Coordination Confidence based Human-Multi-Agent Transfer Learning for Collaborative Teams

Bikramjit Banerjee
University of Southern Mississippi
Bikramjit.Banerjee@usm.edu

Matthew E. Taylor
Washington State University
taylorm@eecs.wsu.edu

ABSTRACT

Among an array of techniques proposed to speed-up reinforcement learning (RL), learning from human demonstration has a proven record of success. A related technique, called Human Agent Transfer (HAT), and its confidence-based derivatives have been successfully applied to single agent RL. This paper investigates their application to collaborative multi-agent RL problems. We show that a first-cut extension may leave room for improvement in some domains, and propose a new algorithm called *coordination confidence* (CC). CC analyzes the difference in perspectives between a human demonstrator (global view) and the learning agents (local view), and informs the agents' action choices when the difference is critical and simply following the human demonstration can lead to miscoordination. Experiments in two domains—one where the difference in perspectives is critical, and one where it is not—investigate the performance of CC in comparison with relevant baselines.

KEYWORDS

Multi-agent Learning; Reinforcement Learning; Learning from Demonstration

1 INTRODUCTION

Reinforcement learning (RL) [10] has seen many successful applications, most recently with high visibility in Atari [8] and Go [9]. Some of these applications, such as [9], exploit prior human experiences in the form of initial knowledge. Due to the high sample complexity of reinforcement learning, many speed-up techniques have been proposed in the past, of which leveraging human demonstration has indeed proved to be highly successful. One relevant recent technique, called Human Agent Transfer (HAT) [12], is framed as a *transfer learning* [11] technique. It allows a human (a source agent) to demonstrate its policy, and a learner (the target agent) to bootstrap its learning based on this policy. A major distinction of this method from other work on learning from demonstration is that a HAT learner seeks to eventually outperform the demonstrator, rather than just to reproduce the demonstrator's policy (as in imitation, or apprenticeship learning).

Given the success of demonstration-based RL in single agent settings, and HAT in particular, it is natural to ask to what extent can this technique be used in a multi-agent RL setting? We are particularly interested in framing collaborative multi-agent teams within the HAT framework. There has been very little work on training a team of autonomous agents from demonstration in the past. RLaR [5] is a related approach, where rather than a complete demonstration, an external entity (which could be a human) informs the learning agents about the parts of their state spaces invisible to them, enabling them to perform RL as a *rehearsal*, but the agents must learn policies that do not rely on the invisible parts of their states. In [3], similar on-line feedback is exchanged among the agents themselves, but in this work we seek to limit any advice to be an off-line prior.

One obvious dilemma with multi-agent learning from demonstration is regarding the size of the team trainable by human demonstration. On the one hand, it may be difficult for a single human to demonstrate multi-agent control when the number of autonomous agents in the team is large. On the other hand, requiring multiple human demonstrators—each controlling one or a small number of agents—delegates the question of agent coordination to that of coordination among the human demonstrators themselves. However, for teams of small sizes, it may be quite easy for a human to demonstrate effective coordination and help the team bootstrap RL. Hence we focus on small teams in this paper. In particular, our experiments are in 2-agent problem domains.

Another recent related work [6] seeks to exploit sports tracking data as demonstration for coordinated multi-agent imitation learning. This scenario also has multiple demonstrators, but they play a more passive role, and their coordination—often implicit in such data, and modeled as a latent variable—*is* the target of inference. As previously distinguished from the goal of imitation learning, we seek to exploit the demonstration to enable our agent team to eventually outperform the demonstrator team.

This paper investigates the following two questions. First, does the advantage of HAT extend to multi-agent systems? In particular, is a team of autonomous agents able to exploit human demonstration to bootstrap RL as much as a single agent? We show that, compared to single agent HAT where the initial performance (and often the asymptotic performance as well) of a HAT-agent is better than an unbiased agent, the difference is dramatically more pronounced in multi-agent HAT. In our experiments, the unbiased RL agents were completely unable to learn the tasks, whereas with multi-agent HAT, they were able to learn with a reasonable sample complexity and even outperform the demonstrator. We ground this investigation on two recent HAT methods: confidence based HAT (CHAT) and dynamic reuse of prior (DRoP).

Second, given that the demonstrator's global view of the task state can differ from the individual agents' (local) view of the state, what impact does this difference have on the performance of multi-agent HAT? In particular, agents following human demonstrations with respect to their individual observations may select actions different from what the human demonstrator would have picked with a global view of all agents' observations. We propose a novel confidence based technique—*Coordination Confidence* (CC)—that

Proc. Adaptive Learning Agents Workshop at the Federated AI Meeting, July 2018, Stockholm, Sweden

https://doi.org/doi

predicates the confidence of its action recommendations on this difference, allowing the team to improve its learning rate in a domain where this difference is critical. Our experiment in a second domain, where this difference is not critical, verifies that CC cannot improve the learning rate in such cases, but it does not impact the learning negatively.

2 BACKGROUND

This section discusses relevant background, including reinforcement learning, learning from demonstration, and human-agent transfer, and a recent advance to the latter.

2.1 Reinforcement Learning

Reinforcement learning problems are modeled as *Markov Decision Processes* or MDPs [10]. An MDP is given by the tuple $\langle S, A, R, P\rangle$, where S is the set of visible environmental states that an agent can occupy at any given time, A is the set of actions from which it can select one at a given state, $R : S \times A \mapsto \mathbb{R}$ is the reward function, i.e., $R(s, a)$ specifies the reward from the environment that the agent gets for executing action $a \in A$ in state $s \in S$; $P : S \times A \times S \mapsto [0, 1]$ is the state transition probability function specifying the probability of the next state in the Markov chain following the agent's selection of an action in a state. The agent's goal is to learn a policy $\pi : S \mapsto A$ that maximizes the sum of current and future rewards from any state s, given by, $V^\pi(s^0) = E_P[R(s^0, \pi(s^0)) + \gamma R(s^1, \pi(s^1)) + \gamma^2 \ldots]$ where $s^0, s^1, \ldots$ are successive samplings from the distribution P following the Markov chain with policy π.

