Dec-AIRL: Decentralized Adversarial IRL for Human-Robot Teaming

Prasanth Sengadu Suresh
THINC lab, School of Computing
University of Georgia
Athens, GA, USA
ps32611@uga.edu

Yikang Gui
THINC lab, School of Computing
University of Georgia
Athens, GA, USA
Yikang.Gui@uga.edu

Prashant Doshi
THINC lab, School of Computing
University of Georgia
Athens, GA, USA
pdoshi@uga.edu

ABSTRACT
We present a new method for inverse reinforcement learning (IRL) that allows an agent to learn from expert demonstrations and then spontaneously collaborate with a human on the same task. We generalize adversarial IRL (AIRL) to work in a decentralized setting using a decentralized Markov decision process (Dec-MDP) as the underlying model. We posit that a Dec-MDP is a better-suited model for pragmatic multi-agent IRL compared to the multi-agent Markov decision process (MMDP) or the Markov game, which have been utilized thus far. This is because the latter models require an agent to know the global state of the environment, which is impractical in the real world as it may include agent-specific attributes (e.g., joint angles) that may not be directly observable by the other agents. We test our method on two domains: a formative simulated patient assistance scenario and a summative real-world use-inspired domain of sorting onions on a line conveyor. Our method (Dec-AIRL) significantly improves on the previous techniques in both domains. These results indicate that a decentralized multi-agent IRL formalism promotes effective teaming in human-robot collaborative tasks.

KEYWORDS
Cobots; Human-Robot Teaming; Learning from observations; IRL

ACM Reference Format:

1 INTRODUCTION
Inverse reinforcement learning (IRL) [28] is the paradigm of using demonstrations provided by expert agent(s) to learn complex task preferences that are often non-trivial to program. It aims to learn a reward function that provides an accurate representation of the expert’s preferences. While there is much work on IRL applied to single agent scenarios, IRL for multi-agent tasks is less explored [2]. One key application of multi-agent IRL is in human-robot collaborative settings where the demonstration is provided by a human-human team, which may then be used by a robot to learn how to perform the task and actively collaborate with a human in the shared environment. For instance, consider a produce processing shed where humans stand across from each other and sort produce on a conveyance system by removing the blemished produce. Is it possible for a cobot to observe and learn to seamlessly collaborate with the humans on the line?

Toward realizing this, a key challenge is to develop IRL methods to learn the reward function of a collaborative multi-agent team. In this regard, recent methods generalize the well-known adversarial IRL (AIRL) and generative adversarial imitation learning (GAIL) [8, 15] to a multi-agent setting. These include multi-agent AIRL [37, 40] and its extension to demonstrations with latent variables [13], as well as Co-GAIL [35] and MA-GAIL [32]. These methods generalize the Markov decision process (MDP) model of the expert’s behavior to either the Markov game [20] or the MMDP [4]. However, both these models require each participating agent to perfectly know the global state of the system, which may not be practicable. We illustrate this limitation in the context of produce processing where the produce is, say onions: the sufficient global state also includes variables indicating whether an agent has identified the onion as being blemished or not. While the robot’s sensors facing the human can assess the onions on the conveyor, this may not align with the human’s inspection of the onion. As such, some of the variables in the global state that pertain to the other agent may not be perfectly determined, and this may lead to adverse interactions such as both agents reaching for the same onion.

Given this limitation of extant methods, we posit that a decentralized MDP (Dec-MDP) [12], which can be solved to produce a vector of policies (one for each agent) in which an agent’s actions are conditioned on its own history of observations, is a more appropriate model. Therefore, we present a new generalization of AIRL to multi-agent settings, which we label as Dec-AIRL, that ascribes a Dec-MDP to model the multi-agent collaboration. Dec-AIRL follows the generator-discriminator architecture where the generator is modeled as performing RL using centralized training and decentralized execution (CTDE) [39]. It generates trajectories of states and actions for each agent, which are used by the discriminator to inversely learn the joint reward function.

