^*$ out of all candidate policies is selected.

IV. METHODOLOGY

We introduce Explainable Action Advising (see Fig. 1), in which a teacher agent provides action advice as well as an explanation to a student agent.

A. Explainable Action Advising

Algorithm 1 Explainable Action Advising

Input: Distilled teacher policies $\hat{\pi}_1^*, \dots \hat{\pi}_n^*$ for n-agent mission, heuristic function $h(\cdot)$
Parameter: Advice budget b, advice decay rate γ

1: Internal model $\hat{\pi}_i = \emptyset$, student $\pi_i = \emptyset \ \forall i = 1, \dots, n$
2: **for** iter $j = 1, 2, \dots$ **do**
3:    **for** time step $t = 1, \dots, T$ **do**
4:       **for** agent $i = 1, \dots, n$ **do**
5:          **if** $\gamma^j < \alpha \sim \mathcal{U}[0, 1]$ and $s_i^t \in \hat{\pi}_i$ **then**
6:             take action $a_i^t = \hat{\pi}_i(s_i^t)$
7:          **else if** $b > 0$ and $h(s_i^t)$ **then**
8:             give advice $\hat{a}_i^t$ and expl. $\hat{p}_i^t = \texttt{path}(\hat{\pi}_i^*, s_i^t)$
9:             update internal model $\hat{\pi}_i = \texttt{store}(\hat{\pi}_i, \hat{p}_i^t)$
10:            take action $\hat{a}_i^t$
11:          **else**
12:            take action $a_i^t = \pi_i(s_i^t)$
13:          **end if**
14:       **end for**
15:    **end for**
16:    Update π_i via policy optimization for $i = 1, \dots, n$
17: **end for**

An outline of the EAA algorithm is shown in Algorithm 1. The distilled teacher policies $\hat{\pi}_1^*, \hat{\pi}_2^* \dots \hat{\pi}_n^*$ are provided as input to EAA – one for each student agent – along with a heuristic function $h(\cdot)$ which determines when advice should be issued to an agent. We define two hyperparameters: an advising budget b and a decay parameter γ. The budget b is used to limit the amount of advice that is issued during the student's training, which is useful for limiting the communication between teacher and student, and motivated by preventing cognitive overload in potential human students [2]. The decay rate γ determines the likelihood with which an agent will re-use advice that has been stored in its internal model $\hat{\pi}_i$, giving the student freedom to execute its own policy

after it has sufficiently learned from the teacher's advice and potentially exceed the teacher's own performance.

Our algorithm operates over multiple iterations, during which we roll out a policy and sample T timesteps from the environment. For each time step t, one of three things happens: a) the student will query its internal model for an action, b) the teacher will issue advice and an explanation which the student will incorporate into its internal model and then follow, or c) the student will sample an action from its own current policy. In the first case on Line 6, if the student has previously received and stored advice applicable to its current state, it follows the corresponding action with a probability determined by the decay rate γ. Intuitively, the agent should become less reliant on the teacher's prior advice over time and increasingly follow its own actions.

In the second case on Line 8, if the student has not previously received relevant advice the teacher agent determines whether advice should be issued to its student, which is controlled by the heuristic function $h(\cdot)$, the remaining budget b, and the uncertainty associated with the advised action such that only sufficiently confident predictions are used as advice. If this occurs, the teacher samples a leaf node action $\hat{a}_i^t$ and its decision path explanation $\hat{p}_i^t$ from its distilled decision tree and sends them to the student. The student takes this explanation, incorporates it into its own internal model $\hat{\pi}_i$, and follows the advised action.

Lastly, in the third case on Line 12 the student simply follows its own current policy π_i, which is itself updated on each iteration with the data collected from the rollouts. Additionally, there is an optional check in which a teacher agent may only issue advice if: a) the uncertainty of executing the action is sufficiently small, and b) the advice from its internal model matches the action from its expert policy – useful if the distilled tree is overly noisy. If the action uncertainty is too high, the teacher may choose to issue only action advice and withold the explanation from the student. This allows the student to leverage the teacher's learned action probabilities without committing them to memory.