Reinforcement learning algorithms, bereft of any prior knowledge of P or R, often evaluate an action-quality value function Q given by $Q(s, a) = R(s, a) + \max_\pi \gamma \sum_{s'} P(s, a, s')V^\pi(s')$. This quality value stands for the sum of rewards obtained when the agent starts from state s, executes action a, and follows the optimal policy thereafter. Model free RL methods directly learn $Q(s, a)$ by online stochastic approximation, e.g., *Q-learning* [10]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)],$$

where r and s' are sampled from R and P respectively. The final policy is calculated as

$$\pi(s) = \arg\max_a Q(s, a). \quad (1)$$

2.2 Learning from Demonstration

While often effective, RL can suffer from slow learning rates. One way of learning faster is to re-frame the problem to that of mimicking a demonstrator. In this case, the agent typically solves a supervised learning problem so that it can perform similarly to a demonstrator [1]. While the majority of such such *learning from demonstration* (LfD) methods assume there is only a single agent learning, there are some methods [2] that allow the demonstrator to provide demonstrations to individual agents in a multi-agent setting. Additionally, LfD methods typically assume that the demonstrator is near-optimal—even if there is an environmental reward signal, LfD methods typically do not use it to improve the policy.

To speed up RL, there are many methods for incorporating biases that can help an agent learn faster. One approach is to combine the sample-efficient learning of LfD with the reward-seeking behavior of RL, as discussed in the next section.

2.3 Human-Agent Transfer

Human-agent transfer (HAT) [12] reimagines demonstration based learning from the perspective of transfer learning [11], where knowledge gained from a prior task (often referred to as *source task*) is used as inductive bias to bootstrap RL in a related but different task (often referred to as *target task*). In lieu of a related source task, HAT views the demonstration itself as initial bias (coming from a *source* agent) that informs the RL's action choices, but gradually weans RL off this bias as the RL's own value function improves over time. In particular, the idea of HAT, in a single agent context, engages the following steps:

(1) Induce a supervised classifier with the demonstration, to approximate the source agent's policy $\pi_{Source} : S \mapsto A$.
(2) Bootstrap RL with the trained classifier, i.e., instead of acting randomly as an RL agent would initially, use the classifier for action selection, but only with a probability, Φ, that decreases over time. Thus an ϵ-greedy RL agent would select action as follows: select an action randomly with probability ϵ, select the classifier's predicted action with probability Φ, or select action according to Equation 1 with probability $1 - \Phi - \epsilon$.

Even though the demonstration may cover a small part of the state space, with a sufficiently rich representation of states the classifier's capability of generalization can be effectively leveraged to predict what the demonstrator's actions might have been in unseen states. Thus the classifier can serve as a powerful initial bias for RL. Different types of classifiers have been used in the past, e.g., decision trees, neural networks, and Gaussian processes [14]. Typically, an initial comparative analysis of different classifiers on the available demonstration data can inform the choice of the most appropriate classifier.

The above framework was originally introduced as *probabilistic policy reuse* [12], where Φ is called the *policy reuse parameter*. More recently, Wang and Taylor [14] leveraged the confidence in the classifier predictions to decide whether to rely on its prediction, or on RL's value function, in a refined framework called Confidence based HAT (CHAT). However, this framework still uses the policy reuse parameter Φ, which typically starts close to 1 but decays exponentially to enable the agent to rely more on its (increasingly better) learned action value function over time. The choice of the decay factor may pose a difficult tuning problem; too rapid decay may make RL rely prematurely on its learned Q-values, resulting sometimes in a temporary drop in performance and an associated "valley" in the learning curve. On the other hand, if the decay in Φ is too slow, then the performance of the learner, even when mature, can be skewed in the learning curve by the classifier's own performance. To address this dilemma, Wang and Taylor [15] introduced a new variant of confidence based HAT that does not require explicit decay of the reuse parameter, described in the next section.

2.4 Dynamic Reuse of Prior

The Dynamic Reuse of Prior (DRoP) algorithm [15] builds upon HAT by adding an on-line confidence measure to the transferred data. While the original DRoP paper includes two types of temporal difference confidence measures and three types of action decision models, we focus on one, each. In particular, step (1) in HAT, as

described in the previous section, is modified so that a confidence measure of the classifier is learned (i.e., the classifier can output both a label and a confidence for a given input). Step (2) is modified as described next, and then summarized in Algorithm 1.

A confidence-based TD model is used to analyze the performance level of every action source with respect to every state. The confidence in the classifier of prior knowledge for a given state, $CP(s)$, is initialized to be the confidence of the classifier, and is updated by the following equation over time

$$CP(s) \leftarrow (1-\alpha) \times CP(s) + \alpha \times [G(r) + \gamma \times CP(s')] \quad (2)$$

γ is the discount factor, r is the reward, and α is the update parameter. The reward is normalized[1] based on the confidence in the different actions:

$$G(r) = \frac{r}{r\_max} \times \max\left\{\frac{1}{\sum_i exp(\theta_i^T \cdot x)} \begin{bmatrix} exp(\theta_1^T \cdot x)) \\ exp(\theta_2^T \cdot x)) \\ \dots \\ exp(\theta_i^T \cdot x)) \end{bmatrix}\right\} \quad (3)$$

where $\frac{r}{r\_max}$ is a normalized reward (r_max denotes the maximum absolute reward value) and $G(r)$ re-scales the reward using the confidence distribution.