We evaluate the performance of Dec-AIRL on two domains and compare its empirical learned behavior accuracy (LBA) with that of Co-GAIL and MA-AIRL. The expert trajectories for our simulated assistive domain comes from a trained RL model. For the realistic produce-processing domain, we deploy our learn-from-observation (LFO) pipeline (see Fig. 1) to record and process RGB-D camera streams of humans engaged in sorting batches of onions on a table. While Co-GAIL uses an MMDP to model the human-robot collaborative problem and assumes access to the global state and the latent features pertaining to the human behavior, MA-AIRL models it as a Markov game, which also maps the global state but to each
agent’s action, and obtains a game-theoretic equilibrium as the convergence criterion. Our results show that Dec-AIRL significantly improves on the accuracy of both Co-GAIL and MA-AIRL mainly because of the challenge to each agent of inferring the true global state perfectly. On deploying the learned policy in simulation and on a physical cobot, we show that a collaborative robot (cobot) is able to successfully avoid adverse interactions much better than the baseline policies, as it collaborates with a human.

2 BACKGROUND

In a single agent scenario, IRL models the expert’s environment as a Markov decision process (MDP) [24], which it solves optimally. Formally, the MDP of an expert is defined as a tuple \((S, A, T, R, y, \rho_0)\), where \(S\) is the set of states defining the environment, \(A\) is the expert’s set of possible actions, \(T : S \times A \times S \to [0, 1]\) gives the transition probabilities from any given state to a next state for each action, \(R : S \times A \to \mathbb{R}\) is the reward function modeling the expert’s reward for performing an action from a state, \(y \in [0, 1]\) is the discount factor and \(\rho_0\) is the initial state distribution. Typically, the learner is aware of the expert’s \(S, A, y, \rho_0\), and \(T\) but not \(R\). Model-free techniques do not require \(T\) either. A (stationary) policy for a MDP is a mapping from states to actions \(\pi : S \to A\). Let \(X^E\) be the set of expert demonstrations and a complete trajectory \(X\) is given by, \(X = (s_1, a_1, s_2, a_2, s_3, ..., s_T, a_T)\); where \(X \in X^E\).

2.1 Review of Adversarial IRL

AIRL builds on the maximum causal entropy framework [41], which considers an entropy-regularized MDP. Forward RL aims to find the optimal policy \(\pi^*\) that maximizes the expected entropy-regularized discounted reward, under \(\pi\), \(T\), and \(\rho_0\):

\[
\pi^* = \arg\max_{\pi} \mathbb{E}_{X \sim \pi} \left[ \sum_{t=0}^{T} \gamma^t (r(s_t, a_t) + H(\pi(\cdot | s_t))) \right].
\]

The distribution of the trajectories derived from the optimal policy \(\pi^*(a|s)\) has been shown to take the form \(\pi^* (a|s) \propto e^{Q^\text{sof}(s,a)}\) [14, 41], where:

\[
Q^\text{sof}(s_t, a_t) = r_t(s,a) + E_{s_t+1:\gamma \sim \pi} \left[ \sum_{t'=t}^{T} \gamma^{t'-t} (r(s_{t'}, a_{t'}) + H(\pi(\cdot | s_{t'}))) \right].
\]

The maximum causal entropy framework parameterizes the reward functions \(r_\theta(s,a)\) but fixes initial state distribution and the dynamics. The structure of the discriminator is adapted from GCL [7] where the discriminator \(D_\theta(X)\) learns a function \(f_\theta(X)\) [8] which at optimality approximates the expert policy’s advantage function [1]. The softmax policy [11] is given as:

\[
\pi(X) = e^{Q^\text{sof}(s,a)} = e^{Q_\theta^\text{sof}(s,a) - V_\theta^\text{sof}(s)}
\]

(1)

The discriminator takes the form, \(D_\theta(X) = \frac{e^{Q_\theta(X)}}{e^{Q_\theta(X)} + \pi(X)}\) and the reward update rule is given as:

\[
R_\theta(X) \leftarrow \log D_\theta(X) - \log (1 - D_\theta(X)).
\]

AIRE minimizes the reverse KL divergence between the learner’s and expert’s marginal state-action distribution \(KL(\rho_\theta(s,a)||\rho_{\text{exp}}(s,a))\) since this results in mode-seeking behavior in contrast to behavior cloning methods that use forward KL divergence with a mode-covering behavior [10]. One key difference between the original formulation by Fu et al [8] and our approach is that we model the reward function as a function of both states and actions as the disentangled reward learning comes with some strong assumptions [9, 11, 34].