B. Decision Tree Reconstruction

The reconstruction process performed in the `store` function, shown in Fig. 2, consists of merging the current decision tree $\hat{\pi}_i$ and the decision path explanation, $\hat{p}_i^t$ which reduces to merging two trees. Following standard decision tree architecture, we assume that the tree consists of a set of K nodes $u_0, \dots, u_K$, where each node contains information regarding its child nodes and branching condition. By transferring the teacher's tree in a piece-wise fashion, we transmit only useful advice which aids in the generalization process – uncertain actions are ignored – while minimizing the amount of information transmitted. The size of the student's tree is conditional on the budget and complexity of the policy.

C. Generalization and Transfer Learning

Using a decision tree representation for the internal model allows the student to generalize the advice to novel states which differ from that in which it was issued. We do this by

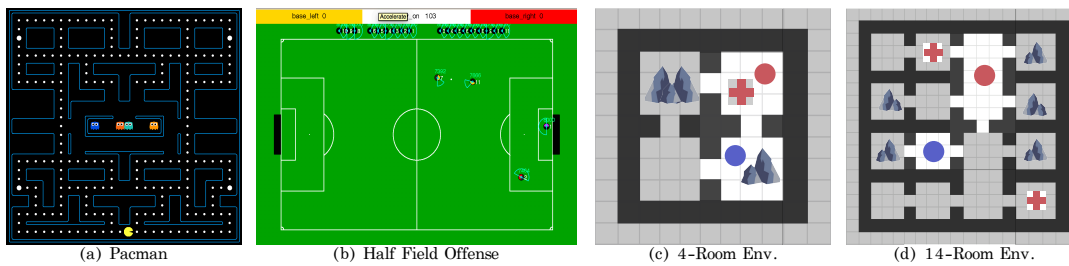| (a) Pacman | (b) Half Field Offense | (c) 4-Room Env. | (d) 14-Room Env. |

Fig. 3: Environment Screenshots. The 4 and 14 room environment have a medic (red circle), engineer (blue circle) and victims (red plus).

treating the decision tree as a classification model, in which the state s is the input and the action a is the target, and extracting a generalizable classification tree during distillation. Formally, let the set of all possible observed features be $\mathcal{F}_S$ and the set of features modeled by our reconstructed decision tree be $\mathcal{F}_R$ where $\mathcal{F}_R \subseteq \mathcal{F}_S$. Then a given decision path corresponding to a feature vector $f \in \mathcal{F}_S$ will also generalize to any other feature vector $f' \in \mathcal{F}_S$ as long as the features in which they differ are not contained in $\mathcal{F}_R$. Such generalization leads to improved advice reuse, and as a result improved learning performance as is empirically shown in Sec. V-D.

Similarly, a decision tree representation also lends itself to transfer learning, by allowing a student to recognize situations in which advice is *not* applicable and as such should not be followed, e.g. when the teacher is trained in a different environment. Let us denote the set of features in the source environment as $\mathcal{F}_S$ and the set of features in the target environment as $\mathcal{F}_T$. We impose an additional constraint on the student such that it *rejects* advice if the decision path explanation contains features in the set $\mathcal{F}_S - \mathcal{F}_T$ or if the state vector contains features in $\mathcal{F}_T - \mathcal{F}_S$, and the student instead follows its own policy. Intuitively, we only want to follow advice if it is dependent on features found in both environments. We refer to this process as *learning through reflection*, and empirically demonstrate that this yields significantly improved policy transfer in Sec. V-D.

V. Experiments

We empirically evaluate EAA in three simulated environments: Pacman, Half Field Offense (HFO), and Urban Search and Rescue (USAR). In each of the environments we first train a teacher policy to convergence and use this policy to subsequently advise a student policy using EAA as well as several baseline algorithms. To evaluate EAA's transfer learning ability, we perform an additional experiment in the USAR environment where the teacher is trained in a variant where a feature is missing, and then advise a student in the environment with that feature. In total, we evaluate EAA against 11 benchmark methods: four action advising methods in Fig. 4, three memory-based action advising methods in Fig. 5 (a), one adaptive teacher action advising method in Fig. 5 (b), and three transfer learning techniques in Fig. 6.