The confidence in the Q-value of a given state, $CQ(s)$, is initialized to be zero. It is updated over time via the following equation:

$$CQ(s) \leftarrow (1-\alpha) \times CQ(s) + \alpha \times [r + \gamma \times CQ(s')] \quad (4)$$

We use a probabilistic action selection mechanism[2]:

$$AS = \begin{cases} Q & P = \frac{\tanh(rCQ)+1}{\tanh(rCP)+\tanh(rCQ)+2} \\ Prior & P = \frac{\tanh(rCP)+1}{\tanh(rCP)+\tanh(rCQ)+2} \end{cases} \quad (5)$$

so that a high confidence in prior knowledge makes following the prior knowledge more likely, and a low confidence in the prior knowledge makes it less likely.

Algorithm 1 DRoP

```
for each episode do
  Initialize state s to start state
  for each step of an episode do
    if rand() ≤ ϵ then
      a ← random action %Exploration
    else
      %Select Action source (AS) via Equation 5
      if AS == Prior Knowledge then
        a ← action from Prior Knowledge
        Update CP via Equations 2 and 3
      else
        a ← action that maximizes Q
        Update CQ via Equation 4
      end if
    end if
    Execute action a, Observe new s' and r
    Update Q via SARSA, Q-Learning, etc.
  end for
end for
```

[1]This is called the *Dynamic Confidence Update* method in the original DRoP paper.
[2]This is called the *soft decision model* in the original DRoP paper.

3 HUMAN MULTI-AGENT TRANSFER

Human multi-agent demonstrations are collected as joint-state joint-action vectors $\langle \vec{s}, \vec{a} \rangle$, where the state s_i of the ith agent in the team is a feature vector over its local sensor data. Agents must learn to map their own states to their own actions ($\pi_i : S_i \mapsto A_i$), therefore the demonstration can be separated out agent-wise, and the ith agent can be given its own portion of state-action demonstration, $\langle s_i, a_i \rangle$, to use as prior as shown in Figure 1. This is a straightforward extension of HAT to a multi-agent setting. However, as we argue below, this first-cut approach may not work well in some domains.

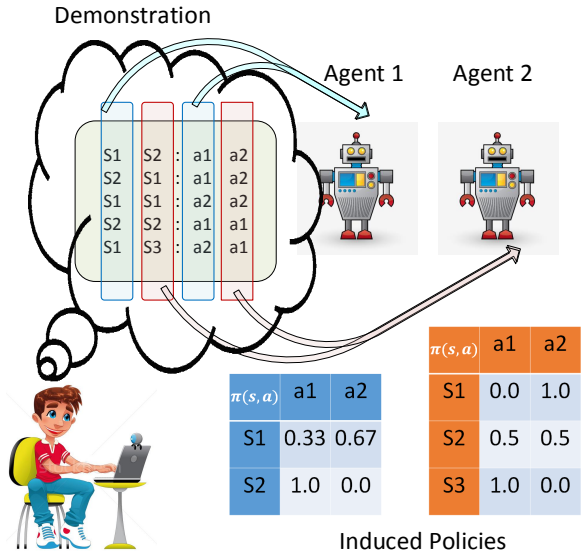

Figure 1: The human-multi-agent transfer framework.

3.1 Difference in Perspectives

An important difference between the human demonstrator and the agents is the difference in perspectives. In single agent LfD, the human and the agent will perceive the task environment through different sets of sensors, raising the question whether, or to what extent, the policy of one is usable by the other. In a multi-agent setting, this question is of magnified import. The demonstrator usually sees the global state (i.e., the states of all agents in the team) and demonstrates actions for all agents. However, each agent is only capable of observing its own local state, and can only execute its own actions. This scenario is depicted in Figure 1 for a two-agent team. While the demonstrator records joint actions against joint states, agents induce classifiers that map their individual states to individual actions. In the example of Figure 1, the induced policies will only be perfectly coordinated when the agents observe joint state $\langle S2, S1 \rangle$. For joint states $\langle S1, S1 \rangle$ and $\langle S2, S2 \rangle$, they will be miscoordinated 33% and 50% of the time respectively, but in joint state $\langle S1, S2 \rangle$ they will be miscoordinated over 83% of the time! It seems in the last case the agents are better off randomly exploring instead of following the demonstrator's recommendation. Notice that an agent cannot infer such states (where the risk of miscoordination is high) from its own classifier's confidence alone. E.g., in state $\langle S2, S2 \rangle$, even though agent 1's classifier is fully confident about recommending action $a1$, agent 1 will still be miscoordinated 50%

of the time, because it does not know agent 2's state (and hence its confidence at that step).

3.2 Coordination Based Prior

The key to solving the above problem is to note that the information needed to predict when an agent might be more prone to miscoordination when following the human demonstration is already present in the joint demonstration $\langle \vec{s}, \vec{a} \rangle$. Therefore we propose to make the entire joint demonstration available to each agent, so that each agent can perform an independent offline analysis that can be used later to predict where the agent is more (or less) likely to be miscoordinated with its teammates. This information can be seen as an additional prior for each learning agent, apart from the individual policy induced from the human demonstration. We discuss the methodology in detail next.