2.2 Proximal Policy Optimization

Proximal policy optimization (PPO) is the forward rollout method used by AIREL [8]. PPO improves upon trust-region policy optimization (TRPO) by using a lower-bound on the unclipped objective. More specifically, TRPO maximizes the policy objective: \(L(\omega) = E_\pi \left[ A^\prime \pi \omega \right]\); where \(A^\prime\) is the advantage function (generalized advantage estimate used in practice) at the current timestep and the importance sampling ratio \(\lambda^\omega\) is given as \(\frac{\pi^\omega(a|s)}{\pi_{\text{old}}(a|s)}\), where \(\pi_{\text{old}}\) is the policy from the previous iteration and \(\omega\) are the weights for the policy. However, this may lead to an excessively large policy update [29], and PPO suggests the following policy objective:

\[
L^{\text{CLIP}}(\omega) = E_t \left[ \min \left( \lambda^\omega A^\prime_{\text{old}}, \text{clip}(\lambda^\omega, 1 - \epsilon, 1 + \epsilon) A^\prime_{\text{old}} \right) \right]
\]

where \(L^{\text{CLIP}}(\omega)\) is the clipped surrogate objective that takes the minimum of the clipped and unclipped objective to maintain a pessimistic bound over the final objective. Clipping the importance...
This may result from hardware inaccuracies (e.g., sensor noise) or environmental limitations. If no attribute of the system state is hidden from the agents and the combination of agents’ observations results in the global state, we may adopt a jointly fully-observable Dec-MDP model [3]. Finally, in cases where certain features of the system are never perfectly revealed to the agents the appropriate way to model the environment would be as a Dec-POMDP.

3.2 IRL from Team Demonstrations

Let the joint experts’ demonstration be given as:

\[ X^E = (s^E_0, s^I_0, a^E_0, a^I_0, s^E_1, s^I_1, a^E_1, a^I_1, \ldots, s^E_T, s^I_T) \]

As we mentioned in Section 2.2, AIRM makes use of state-action samples drawn from both the expert (given) and the currently learned data, which is used by the discriminator to compute a binary cross entropy (BCE) loss at each iteration. Let the expert’s demonstrations be \( X^E \) (as shown above) and trajectories drawn from the currently learned decentralized policy \( \pi \) be denoted as \( \hat{X} \). Samples are drawn from \( X^E \) and \( \hat{X} \) at every iteration and used to minimize the distance between the expert’s and learned state-action marginal distributions. The reward is updated using the discriminator’s confusion [11] and the updated reward is sent to the generator. The latter uses a decentralized generalization of PPO [29], which we call Dec-PPO, which follows the same CTDE architecture as MA-PPO [39].

Dec-PPO generalizes PPO from Section 2.2. The centralized critic network updates the value function as a squared-error loss:

\[
L_i^{VF}(\omega) = \left( V^\omega(s^I_t) - \hat{V}^{\text{argmax}}_i(s^I_t) \right)^2
\]

where \( \hat{V}^{\text{argmax}}_i \) is the discounted reward-to-go obtained during the corresponding episode at timestep \( t \) and \( V^\omega(s^I_t) \) is the predicted value of global state \( s^I_t \) (Section 3.1). Let,

\[
\lambda_i^t = \frac{\pi_{o,i}(a_{s^I_t}^t)}{\pi_{o,old}(a_{s^I_t}^t)} \quad \text{and} \quad \lambda_j^t = \frac{\pi_{o,j}(a_{s^I_t}^t)}{\pi_{o,old}(a_{s^I_t}^t)}
\]

Then the policy loss of the decentralized actor networks for the agents \( i \) and \( j \) respectively is given as:

\[
L_i^{\text{clip}}(\omega) = E_{(s^I_t, a_{s^I_t}^t) \sim \pi_{o,i}} \left[ \min \left( \lambda_i^t A^\alpha_{o,i}, \text{clip}(\lambda_i^t, 1 - \epsilon, 1 + \epsilon) A^\alpha_{o,i} \right) \right]
\]

\[
L_j^{\text{clip}}(\omega) = E_{(s^I_t, a_{s^I_t}^t) \sim \pi_{o,j}} \left[ \min \left( \lambda_j^t A^\alpha_{o,j}, \text{clip}(\lambda_j^t, 1 - \epsilon, 1 + \epsilon) A^\alpha_{o,j} \right) \right]
\]

The policy entropy loss is added to the clipped surrogate objective to promote exploration. This is given as:

\[
L_i^{\text{ENT}}(\omega) = \sigma H \left[ \pi_{o,i}(s^I_t) \right] \quad \text{and} \quad L_j^{\text{ENT}}(\omega) = \sigma H \left[ \pi_{o,j}(s^I_t) \right]
\]

where \( H \) is the policy entropy and \( \sigma \) is the entropy hyperparameter. The total loss for the two actors is then given as:

\[
L_i(\omega) = L_i^{\text{clip}}(\omega) + L_i^{\text{ENT}}(\omega)
\]

\[
L_j(\omega) = L_j^{\text{clip}}(\omega) + L_j^{\text{ENT}}(\omega)
\]

