Adaptive Human-Robot Collaboration using Type-Based IRL

Prasanth Sengadu Suresh[1] **Prashant Doshi**[1] **Bikramjit Banerjee**[2]

[1]THINC Lab, School of Computing, University of Georgia, 200 D. W. Brooks Drive, Athens, GA 30602, USA
[2]School of Computing Sciences & Engg., University of Southern Mississippi, 118 College Dr., Hattiesburg, MS 39406, USA

Abstract

Human-robot collaboration (HRC) integrates the consistency and precision of robotic systems with the dexterity and cognitive abilities of humans to create synergy. However, human performance may degrade due to various factors (e.g., fatigue, trust) which can manifest unpredictably, and typically results in diminished output and reduced quality. To address this challenge toward successful HRCs, we present a human-aware approach to collaboration using a novel multi-agent decision-making framework. *Type-based* decentralized Markov decision processes (TB-DecMDP) additionally model latent, causal decision-making factors influencing agent behavior (e.g., fatigue), leading to dynamic agent types. In this framework, agents can switch between types and each maintains a belief about others' current type based on observed actions while aiming to achieve a shared objective. We introduce a new inverse reinforcement learning (IRL) algorithm, TB-DecAIRL, which uses TB-DecMDP to model complex HRCs. TB-DecAIRL learns a type-contingent reward function and corresponding vector of policies from team demonstrations. Our evaluations in a realistic HRC problem setting establish that modeling human types in TB-DecAIRL improves robot behavior on the default of ignoring human factors, by increasing throughput in a human-robot produce sorting task.

1 INTRODUCTION

Advances in agent-based decision making are making robots that can take decisions under uncertainty increasingly real. Such intelligence allows platforms such as collaborative robots (cobots) to work safely with humans to effectively contribute to several aspects of human endeavors. Such human-robot collaboration (HRC) enables a paradigm shift in key domains such as healthcare, manufacturing, and other industries by combining the diverse complementary strengths of human and robotic agents to optimize task efficiency and throughput.

To optimize HRC, cobots must learn to adapt to the dynamic latent factors that influence human action choice. One way to capture these factors in learned behavior is to let (expert) humans perform (demonstrate) the task as a team. Then, we may learn the team's underlying preferences for task state and joint actions using inverse reinforcement learning (IRL) [Ng and Russell, 2000], which is known to *generalize beyond the demonstrations.* Whereas IRL has found success in several single-expert tasks [Arora and Doshi, 2021], it remains underexplored in multiagent tasks in contexts such as HRC. Prior work in IRL applied to HRC [Nikolaidis, 2014, Suresh et al., 2023] assumes that the human teammate follows fixed rule-based behavior throughout, and fails to account for the causal latent factors influencing human behavior such as biases or fatigue.

To illustrate, consider a packing shed where humans stand across each other and sort onions on a conveyor. The *optimal* sortation discards visibly blemished onions and closely inspects the seemingly unblemished ones before deciding to discard or return them back to the conveyor. Prolonged periods of sorting leads to fatigue, resulting in a decline in sorting speed. Fatigued workers usually take a short break to regain energy. In this HRC use case, a cobot could recognize and adapt to its fatigued teammate by accelerating its sorting speed to maintain overall throughput, This 'industrial' mode should be maintained by the cobot only while the human is fatigued and away from the shared workspace, as the increased speed may not be safe for humans.

We make **three** key contributions in this paper:

1. We present a novel multiagent decision-making framework, TB-DecMDP, which models dynamic agent types while solving for type-contingent decentralized agent behavior. Agents may detect their teammate's current

type and switch their behavior accordingly to maximize team reward.

2. In the context of TB-DecMDP, we present a new IRL method, TB-DecAIRL, which learns a *type-contingent team reward function* and a corresponding policy vector (one for each agent). This output, backed by a theoretical performance guarantee, enables the cobot to adapt to the human agent's dynamic type, promoting seamless HRC in shared workspaces. A type-contingent reward function is novel to IRL.
3. Finally, we empirically establish the efficacy of TB-DecAIRL towards successful HRC in simulation and validate it on a produce-sorting task using a physical HRC with a UR3e cobot.