4 COORDINATION CONFIDENCE

The collaborative team tasks constituting the focus of this paper can be viewed as *identical payoff games*, where every agent in the team receives the same reward. Joint actions that are coordinated lead to higher values of the agents' payoffs, while miscoordinated joint actions lead to poor payoffs. The coordinated joint actions form the *Nash equilibrium* [4] of the game, the highest valued ones being Pareto dominant. While a Nash equilibrium assumes a probabilistically independent choice of agent actions, when the agent choices are somehow correlated, the resulting equilibrium—called *correlated equilibrium*—can provide higher payoffs to the agents in identical payoff games [4]. However, the latter requires a correlation device—a signal that is commonly observed by the agents. One view of our idea of coordination based prior is that apart from the commonly observed reward, the joint demonstration serves as an additional correlation device, observed as an offline prior, but used online during RL. We speculate that this additional correlation device may allow the agents to learn a higher valued correlated equilibrium than those that do not have access to this device. We now discuss how the agents distill this correlation device into a confidence measure to bias independent RL.

4.1 Offline Analysis

Given the agent-specific demonstration, $\langle s_i, a_i \rangle$, the ith agent trains a classifier via supervised learning, $C_i : S_i \mapsto \Delta A_i$, that returns a confidence distribution over its actions. But since it also receives the entire demonstration, it can also train such classifiers for other agents $j \neq i$, $C_j : S_j \mapsto \Delta A_j$, as well as a single classifier that maps a joint state to a confidence distribution over joint actions, $C : S \mapsto \Delta A$. Then given a state s_i, the agent computes a confidence measure, that indicates that if it follows its classifier then the agent will be coordinated with the other agents, as

$$\theta(s_i) = 1 - \frac{\sum_{s_j \in \vec{s}, j \neq i} \| \otimes_j C_j(s_j) - C(\vec{s}) \|_2 N_{j,i}}{M_i \sqrt{2}}, \tag{6}$$

where $\otimes$ is the Cartesian product, $N_{j,i}$ is the number of times in the full demonstration where the ith agent's state was s_i while the others' state matched s_j, M_i is the number of times in the full demonstration where the ith agent's state was s_i, and $\sqrt{2}$ is the required normalization factor for norm-2 distance between probability distributions. Because $\sum_{s_j} N_{j,i} = M_i$, and the maximum norm-2 distance between two probability distributions is $\sqrt{2}$, $\theta(s_i)$ in Equation 6 is bounded in $[0, 1]$. At one extreme, if $\theta(s_i) = 1$, then agent i can expect that if it uses C_i and other agents use their classifiers, then their joint action will be perfectly coordinated. In other words, the agent's local view s_i of the joint state $\vec{s}$ is completely sufficient for it to select the proper action. At the other extreme, $\theta(s_i) = 0$, which means that if it uses C_i and other agents use their classifiers, then their joint action will be perfectly *mis*coordinated. That is, the agent's local view s_i of the joint state $\vec{s}$ is insufficient; someone with a global view of $\vec{s}$ would be able to select a coordinated joint action where i's portion a_i was deterministically different.

Although the computation of $\theta(\cdot)$ is lower bounded by $\Omega(|\otimes_i A_i|)$, which is exponential in the number of agents, it is performed offline and hence the cost is amortized.

4.2 Online Use of Coordination Confidence

Let the states of agent i visited during the demonstration form the subset $S_i^D \subset S_i$. During explorative reinforcement learning, the agent will potentially visit many other states not seen by the demonstrator. Rather than training an inductive regressor to predict θ for such states, we apply transductive or instance-based prediction. For a state $s_i \in S_i \setminus S_i^D$, we optimistically predict

$$\begin{aligned} &\bar{\theta}(s_i) = \max\{\theta(s_i') | s_i' \in \Sigma_{s_i}\}, \\ &\Sigma_{s_i} = \{s_i' | d(s_i, s_i') \leq d(s_i, s_i'') \leq \epsilon, \forall s_i', s_i'' \in S_i^D\} \end{aligned} \tag{7}$$

where d is a feature distance measure. In words, among all of agent i's states in the demonstration that are closest (by distance measure d) to s_i but not farther than ϵ, $\bar{\theta}$ optimistically selects the highest θ confidence using Equation 6. Depending on the choice of ϵ, Σ_{s_i} could be empty for some s_i, in which case $\bar{\theta}$ is set to 0. This online computation is $O(|S_i^D|^2)$, hence a constant for a fixed amount of demonstration. Algorithm 2 shows the CC-Agent algorithm, which opts for coordination confidence when it is high, but otherwise falls back on CHAT (or Q when CHAT's confidence is low).

5 EXPERIMENTAL EVALUATION

We use two domains for experimental evaluation: Furniture Movers [2] and Block Dudes. We describe these domains in the following subsections, and present the experimental results alongside. In all experiments, we used random forests as the classifier because of its improved accuracy in both domains compared to other classifiers reported in previous literature. The result of this preliminary analysis, performed in Weka 3.8.1 is shown in Table 1.

Table 1: Classifier accuracies by F-Measure averaged over 10-fold cross validation, performed on human demonstration in Furniture Movers (FM) and Block Dudes (BD). All experiments used the default parameters in Weka.