A. Experimental Setup

Figure 3 depicts our three environments (Pacman, HFO, and USAR), with the USAR task implemented in a simpler 4-room and more difficult 14-room setup.

Pacman: a single-agent scenario with a discrete state space [27]. We use the standard single-agent environment [27] with the *OriginalClassic* layout (Fig. 3 (a)) and 4 aggressive ghosts. An additional reward penalty of -1 is received when an invalid action is predicted, in place of action masking. The student and teacher policies were trained via PPO [28] for $10,000$ episodes with an advice budget of $b = 10$. The teacher and distilled teacher achieved average rewards of 450.

HFO: a multi-agent scenario with a continuous state space where two homogeneous RoboCup agents play as a team [29], [30]. We use the standard 2v2 environment [29] (Fig. 3 (b)) and train teacher and student policies for the offense agents with SARSA [31] as in prior work [16] for $2,000$ episodes with an advice budget of 300. The teacher and distilled teacher achieved an average reward of 0.8.

USAR: a multi-agent scenario with a discrete state space and heterogeneous agents with different action spaces and hard coordination constraints [32], [33], [34]. We have two layouts: a simple 4-room layout (Fig. 3 (c)) and a more complex 14-room layout (Fig. 3 (d)). The environment consists of rubble which can be cleared and victims which must be healed, and a team of two agents: a medic who heals unblocked victims and an engineer who can clear rubble from blocked victims. The reward is team-level and gives $+10$ for each victim healed. Both layouts contain rubble in every room, while the 4-room layout contains 1 victim randomly distributed and the 14-room layout contains 2 victims. The teacher and student policies are trained using COMA [35] for $16,000$ episodes with an advice budget of $100,000$. The teacher and distilled teacher achieved an average reward of 10 in both layouts, indicating a sub-optimal teacher for the 14-room layout. This task necessitates cooperation between a medic and an engineer robot to save trapped victims – the engineer must remove rubble before the medic comes across and heals the victim, and is the most difficult environment.

B. Benchmarks

In all experiments, we define four variants of EAA utilizing the heuristic functions established in prior work [2] as $h(\cdot)$: EAA with early advising (EAA-E), alternative advising (EAA-A), importance advising (EAA-I), and mistake correcting advising (EAA-M). EAA-Median indicates the median performance of all EAA variants. The heuristics are the same as in the Action Advising baselines and are defined below.