At the end of training, the discriminator returns the learned common reward function while the generator provides the converged vector of policies.
Algorithm 1: Dec-AIRL

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{Input:} $\mathcal{DM}$ sans $R$ and $T$; Exp trajs $X^E$.
\State \textbf{Output:} Learned common reward function $\hat{R}$.
\State Initialize decentralized policy $\pi$, Discriminator $D_\theta$.
\For {$\text{iter} \leftarrow 0 \text{ to } \text{train\_iters}$}
\State Generate joint trajectories $\hat{X}$ using $\pi$.
\State Sample joint $(s, a)$ pairs $\hat{y}, \hat{y}^E$ from $\hat{X}, X^E$, respectively.
\For {$ep \leftarrow 0 \text{ to } \text{discriminator\_epochs}$}
\State \text{Train discriminator $D_\theta$ via BCE to classify expert data $\hat{y}$ from learned policy data $\hat{y}$}.
\EndFor
\State Update reward $\hat{R} \leftarrow \log D_\theta(S, A, S') - \log(1 - D_\theta(S, A, S'))$.
\For {$ep \leftarrow 0 \text{ to } \text{generator\_epochs}$}
\State \text{Train generator $G(\hat{R}) \leftarrow$ Dec-PPO.}
\EndFor
\State Get updated policy $\pi^L \leftarrow G(\hat{R})$.
\EndFor
\State \textbf{Return} $\hat{R}, \pi^L$.
\end{algorithmic}
\end{algorithm}

3.3 Algorithm

The algorithm for Dec-AIRL (Algorithm 1) takes the model $\mathcal{DM}$ without the reward function and transition function (as Dec-AIRL is model free), the expert trajectories $X^E$ as input. The aim is to learn the common reward function for the task $\mathcal{R}$. It starts by generating a random decentralized policy vector $\pi$, and a discriminator $D_\theta$ with random weights is initialized. The algorithm begins its iterative updates until the end of training iterations (line 2). In each iteration, joint trajectories $\hat{X}$ of the agents are generated using the current policy vector $\pi$, followed by picking some state-action pairs $\hat{y}$ from $\hat{X}$ and some state-action pairs $\hat{y}^E$ from $X^E$ (line 4). Using these samples, the discriminator is trained to differentiate between the expert and learned sample data using BCE loss (line 6). From the trained discriminator, the updated reward $\hat{R}$ is then extracted in line 7. Next, the generator $G(\hat{R})$ is trained with a centralized critic and decentralized actors based Dec-PPO to generate the forward rollout vector of policies. Finally, the learned reward function $\hat{R}$ and converged policy $\pi^L$ are returned by the algorithm.

3.4 Human-Robot Collaborative Task Execution

Notice that in Algorithm 1 Dec-AIRL yields the learned reward function as well as the learned policy vector, $\pi^L$. From this vector, the policy of the agent being replaced by the robot is then used to control the robot in the collaboration. The environment state at each time step is perfectly observed through the robot’s sensors and the corresponding action from the decentralized robot policy is performed by the robot. This execution continues until the task is completed or a goal state is reached. To illustrate, in a simulated assistive domain, the robot’s policy would map the state obtained from the simulation to the target robot’s joint torques. Whereas for the real-world produce sorting domain, discrete state values consisting of the onion location, blemished or not, and the location of the human’s gloved hand is mapped to a high-level action, which is then performed by the robot. Section 4 explains the specifics in more detail.

4 EXPERIMENTS

We implemented Algorithm 1 in Python by extending an existing AIRL implementation [36]. Our codebase is freely available on GitHub at https://github.com/prasuchit/Dec-AIRL. We evaluate Dec-AIRL using two domains, one of which is a simulation while the other uses human agents and a physical collaborative robot.

4.1 Simulated Patient Assistance

Our formative evaluation domain is available within the simulated open-source assistive-gym [6], which consists of multiple environments that involve a cobot assisting a patient with basic needs such as bathing, scratching, feeding, and dressing. These environments are built on top of OpenAI’s gym testbed [5], and available in an MMDP format in version 0.1 and a Dec-MDP format in the latest version. For the purpose of this paper, we chose an environment that allows both the human and robot to act at every time step, and consists of both necessary and adverse interactions.