	Decision Tree	Neural Network	Gaussian Process	Random Forest
FM	94.4%	93.3%	94.8%	96.6%
BD	89.3%	81.1%	90.2%	90.8%

Algorithm 2 CC_Agent_i(*conf_threshold*)

1: Get local state s_i
2: **if** rand()$\leq \Phi$ **then**
3: **if** $(\bar{\theta}(s_i) \geq$ *conf_threshold*$)$ **then**
4: %Use Coordination Confidence
5: $a_i \leftarrow \arg\max_{action} C_i(\arg\max_{s_i' \in \Sigma_{s_i}} \bar{\theta}(s_i'))$
6: Note whether this action is same or different from CHAT action
7: **else if** $(\max C_i(s_i) \geq$ *conf_threshold*$)$ **then**
8: %Use CHAT action
9: $a_i \leftarrow \arg\max_{action} C_i(s_i)$
10: **else**
11: $a_i \leftarrow \arg\max_a Q(s_i, a)$
12: **end if**
13: **else**
14: **if** rand()$\leq \epsilon$ **then**
15: $a_i \leftarrow$ random action
16: **else**
17: $a_i \leftarrow \arg\max_a Q(s_i, a)$
18: **end if**
19: **end if**
20: Execute a_i, get reward r, and update Q using the next state

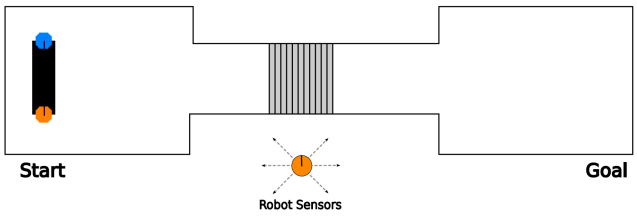

Figure 2: Furniture Movers domain from [2].

5.1 Furniture Movers

5.1.1 Domain Description. The Furniture Movers domain is shown in Figure 2, adopted from [2]. First we give the description of this domain from [2], and then specify the changes we made.

Two agents are jointly tasked with moving a long and heavy piece of furniture from the room on the left to the one on the right, through a narrow corridor which also has a staircase. Each agent has a set of six range sensors as shown, each giving it a noisy reading of the nearest obstacle in that direction. An agent can also sense the stairway, but only when next to (or on) it. An agent can move in one of the four cardinal directions, or execute a "stair" action. Agents can successfully navigate the stairway only when both execute the "stair" action (though both may not see it simultaneously), otherwise they stay stuck. Furthermore, joint actions that make the agents lose their grip on the furniture (e.g., moving in opposite directions) also do not change their locations. Apart from the five actions, an agent can also execute a *communicate* action to inform the other if it sees the stairs, to enable coordination for joint "stair" action. Thus the state feature vector of each agent has, in addition to the 6 range sensor measurements, two binary stair features, one for its own location and one for its teammates'. The teammate's stair feature can only be updated via the communicate action from the other. As in [2], the movements of the agents are discretized, and the task can be completed optimally in 39 steps, indicating that the joint start and goal locations were fixed.

We made the following changes to this domain:

- As movements are discretized, we allow each agent to move in any of the eight directions, rather than four. This expands the action space of each agent, making the joint learning problem even more challenging for independent Q-learners, especially without any prior.
- We select the joint start locations randomly within the left room, and the joint goal location is set to anywhere in the right room. That is, agents can start anywhere in the left room, both holding the furniture at its opposite ends, and as soon as both are located anywhere in the right room (again both holding the furniture) we say that the episode has ended successfully. These settings were the same for the human demonstrator, as for the experiments.
- We introduced 10% action noise for joint actions, that is, agents' joint actions are changed randomly 10% of the time after they have selected their actions, but before the actions are executed in the environment. However, the changed actions are not allowed to include *communicate*. The purpose of this noise is to create variations in the demonstrated trajectories, since the demonstrator could otherwise follow a fixed trajectory after the agents have successfully entered the narrow corridor.
- We set the reward function to be -1 for all actions, except the joint action that ends the episode successfully, which has a reward of +100 for each agent.

5.1.2 Experimental Results in Furniture Movers. Because of the first three changes, the optimal episode length is different from 39, and only fixed in expectation. The task is extremely challenging to independent Q-learners (IQL) with no prior, to the extent that they are completely unable to learn, with all learning episodes being capped at 200 steps, as shown in Figure 3. This figure shows the cumulative average of the discounted episode returns obtained by the learners, with each plot point being averaged over 32 independent trials. Each agent used $\epsilon = 0.05$, $\gamma = 0.999$, and $\alpha = 0.005$; CHAT and CC used a confidence threshold of 0.7, and a Φ decay rate of 0.999977. With an experiment horizon of 1.5×10^5 episodes, this decay rate reduces Φ to ≈ 0.03. Figure 3 shows that in contrast with IQL, all three versions of HAT are able to learn this task, CC outpacing CHAT and DRoP at a statistically significant rate. Furthermore, Table 2 shows the initial and learned episode lengths (average of first 20,000 and last 20,000 episode lengths, respectively) of these three algorithms in this task, together with the demonstrator's performance for comparison. Here we see that CC has a higher jumpstart (difference in initial performance, from no-prior IQL) than CHAT and DRoP, but all three algorithms ultimately learn to perform similarly. Note that they learn to outperform the human demonstrator.

In order to further investigate the performance improvement of CC over CHAT, we scrutinized the action choices of Algorithm 2, particularly how often it selected actions by coordination confidence (line number 5), CHAT (line number 9), or Q (line numbers 11, 17). Furthermore, when Algorithm 2 chose an action recommended by coordination confidence, we also compared them to those that

Table 2: Summary of episode lengths (lower is better) in Furniture Movers (FM) and Block Dudes (BD). Total learning episodes were 1.5×10^5 and 3.0×10^6 respectively.