Action Advising: We compare EAA to four heuristic-based Action Advising algorithms [2] in all environments:
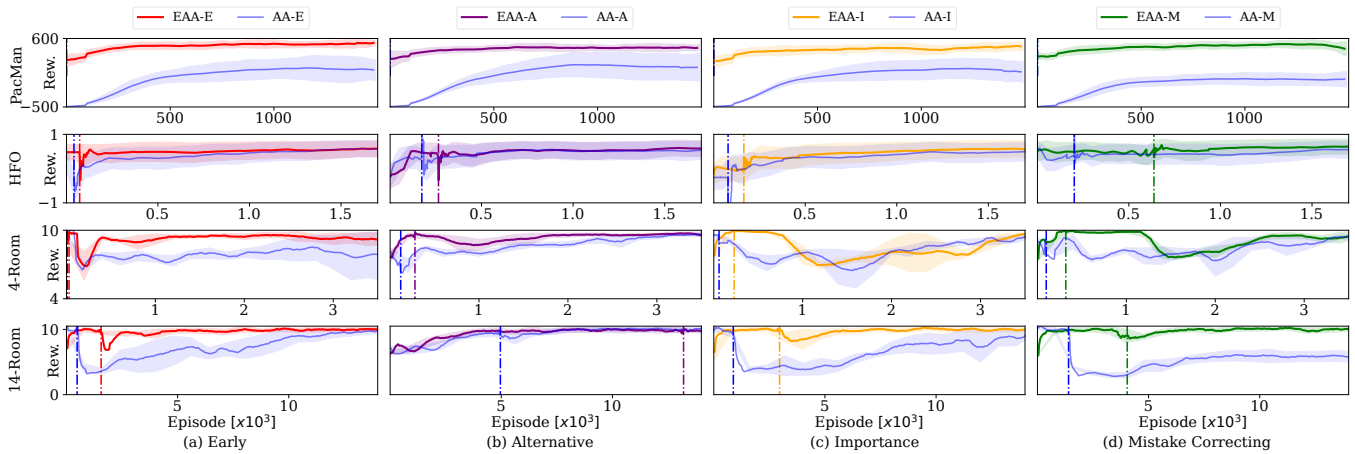
Fig. 4: Comparing Explainable Action Advising (Early: Red, Alternative: Purple, Importance: Yellow, Mistake Correcting: Green) with Action Advising (Blue) on the PacMan, HFO, and USAR (four and fourteen room) environments. Dash-Dotted lines indicate when advising budget is exhausted. Results are averaged over 5 independent trials except for 14-room over 3.
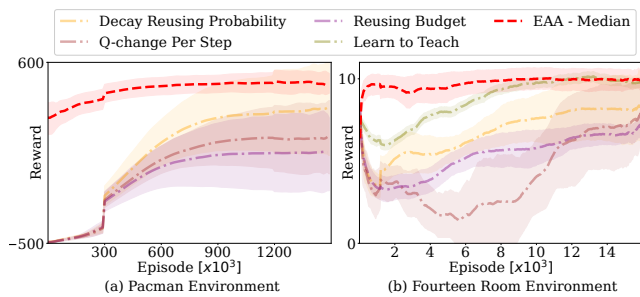

Fig. 5: Baseline of Pacman (left) and 14-room USAR comparing EAA to SOTA algorithms. Results are averaged over 3 random seeds except for Learn to Teach which uses 1 due to run-time.
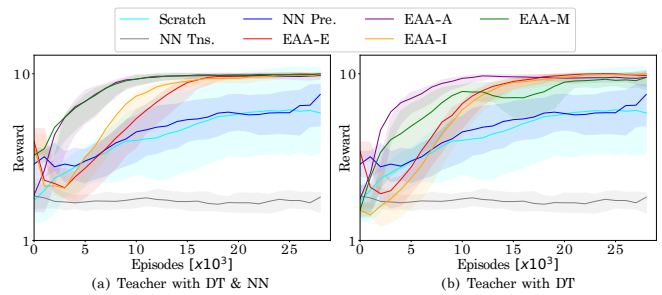

Fig. 6: Transfer learning performance for students training in the USAR environment with rubble by a teacher who was originally trained without rubble. The teacher only has access to the extracted decision tree where the teacher NN is not present in (b), compared to (a) where the teacher NN is present. Results are averaged over 5 independent trials.

1) *(AA-E) Early* : Continuously advise at each state.
2) *(AA-A) Alternative* : Advise with a fixed frequency.
3) *(AA-I) Importance* : Advise when the state importance $I(s) >$ threshold.
4) *(AA-M) Mistake Correcting* : Advise if $I(s) >$ threshold and $a^{student} \neq a^*$.

Memory-based Action Advising: For the discrete state environments, we compare EAA to three recently introduced [4] benchmark algorithms: Q-change Per Step, Decay Reusing Probability, and Reusing Budget. These works are conceptually similar to ours in that students have a memory and reuse teacher advice in the form of state-action pairs, so they serve as a strong state-of-the-art comparison.

Action Advising with Adaptive Teacher: In the USAR environment, we further compare EAA against an approach in which reinforcement learning is used to learn an optimal teacher [14], [3], and denote this as *Learn to Teach*. This algorithm learns over multiple sessions, within each of which the teacher learns to allocate its advice budget with the constraint that the student's policy must converge. All methods except for Learn to Teach are trained for 5 random seeds. Due to the computational demand of the inner reinforcement learning loop, Learn to Teach (100x slower) is only evaluated once for USAR.