Setup. A Sawyer robotic arm (7 DOF cobot from Hahn Robotics previously Rethink Robotics) must safely and successfully feed a human, while the human also moves towards the spoon to receive it, as shown in Fig. 2. Each agent – robot and human – follow a learned policy to successfully complete the task because the start state of both agents is not fixed and they do not follow predetermined paths to reach their respective goals. Therefore, an agent’s action at every time step may be contingent on the other agent’s action.

The task leads to several interactions between the agents: a necessary interaction is where the robot brings the spoonful of food in contact with the human’s mouth and the human moves towards the robot to cooperate. On the other hand, the robot touching the human anywhere other than the mouth, either with the spoon or its arm, is considered an adverse interaction. The resulting penalty is proportional to the force of contact. Figure 2c shows an example of an adverse interaction where the robot touches the human near the eye with the spoon while following a poorly learned policy.

The robot and human state is composed of the following values respective to the agent: spoon: pose and orientation, human head: pose and orientation, spoon distance from human mouth, body joint angles, and the spoon force on the human. The dimension of actions for the human and robot are 4 and 7 respectively based on the number of mobile joints. The state $S$ in the Dec-MDP is a concatenation of the human’s and robot’s states.

Performance evaluation. For Dec-AIRL, we used 256 hidden nodes for the 2 hidden layers in both the $g$ and $h$ networks within the discriminator, a batch size of 128, learning rate for both the discriminator and the actors as 0.0003, discount factor and generalized advantage estimator (g) both as 0.95, max gradient normalization as 0.5, and a random seed between 1 and 100. For Dec-PPO, we used the same hyperparameters as mentioned in gSDE [25] because it has been tuned and tested on multiple pybullet environments. The default hyperparameters were used for Co-GAIL and MA-AIRL.

The expert trajectories were obtained from a trained RL model. We compared the performance of Dec-AIRL with two known baselines, MA-AIRL and Co-GAIL. All three algorithms were provided $10^6$ steps of expert data $X^E$ with each episode lasting a maximum of 200 time steps though usually less, and trained for $10^9$ iterations.
Figure 2: (a) Sawyer picking up the food from the table. (b) Feeding it to the human in the chair while the human also moves towards the spoon to cooperate. (c) The cobot encountering an adverse interaction by touching the human near the eye.

<table>
<thead>
<tr>
<th>Method</th>
<th># steps</th>
<th># adverse interactions</th>
<th>Total reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert</td>
<td>24.5</td>
<td>4</td>
<td>148.6</td>
</tr>
<tr>
<td>MA-AIRL</td>
<td>131.2</td>
<td>12</td>
<td>108</td>
</tr>
<tr>
<td>Co-GAIL</td>
<td>164.5</td>
<td>19</td>
<td>91.4</td>
</tr>
<tr>
<td>Dec-AIRL</td>
<td>35</td>
<td>7</td>
<td>143.2</td>
</tr>
</tbody>
</table>

Table 1: Dec-AIRL improves on both baseline methods significantly and is closest to the expert’s performance.

The forward RL step in MA-AIRL and Co-GAIL requires to be run on perfectly observed global state-action pairs, while the Dec-PPO in Dec-AIRL was run such that data that was observable to each agent only was provided to it. All three learned policies were deployed in a setting where the human state was perturbed using a small offset of 0.1 to depict a realistic perception through sensors like RGB-D cameras.

Table 1 compares the methods on the number of steps to complete the feeding, number of adverse interactions, and reward accrued per episode, averaged across 100 episodes. The results show that Co-GAIL and MA-AIRL converge to a joint policy that performs poorly in interacting states as compared to Dec-AIRL and this difference is statistically significant (Wilcoxon rank-sum test on reward accrued and number of adverse interactions, all p-values < 0.0005). Both MA-AIRL and Co-GAIL yield policies that prescribe actions conditioned on the global state. In other words, due to the choice of the underlying model (MMDP or Markov game), the robot is assumed to have access to the human’s joint angles and by extension the accurate pose and orientation of the human head; it must also assess the force experienced by the human due to the spoon. However, these may not be observed perfectly in practice by the robot leading to suboptimal actions. As Dec-AIRL yields a decentralized policy for the robot, which depends only on the parameters that it can observe, its performance is better aligned with the situation. Therefore, not only does Dec-AIRL generate less adverse interactions, it also accrues a better reward per episode compared to the baselines. Figure 3 shows the comparison between the Dec-AIRL policy and the baseline policies when the robot is close to the human’s mouth.