	Human	CHAT	CC	DRoP
FM Initial	42 ± 7.6	101 ± 5	88 ± 3.7	109 ± 4.8
FM Learned	-	34.7 ± 0.76	34.4 ± 0.63	34.6 ± 0.36
BD Initial	87 ± 20.6	80.3 ± 3.7	82.2 ± 1.4	108.6 ± 3.1
BD Learned	-	39.9 ± 0.5	39.3 ± 0.3	48.2 ± 2

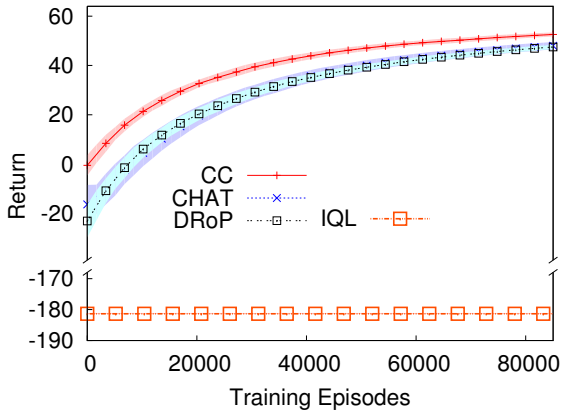

Figure 3: Learning curves in Furniture Movers. Episode lengths were capped at 200 steps, which IQL always reached.

CHAT would recommend, i.e., whether the recommendations would have differed from CHAT (line number 6). Figure 4 shows the result of this experiment, with a moving window average over a window size of 20000 episodes, averaged over 32 trials. Clearly, the CC-Agent leverages coordination confidence much more often than the CHAT confidence. Also notice the high frequency of CC != CHAT, which stands for the action choices that are solely attributable to coordination confidence, and are distinct from CHAT's classifier. Although this rate tapers off due to Φ decay (as do the other rates, while the rate of Q-value utilization replaces them), the distinct action choices prompted by coordination confidence explains the superior jumpstart and learning rate of CC-Agent. Figure 4 shows the actual action counts, thus the sum of the counts at any episode is also close to the episode length (sans the ϵ exploration).

Figure 5 shows the action choice frequency of DRoP due to various sources of confidence: CC, CHAT, or Q; with a moving window average over window size of 20000 episodes, averaged over 32 trials. Since DRoP does not decay the rate of prior usage, we see that all three sources continue to be used (with Q being slightly dominant for the most part), although overall rates fall because learning shortens the episodes, reducing the number of actions chosen. Also notice that CC (i.e., CC = CHAT + CC != CHAT) does not clearly dominate CHAT as a source, unlike in Figure 4, which means that DRoP is unable to leverage the unique information conveyed by coordination confidence, which would explain why it performed similarly to CHAT, and underperformed in Figure 3. Finally, Figure 6 shows the actual values of confidence from different sources for DRoP, verifying that Q is indeed mostly dominant as a source.

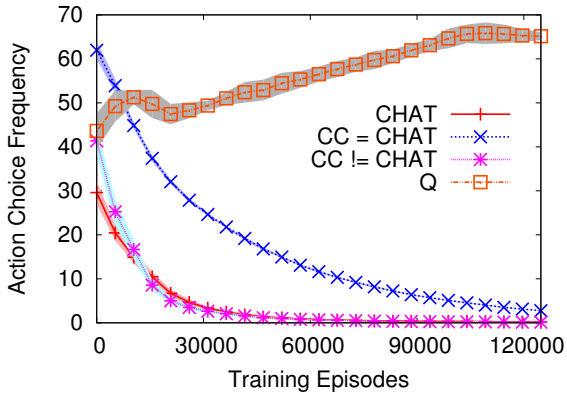

Figure 4: Action choice frequency (sum over 2 agents) of various sources in Algorithm 2: CHAT, Q, or CC, in Furniture Movers. The last count is further broken into whether the action is same as CHAT or different.

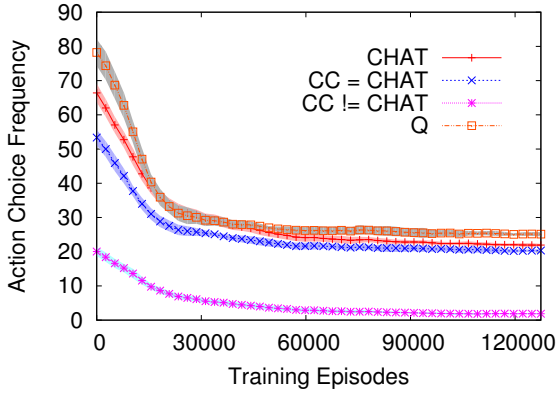

Figure 5: Action choice frequency (sum over 2 agents) of various sources for DRoP, in Furniture Movers. Action selection driven by 3 different sources: CHAT, Q, or CC. The last count is further divided into if the action is same as CHAT or not.

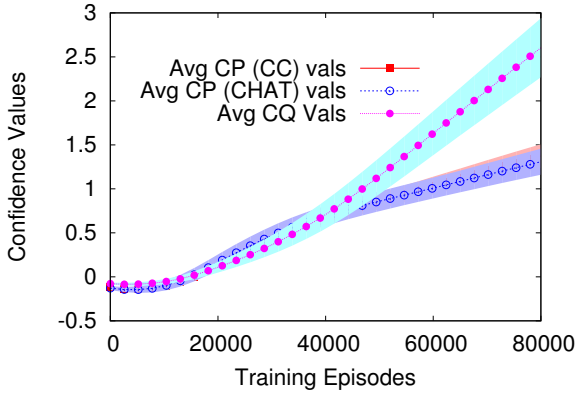

Figure 6: Average confidence values of three sources for DRoP, per agent, in Furniture Movers.

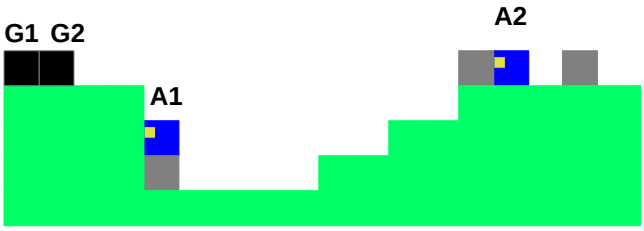

Figure 7: Block Dudes domain.