Fine-tuning: In the transfer learning experiments, we compare EAA against the standard transfer learning technique

in which a policy is pre-trained in a source environment, and then fine-tuned in a target environment, denoted as NN Pre. We also include results for the pre-trained policy without fine-tuning (NN Tns.) and a policy baseline trained from scratch (Scratch).

C. Results: Advice Re-use

Figures 4 and 5 show the results of our EAA variants to the action advising baselines described above, and are intended to demonstrate improved policy return and sample efficiency owing to advise re-use from explanations.

Comparison to Action Advising: In Fig. 4, we compare our four EAA variants described in Sec. V-B to AA [2] in all environments. We observe that EAA converges to the teacher's reward significantly faster than the AA baselines and with lower reward variance, indicating more training stability. Of note is PacMan, where the lack of initial state randomization enables heavy advice re-use and offsets the small advice budget, allowing EAA to perform remarkably well. In HFO, EAA achieves only a slight improvement over AA; we conjecture that this is due to the simplicity of the task – it was designed for classical methods such as SARSA, which enables AA to perform well. Advise re-use and generalization
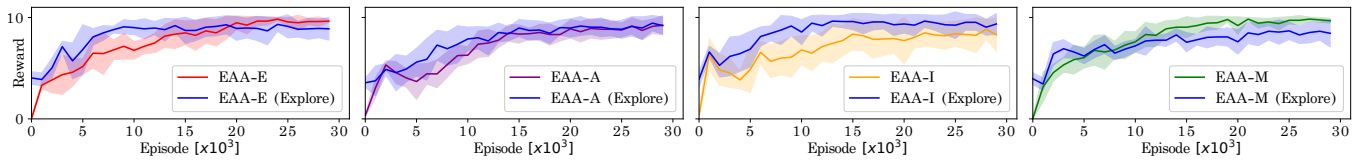
Fig. 7: A comparison of EAA with reflection, called EAA (Explore), to standard EAA. The student is pretrained in a source environment (4-room no rubble) and transferred to a target environment (4-room with rubble). Results are averaged over 5 independent trials.

is evident in the HFO and USAR environments where EAA results in later budget exhaustion (dashed lines, Fig. 4) across all heuristics. This is not evident in PacMan simply due to the extremely small advice budget; it is spent quickly at the start of training. All algorithms except for EAA-A (Fig. 4 (b)) also suffer from a large drop in reward when the teacher's budget is exhausted. While this is expected, it is interesting to observe that our EAA variants tend to recover quicker from this drop than AA.

Comparison to Memory-based Action Advising: In Fig. 5, we see that EAA again demonstrates both improved convergence rate and maximal policy return when compared to the memory-based action advising baselines, achieving optimal performance in both the Pacman and USAR 14 room environments. Q-change Per Step displays the worst performance because it assumes a strict increase in Q-values when the student receives advice, otherwise discarding it. However, this assumption is often violated, especially in USAR where an action such as removing rubble does not directly result in an increase in Q-values – only healing victims does. Decay Reusing Probability and Reusing Budget are able to achieve optimal performance, yet take significantly longer to converge than EAA.

Comparison to Action Advising with Adaptive Teacher: Figure 5 (b) shows the comparison the adaptive teacher action advising method, Learn to Teach, in the USAR 14-room layout. While Learn to Teach converges to the teacher's reward, it takes considerably longer to do so, requiring nearly 12,000 episodes compared to EAA's 6,000.

D. Results: Transfer Learning

Figures 6 and 7 show the results when self-reflection is used to selectively transfer advice from a teacher trained in a different environment than the student.