Figure 3: Comparison of the three learned policies implemented on an environment where the human’s pose is not perfectly observed. (a) Co-GAIL policy and (b) MA-AIRL policy reach close to the mouth but fail to accurately feed the human and also drop some food, while (c) Dec-AIRL policy takes a slightly longer path to the mouth than the expert but manages to feed the human correctly while avoiding adverse interactions.

4.2 Human-Robot Line Sorting

Our summative evaluation domain is a use-inspired human-robot line sorting task where the physical cobot UR3e (from Universal Robots) is tasked with sorting onions along with a human on a conveyor belt after observing a human-human team perform the task. A key challenge to a successful sort in the shared workspace is for both agents to avoid reaching for an onion on the table simultaneously or placing a good onion back on the table in tandem after inspection — we refer to these situations as adverse interactions.

We model the collaborative onion-sorting domain as a discrete Dec-MDP sans the reward function, which is learned. The factored local state for each agent is captured by 4 key variables yielding a total of 96 local states: Onion_location{unknown, on conveyor, hover location, in front of face}; EndEffector_location{on conveyor, hover location, in front of face, in bin}; Prediction{unknown, good, bad} and Interaction{true, false}. The value ‘hover location’ is a
region of space just on top of the conveyor and ‘in bin’ indicates just inside a bin that holds discarded onions. The interaction variable is activated for states where the other agent also reaches out simultaneously to pick or place an onion on the table. An example global state where the robot’s target onion is on the table, its end-effector is in the hover location, the onion prediction is bad, both the human’s onion and hand are in front of the face, the prediction is good, and the agents are not in an interaction state, would be represented as ((on conveyor, hover location, bad, false),(in front, in front, good, false)). Prior to detection, each onion’s location and status is unknown.

Each sorter performs one of six abstract actions in the Dec-MDP: Detect, which shifts the sorter’s focus to a new onion on the table and also gives a preliminary prediction; Pick up the onion which is self explanatory; Inspect the onion by rotating it and checking for blemishes; Place onion on the conveyor after it is picked and found to be unblemished; otherwise Place onion in the bin after it is picked; No-Operation to wait for one timestep in the same state. Both the agents have the same set of local state variables and local actions. The Dec-MDP model is implemented as an environment in MA-Gym for convenient dissemination.

Setup. Two depth cameras alongside each agent observe and record the other agent (human or robot) standing opposite (see Fig. 4). The time-synchronized, fused RGB-D frames from these cameras are quantized into appropriate state variables by SA-Net while a trained YOLOv5 deep neural network model is used to detect and classify the onions as blemished or not, which is not perfect. These state recognitions are shown in red text on the frames in Fig. 5. For Dec-AIRL, we used the same hyperparameters as mentioned in Section 4.1 except for the size of hidden nodes in the g and h networks, which we modified to 128 hidden nodes for the 2 hidden layers because the Dec-MDP model of agent sorting is smaller and discrete compared to the assistive domain.

Performance evaluation. We recorded 10 rounds of sorting by 4 different teams of 2 humans each. Each round is a sort of a batch of 10 onions. This yielded 40 trajectories with an average of 650 frames from the two cameras are time synchronized through ROS at each time step. All three methods were run for 25000 states from SA-Net trajectories were used to obtain the robot action from these policies and compared to the ground truth robot action.

Co-GAIL for various sets of input trajectories. One reason for this improvement is that MA-AIRL and Co-GAIL learn policies which prescribe actions that led to more adverse interactions compared to the actions from Dec-AIRL. The best policies yielded 153 interactions for Co-GAIL and 96 interactions for MA-AIRL compared to 27 for Dec-AIRL across 2,304 states of the sort. A key reason for this is that the human’s blemished onion assessment, which is available during training as part of the global state, cannot be observed during execution. It may lead the cobot to pick an onion thinking that the human will discard its picked onion into the bin. However, if the noisy assessment has mislabeled the onion being inspected by the human as blemished, the human actually places it back on the table leading to a collision (for illustration, see Fig. 7).
Table 2: Average precision and recall from 5 cobot-human sorting sessions with 10 onions sorted per session using policies learned via MA-AIRL and Dec-AIRL. The cobot-human team’s performance with Dec-AIRL policy improves on MA-AIRL performance in terms of recall. TP - True positive, FP - False positive, TN - True negative, FN - False negative.