5.2 Block Dudes

5.2.1 Domain Description. We introduce a new challenging domain for cooperative multi-agent planning and learning, called Block Dudes. It is based on a TI calculator puzzle concept as implemented in BURLAP [7], but with two agents instead of one. The initial state is depicted in Figure 7. The agents, marked A1 and A2 (blue, with an eye showing its current heading), are located in some landscape, along with three blocks (gray). The goal of agent i is to reach the cell marked Gi (black). The initial configuration is such that the agents are incapable of reaching the goal without using the blocks. Furthermore, the world is two dimensional, so the agents cannot pass each other, or co-occupy any cell. Thus the quickest way for the agents to reach their goals is to cooperate and pass blocks from right to left to stack at the bottom of the cliff, creating a stairway to climb up to their goals.

Each agent is only capable of local sensing, to know its own location, heading, and whether it is currently carrying a block. Further, an agent can sense the features of the other agent when within a horizontal distance of 4, but not otherwise. However, we found that merely local sensing of the blocks made independent RL difficult, therefore following the implementation in BURLAP we made all blocks visible to both agents at all times. Note that this allows an agent to infer the location of the other agent if the latter is carrying a block, even when they are farther apart than 4 units. An agent's action space contains 5 actions: *east, west, up, pickup, putdown*. The directional actions, *east* and *west*, either changes the heading of the agent, or progresses it one step in the direction of its heading, as long as this is allowed by the terrain. The action *up* allows the agent to climb up a block (irrespective of whether or not it is carrying a block) that is in front of it and at the same level. The actions *pickup* allows the agent to pickup a block that is in front of it and at the same level; *putdown* allows it to drop a block that it is carrying as long as the height of the terrain in front of the agent is no larger than the height of the agent. As with our modification of Furniture Movers, we added 10% joint action noise, but there is no sensor noise.

The variance of the demonstrator in Block Dudes was high (see Table 2) because certain action choices (either by mistake, or by action noise) can lead to dramatically different trajectories. For instance, if A2 drops a block at the edge of a step, then A1 will neither be able to pick it up, nor be able to climb over, because in either case it needs to be at the same level as the block. This necessitates a costly detour where A1 goes back to grab the left-most block and use it to regain access to the inaccessible block. While the human demonstrator was repeatedly frustrated by such unexpected calls for replanning, the learners were able to minimize them (evident from their low variance) e.g., by dropping blocks early. Some demonstrations ended in failure because the blocks had been configured in ways that made it impossible to achieve the joint goal. Such demonstrations were discarded. In the end, 11 successful demonstrations were collected. The human demonstrator found this task challenging because of unforeseen variations, but also frustrating because noise often preempted future plans.

5.2.2 Experimental Results in Block Dudes. The reward function in this domain is similar to Furniture Movers: +100 to each agent only when both agents are in their respective goal cells, -1 for all other situations. Because this domain appeared challenging, we reduced the learning rate to $\alpha = 0.001$, and slowed the Φ decay rate to 0.9999992, but kept the other parameters ϵ, γ unchanged from Furniture Movers. When reporting performances with windowed averages, we raised the window size from 20,000 in Furniture Movers to 50,000 in this domain, and ran our experiments for 3×10^6 episodes with 32 trials. Notice the Φ decay is slower here due to the increased experiment horizon; Φ decays to ≈ 0.09. Despite the slow decay of Φ, the 'valley' mentioned in Section 2.3 still appeared in the learning curves in Figure 8, exhibiting the difficulty of tuning this parameter with the learning rate.

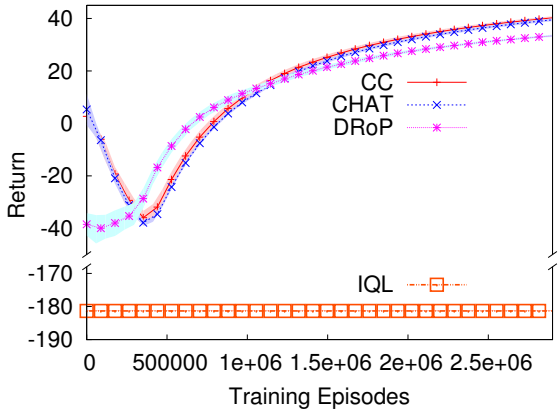

Figure 8: Learning curves in Block Dudes domain. Episode lengths were capped at 200, which IQL always maxed out.

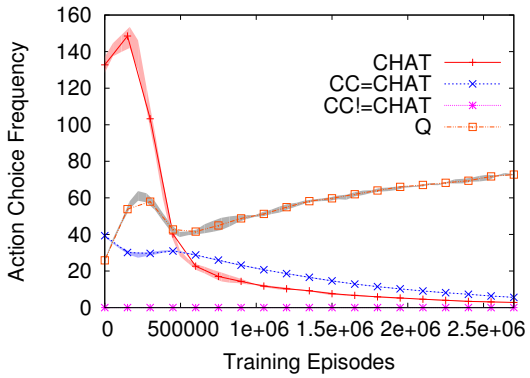

Figure 9: Action choice frequency (sum over 2 agents) of various sources in Algorithm 2: CHAT, Q or CC, in Block Dudes. The last count is further broken into whether the action is same as CHAT or different.