Comparison to Fine-tuning: Figure 6 shows EAA's performance when transferring advice from a source environment to a target environment, as compared to standard transfer learning approaches. The teacher is a trained expert in a USAR 4-room source environment *without* rubble, and is used to advise a student in a target environment *with* rubble, in which it is a non-expert due to the additional coordination requirement of removing it. As discussed in Sec. IV-C, the student rejects advice if the teacher's advised explanation contains no features related to rubble when encountering a room with rubble. Fig. 6 (a) EAA has access to both the distilled and original policy and does not store explanations in which the action has a low probability, as discussed in Sec. IV-A. This is the default behavior used for prior experiments in Sec. V-C, and represents the best-case transfer learning

scenario. In Fig. 6 (b), the EAA variants are trained by a teacher who only has access to its distilled decision tree and not its original policy. This represents a more difficult task, and establishes a baseline for the later case where there is no teacher and the student relies only on its internal model trained in the source environment (Fig. 7).

In Figure 6 all four variants of EAA significantly outperform the neural network baselines and converge to the optimal reward within 10,000 episodes. The transferred teacher policy (NN Tns.) serves as a baseline of the non-expert teacher behavior in the target environment without any additional training; this is the upper limit on the expected sub-optimal advice performance. Notably neither training from scratch (Scratch) nor pre-training on the source environment followed by fine-tuning (NN pre.) consistently converges to the optimal reward. Given that EAA performs nearly as well in Fig. 6 (b) as Fig. 6 (a), we conclude that maintaining the original black-box teacher policy is unnecessary.

Transfer Learning through Self-Reflection: Lastly, Fig. 7 shows the results when a student trained via EAA in a source environment continues training in a target environment, when no teacher – and therefore no advising – is present. We implement a version of EAA with reflection in which the student examines its own reconstructed decision tree and ignores actions learned from unhelpful advice and instead executes an exploratory action (EAA Explore). When compared to an ablated student without reflection, we find that reflection not only leads to improved rewards at the start of training, but also an improved convergence rate for EAA-E, EAA-A, and EAA-I variants. These results show that not only does EAA improve policy transfer to novel environments in the presence of a teacher, but it even yields benefits when the student operates in novel environments *by itself*.

VI. Conclusion

We propose EAA, a novel action advising algorithm in which a teacher agent issues explanations in addition to action advice to student agents. Our results show that EAA improves both policy reward as well as convergence speed compared to eight state-of-the-art benchmark algorithms, while also improving advice generalization and transfer to novel environments through reflection. We demonstrate that our method is especially effective in budget-constrained advising, as each piece of advice now conveys more information and helps reduce communication frequency with respect to advice in the case of artificial agents. We designed EAA with human students in mind, and intend to investigate the degree to which our explanations are intelligible to humans in the future.

References

[1] F. L. Da Silva, G. Warnell, A. H. R. Costa, and P. Stone, "Agents teaching agents: a survey on inter-agent transfer learning," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 1, pp. 1–17, 2020.

[2] L. Torrey and M. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1053–1060.

[3] S. Omidshafiei, D.-K. Kim, M. Liu, G. Tesauro, M. Riemer, C. Amato, M. Campbell, and J. P. How, "Learning to teach in cooperative multiagent reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6128–6136.

[4] C. Zhu, Y. Cai, H.-f. Leung, and S. Hu, "Learning by reusing previous advice in teacher-student paradigm," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1674–1682.

[5] F. L. Da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-aware action advising for deep reinforcement learning agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5792–5799.

[6] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[7] J. Grover, Y. Lyu, W. Luo, C. Liu, J. Dolan, and K. Sycara, "Semantically-aware pedestrian intent prediction with barrier functions and mixed-integer quadratic programming," *IFAC-PapersOnLine*, vol. 55, no. 41, pp. 167–174, 2022.

[8] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.

[9] R. Helyer, "Learning through reflection: the critical role of reflection in work-based learning (wbl)," *Journal of Work-Applied Management*, 2015.

[10] M. T. Chi, N. De Leeuw, M.-H. Chiu, and C. LaVancher, "Eliciting self-explanations improves understanding," *Cognitive science*, vol. 18, no. 3, pp. 439–477, 1994.

[11] F. L. Da Silva and A. H. R. Costa, "A survey on transfer learning for multiagent reinforcement learning systems," *Journal of Artificial Intelligence Research*, vol. 64, pp. 645–703, 2019.

[12] A. Fachantidis, M. E. Taylor, and I. Vlahavas, "Learning to teach reinforcement learning agents," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 21–42, 2019.