<table>
<thead>
<tr>
<th>Method</th>
<th>(TP,FP,TN,FN)</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human-human</td>
<td>(4,1,5,0)</td>
<td>0.8</td>
<td>1.0</td>
</tr>
<tr>
<td>Human-cobot (MA-AIRL)</td>
<td>(3,2,3,2)</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>Human-cobot (Dec-AIRL)</td>
<td>(3,2,4,1)</td>
<td>0.6</td>
<td>0.75</td>
</tr>
</tbody>
</table>

As MA-AIRL exhibits a higher LBA compared to Co-GAIL, we selected MA-AIRL as the baseline for comparison with Dec-AIRL on the physical sorting. We deployed the best performing decentralized policy of one of the agents to control a physical cobot UR3e to guide collaboration with a human to sort faux onions. We ran 5 rounds of cobot-human sorting where lab members (not involved in this paper) played the role of the human sorter. Our evaluation focuses on how well the cobot sorts in the team and whether it avoids adverse interactions; the human sorters follow a fixed behavior in sorting the onions (see the supplementary file for more details). The cobot performs the sort by receiving the bounding boxes from YOLO v5, on which techniques such as central orthogonal projection, direct linear, and affine transforms are used to obtain the coordinates of the onions in its 3D workspace. The cobot and the human each pick up the onions and place them either in the bin or back on the conveyor table after inspection. We measure the team’s sorting performance using the domain-specific metrics of precision and recall where \( \text{precision} = TP/(TP + FP) \) and \( \text{recall} = TP/(TP + FN) \). Here, true positive (TP) is the count of blemished onions in the bin, false positive (FP) is the number of good onions in bin, true negative (TN) is the number of good onions remaining on table, and false negative (FN) is the count of blemished onions remaining on table. From Table 2, we note that Dec-AIRL leads to the cobot-human team exhibiting recall that is better than that of MA-AIRL, but still somewhat far out from the precision and recall of the human-human team, which indicates room for improvement.

Figure 7: Adverse interaction from a MA-AIRL learned policy. (a) Both the cobot and human are engaged in inspection. (b) The human decides to place the onion back on the conveyor because he assesses it as a good onion. Meanwhile, the cobot is also placing its onion back because it wrongly estimated the global state as it cannot see the onion from the human’s perspective. This particular onion had a blemish not visible while on the conveyor, which led the cobot to (erroneously) reason that the human will discard the onion. (c) Both agents place their onion on the conveyor thereby encountering an adverse interaction.

Figure 8 shows some frames from the cobot-human collaborative sort when the cobot is using the Dec-AIRL-learned policy.

5 RELATED WORK

Recently, few IRL techniques have targeted the problem of multi-agent coordination/cooperation. They model the problem either as a game-theoretic equilibrium, solve an MMDP with the assumption that all agents have access to the global state, or just consider the other agent(s) as part of the transition noise in the environment [23].

Multi-agent imitation learning. Imitation learning techniques that deal with multi-agent tasks include MAGAIL [52], CoDAIL [21], MA-DAAC [16], and Bayesian MA-DAAC [38]. These techniques extend GAIL to a multi-agent setup. MAGAIL and CoDAIL model the problem as a Markov game with Nash equilibrium as the convergence criteria. MA-DAAC aims to improve the scalability of multi-agent GAIL by representing the generator with an actor-attention-critic instead of the regular actor-critic and using a centralized or decentralized discriminator. All of these methods evaluate their approach on multi-agent particle environments (a simulated platform by OpenAI containing small multi-agent domains that do not involve adverse interactions) in both cooperative and competitive setups. Recently, Co-GAIL [35] extended InfoGAIL [19] to model human-robot interaction scenarios with multiple human strategies. They model it as an MMDP, where all agents have access to the global state, and assume that the latent feature, the human strategy, is shared between agents. Co-GAIL is implemented on Gibson’s simulated human-robot domains and the expert trajectories are recorded using Amazon turk. BTIL [30] uses a task model and agent model and infers the time-varying mental states of team members. It learns decentralized team policies from demonstrations of suboptimal teamwork. It also jointly learns their transition models. BTIL was evaluated on a single movers-and-packers domain, which is a relatively small and simulated platform. Although, the paper claims to learn a decentralized policy, their policy is a mapping of the global task state to agent action, which is not a fully decentralized representation. None of these techniques utilize a decentralized MDP as the underlying model of the task, which is better suited for modeling problems where an agent cannot perceive the global state.
They also do not evaluate the method on domains that require a human to actively interact with a robot.