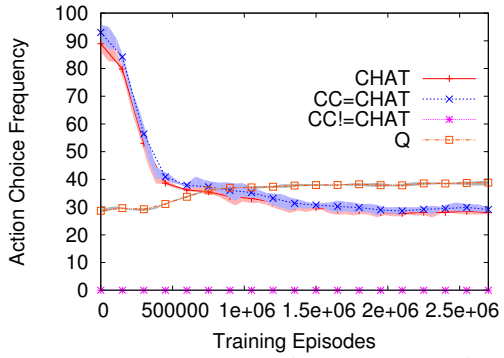

Figure 10: Action choice frequency (sum over 2 agents) of various sources for DRoP in Block Dudes. Action selection driven by 3 different sources: CHAT, Q, or CC. The last count is further broken into whether the action is same as CHAT or different.

It is noteworthy in Figure 8 that CC does not have a statistically significant advantage over CHAT in this domain, but DRoP outperforms both in learning rate, and does not exhibit the 'valley' characteristic of other algorithms that depend on a decaying prior reuse rate. However, its asymptotic performance is lower than CC or CHAT. As with Furniture Movers, IQL fails to learn once again.

The reason why coordination confidence fails to improve performance over CHAT is that the agents in Block Dudes are far more independent and less reliant on coordination, i.e., they are *weakly coupled*. In fact, most of the time, agents can execute their own policies without regard to what the other agent is doing at that time. The agents only need to coordinate their actions when they are in close proximity, but in these cases their individual states contain the features of the other (visible) agent, thus giving them sufficient information to select a coordinated action. The difference in perspectives between local states and global states becomes irrelevant, therefore the CC-Agents show the same performance as CHAT. This intuition is verified in Figure 9, which shows that the action frequency for cases that are distinct from CHAT (CC != CHAT) are practically 0. Also, in contrast with Figure 4, the action frequency of CHAT is dominant in Figure 9, indicating that the test in line 3 of Algorithm 2 fails frequently, verifying the low utility of coordination in this domain. Transfer learning approaches [13] designed for such sparse interaction domains might yield better results in Block Dudes.

Finally, Figure 10 shows the action choices of DRoP due to the three sources, CC, CHAT and Q. In contrast with Figure 5, the HAT sources dominate initially, with the learned Q function eventually exploited more often. But the initial dominance of HAT sources explains the improved learning rate of DRoP.

6 CONCLUSION

We have verified via experiments in two domains that Human-Agent Transfer (HAT) when extended to multi-agent RL, is indeed advantageous for agents to bootstrap RL, but to a greater extent in multi-agent systems than in single agent RL as studied in the past. While it imparts a jumpstart and improves the learning rate in single agent systems, multi-agent systems may be completely unable to learn without demonstration.

We have proposed a method to measure the degree of confidence that agent teams can have regarding coordination in demonstrated data, and leverage this confidence for action selection. We have found that this information can be advantageous in a tightly coupled domain where agents need to coordinate often, but their individual local states may not correlate strongly with the global state. On the other hand, in weakly coupled domains, where agents need to coordinate infrequently, and when they do, their individual local states are strongly correlated with global states, this confidence does not provide any added advantage.

We have also verified the difficulty of parameter tuning for the decay of prior reuse rate in HAT derivatives, and that a recently proposed technique to avoid this tuning in single-agent systems, called DRoP, can indeed avoid the "valley" in learning curves in multi-agent learning. This can result in faster learning for the team. However, we found that DRoP may not achieve similar jumpstart as CHAT or CC, which may call for a more informed way to initialize the confidence values of various sources of priors in DRoP.

Acknowledgment This work was supported in part by the National Science Foundation (IIS-1526813).

REFERENCES

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A Survey of Robot Learning from Demonstration. *Robot. Auton. Syst.* 57, 5 (May 2009), 469–483. DOI: http://dx.doi.org/10.1016/j.robot.2008.10.024

[2] Sonia Chernova and Manuela Veloso. 2007. Multiagent collaborative task learning through imitation. In *Proceedings of the 4th International Symposium on Imitation in Animals and Artifacts (AIBS-07), Artificial and Ambient Intelligence*. Newcastle, UK.

[3] Felipe Leno da Silva, Ruben Glatt, and Anna Helena Reali Costa. 2017. Simultaneously Learning and Advising in Multiagent Reinforcement Learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS-17)*.

[4] D. Fudenberg and K. Levine. 1998. *The Theory of Learning in Games*. MIT Press, Cambridge, MA.

[5] Landon Kraemer and Bikramjit Banerjee. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190 (2016), 82–94.

[6] Hoang M. Le, Yisong Yue, Peter Carr, and Patrick Lucey. 2017. Coordinated Multi-Agent Imitation Learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*.

[7] James MacGlashan. 2014. The Brown-UMBC Reinforcement Learning and Planning (BURLAP) Library. http://burlap.cs.brown.edu/. (2014).

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533.

[9] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016), 484–489.

[10] R. Sutton and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

[11] Matthew E. Taylor and Peter Stone. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research* 10, 1 (2009), 1633–1685.

[12] Matthew E. Taylor, Halit Bener Suay, and Sonia Chernova. 2011. Integrating Reinforcement Learning with Human Demonstrations of Varying Ability. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[13] Peter Vrancx, Yann-Michael De Hauwere, and Ann Nowe. 2011. Transfer Learning for multi-agent coordination. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*.

[14] Zhaodong Wang and Matthew E. Taylor. 2017. Improving Reinforcement Learning with Confidence-Based Demonstrations. In *Proceedings of the 26th International Conference on Artificial Intelligence (IJCAI)*.

[15] Zhaodong Wang and Matthew E. Taylor. 2018. *Interactive Reinforcement Learning with Dynamic Reuse of Prior Knowledge from Human/Agent's Demonstration*. Technical Report. https://arxiv.org/abs/1805.04493