[13] O. Amir, E. Kamar, A. Kolobov, and B. Grosz, "Interactive teaching strategies for agent training," 2016.

[14] M. Zimmer, P. Viappiani, and P. Weng, "Teacher-student framework: a reinforcement learning approach," in *AAMAS Workshop Autonomous Robots and Multirobot Systems*, 2014.

[15] E. Ilhan, J. Gow, and D. Perez-Liebana, "Action advising with advice imitation in deep reinforcement learning," *Proceedings of the AAMAS Conference*, pp. 629–637, 2021.

[16] F. L. Da Silva, R. Glatt, and A. H. R. Costa, "Simultaneously learning and advising in multiagent reinforcement learning," in *Proceedings of the 16th conference on autonomous agents and multiagent systems*, 2017, pp. 1100–1108.

[17] D.-K. Kim, M. Liu, S. Omidshafiei, S. Lopez-Cot, M. Riemer, G. Habibi, G. Tesauro, S. Mourad, M. Campbell, and J. P. How, "Learning hierarchical teaching policies for cooperative agents," *arXiv preprint arXiv:1903.03216*, 2019.

[18] T. Yang, W. Wang, H. Tang, J. Hao, Z. Meng, H. Mao, D. Li, W. Liu, Y. Chen, Y. Hu *et al.*, "An efficient transfer learning framework for multiagent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 037–17 048, 2021.

[19] S. Wadhwania, D.-K. Kim, S. Omidshafiei, and J. P. How, "Policy distillation and value matching in multiagent reinforcement learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 8193–8200.

[20] E. Ilhan, J. Gow, and D. Perez-Liebana, "Teaching on a budget in multi-agent deep reinforcement learning," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–8.

[21] N. Gupta, G. Srinivasaraghavan, S. K. Mohalik, and M. E. Taylor, "Hammer: Multi-level coordination of reinforcement learning agents via learned messaging," *arXiv preprint arXiv:2102.00824*, 2021.

[22] S. G. Subramanian, M. E. Taylor, K. Larson, and M. Crowley, "Multi-agent advisor Q-learning," *Journal of Artificial Intelligence Research*, vol. 74, pp. 1–74, 2022.

[23] D. Anand, V. Gupta, P. Paruchuri, and B. Ravindran, "An enhanced advising model in teacher-student framework using state categorization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 6653–6660.

[24] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 2018, pp. 80–89.

[25] O. Bastani, Y. Pu, and A. Solar-Lezama, "Verifiable reinforcement learning via policy extraction," *arXiv preprint arXiv:1805.08328*, 2018.

[26] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[27] John DeNero, Dan Klein, Ketrina Yim, and Pieter Abbeel, "UC Berkeley CS188 intro to AI," University of California, Berkeley.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[29] S. Kalyanakrishnan, Y. Liu, and P. Stone, "Half field offense in robocup soccer: A multiagent reinforcement learning case study," in *Robot soccer world cup*. Springer, 2006, pp. 72–85.

[30] M. Hausknecht, P. Mupparaju, S. Subramanian, S. Kalyanakrishnan, and P. Stone, "Half field offense: An environment for multiagent learning and ad hoc teamwork," in *AAMAS Adaptive Learning Agents (ALA) Workshop*, vol. 3. sn, 2016.

[31] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[32] M. Lewis, K. Sycara, and I. Nourbakhsh, "Developing a testbed for studying human-robot interaction in urban search and rescue," in *Human-Centered Computing*. CRC Press, 2019, pp. 270–274.

[33] Y. Guo, R. Jena, D. Hughes, M. Lewis, and K. Sycara, "Transfer learning for human navigation and triage strategies prediction in a simulated urban search and rescue task," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 784–791.

[34] J. T. Freeman, L. Huang, M. Woods, and S. J. Cauffman, "Evaluating artificial social intelligence in an urban search and rescue task environment," in *AAAI 2021 Fall Symposium, 4-6 Nov 2021*. AAAI, 2021.

[35] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.