**Multi-agent inverse RL.** In the IRL context, MA-AIRL [40], ME-AIRL [37], and MA-AIRL-Latent [13] extend AIRL to a multi-agent setup. MA-AIRL models the problem under logistic stochastic best response equilibrium (also known as the quantal response) and tests the method on multi-agent particle environments. MA-AIRL-Latent aims to improve the sample efficiency of MA-AIRL when the agents have shared latent variables, and evaluates on highway trajectories from the highD data set [18] and trajectories of aircraft in terminal airspace from non-public Federal Aviation Administration (FAA) data (also simulated domains). ME-AIRL uses an analytical method to connect the actor-critic model with AIRL and GAIL and measures the performance of the method on a simple single-state continuous game for 2 agents with one action dimension per agent. Older techniques such as Joint-Action-IRL [23] and BA-IRL [33] consider the other agent to be part of the environment as transition noise and model the IRL problem as a single-agent scenario. Joint-Action-IRL implements the method on a hand-finishing task where the human’s role is to refinish the surface of a box attached to an ABB industrial robot and the robot moves to aid the process. BA-IRL evaluates their method on a ROS/Gazebo simulation of two Turtlebots trying to sort balls on a table using attached manipulators. An early technique GSSG-IRL [26] models decentralized, non-cooperative multiple agents using IRL, but formulates the problem as a Nash equilibrium based general-sum stochastic game between the agents and tests the technique on a simple Gridworld domain only. The key difference between GSSG-IRL and Dec-AIRL is that in our case, the agents are cooperative and maximize a common system reward instead of working in self-interest. Additionally, we test our method on both a simulated continuous domain and a use-inspired discrete domain with human-robot collaboration scenarios that model adverse interactions.

6 CONCLUDING REMARKS

Motivated by the paradigm of learn-from-observation to learn human-robot collaborations, we presented a novel multi-agent IRL method that learns from human-human demonstrations to allow a cobot to work collaboratively with a human while avoiding adverse interactions within a shared workspace. We highlighted the importance of using a decentralized MDP to model the learning problem as compared to the previous game-theoretic approaches or using a multi-agent MDP model. Results show that the Dec-AIRL outperforms the previous multi-agent MDP based imitation learning method and a Markov game based IRL technique. Dec-AIRL also performs satisfactorily when compared with a human-human sorting team. We have also shown generalizability of our technique across two domains, one being continuous and simulated and the other being discrete and realistic. Thus, Dec-AIRL could pave the way for facilitating future cobot deployments in human-robot collaborative scenarios involving shared workspaces.

An avenue for future work is to learn type-contingent reward functions where the human teammate may be of varying types engaging in differing ways of completing the task based on her type. A type-contingent reward function will let the cobot adapt its behavior to the different types of teammates. Another possible extension of this work would be to model the human (demonstrators and/or co-worker(s)) as bounded rational instead of fully rational. This could be done by solving the model attributed to them approximately (to their level of reasoning) or by maintaining a model of the human that also factors into the policy training. Additionally, Dec-AIRL models a simultaneous action multi-agent scenario though the human and cobot complete some of their actions at different rates. As such, the Dec-AIRL may need to be generalized to reason with time-varying actions.

In certain cases, the expert (human) demonstrations may not be perfectly observable, or environmental obstacles may cause occlusions. Existing IRL methods use expectation-maximization or marginalization to deal with these problems. Alternatively, we could use human-in-the-loop techniques to receive clarifications or more demonstrations for the uncertain portions. Finally, there may be cases where the human teammate may exhibit divergent behavior than what was observed in the expert demonstrations. In such cases, one approach is to start with the policy learned using the IRL reward function as a starting point and perform online RL to accommodate these new changes progressively.

Some additional domains where Dec-AIRL could be applied include collaborative assembly tasks (e.g. furniture assembly), independent tasks within a shared workspace like an automation line, and emergency search-and-rescue scenarios. As such, Dec-AIRL can be extended and utilized in many other avenues to solve a variety of problems in human-robot collaboration in shared workspaces.
ACKNOWLEDGMENTS

This research is supported in part by NSF grant #IIS-1830421 and a Phase 1 grant from the GA Research Alliance. We thank the demonstrators and other experimenters for their assistance.

REFERENCES