By demonstrating enhancement in collaborative efficacy via better average rewards per episode (on the MAGym [Koul, 2019, Brockman et al., 2016] simulated environment), and increased throughput compared to the baseline (on the physical task), we establish the value of TB-DecAIRL over the default of ignoring human factors. We conclude by providing avenues for future work.

2 BACKGROUND

Multiagent IRL models the expert using a variant of Markov decision process [Puterman, 1994], which it solves optimally. Since HRC is collaborative and decentralized (i.e., the robot may not perfectly observe all attributes of the human's state, such as the human's joint angles) by definition, a Dec-MDP [Goldman and Zilberstein, 2003] is appropriate to model the expert.

A two-agent Dec-MDP can be formally defined as a tuple $\mathcal{DM} \triangleq \langle S, A, T, R \rangle$ where the joint state, $S = S_i \times S_j$. Here, S_i and S_j are the locally observed states of the two agents i and j, respectively, which when combined yield the joint state of the system; $A = A_i \times A_j$ is the set of joint actions of the two agents; $T : S \times A \times S \rightarrow [0, 1]$ is the transition function of the multiagent system; and $R : S \times A \rightarrow \mathbb{R}$ is the common reward function.[1] In IRL, the latter is unknown, whereas the rest are usually known. As such, the agents know their local state and any common task attributes and act independently while optimizing a common reward [Melo and Veloso, 2011]. Let $\mathcal{X}^E$ be the set of expert demonstrations and a complete trajectory $X^E \in \mathcal{X}^E$ is given by,

$$X^E = (\langle s_i^0, s_j^0 \rangle, \langle a_i^0, a_j^0 \rangle, \cdots, \langle s_i^{\mathcal{T}}, s_j^{\mathcal{T}} \rangle, \langle a_i^{\mathcal{T}}, a_j^{\mathcal{T}} \rangle). \quad (1)$$

The agent's task is to learn a reward function and a policy profile, $\boldsymbol{\pi} = \langle \pi_i, \pi_j \rangle$, that optimizes its return, such that the trajectories generated by $\boldsymbol{\pi}$ are indistinguishable from those in $\mathcal{X}^E$.

[1]This Dec-MDP describes a locally fully observable model whose local states when combined yield the fully observable joint state [Goldman and Zilberstein, 2003].

2.1 DECENTRALIZED ADVERSARIAL IRL

Decentralized adversarial IRL (Dec-AIRL) [Suresh et al., 2023] generalizes the single-expert deep-IRL method – adversarial IRL (AIRL) [Fu et al., 2018] (which works on the principle of maximum causal entropy [Ziebart, 2010, Gleave and Toyer, 2022]) – to learn a common reward function for the team from expert demonstrations. AIRL uses a discriminator $D_{\boldsymbol{\alpha}}(X)$ to learn a function $f_{\boldsymbol{\alpha}}(X)$ [Fu et al., 2018] which at convergence approximates the expert policy's advantage function. Dec-AIRL analytically represents the discriminator as $D_{\boldsymbol{\alpha}}(X) = \frac{e^{f_{\boldsymbol{\alpha}}(X)}}{e^{f_{\boldsymbol{\alpha}}(X)} + \pi(X)}$ and the reward is updated as

$$R_{\boldsymbol{\alpha}}(X) \leftarrow \log D_{\boldsymbol{\alpha}}(X) - \log(1 - D_{\boldsymbol{\alpha}}(X)). \quad (2)$$

When simplified, Eq. 2 yields $f_{\boldsymbol{\alpha}} - \log(\pi)$, which is the entropy-regularized reward formulation. In the underlying Dec-MDP, each agent only has access to their local state and some general task attributes. Dec-AIRL uses Dec-PPO - a decentralized generalization of the popular RL method - Proximal Policy Optimization (PPO) [Schulman et al., 2017], for forward-rollout. Dec-PPO uses the centralized training, decentralized execution paradigm where the centralized critic network updates its value function as a squared-error loss: $L_t^{VF}(\omega) = \left(V^{\pi_{\boldsymbol{\omega}}}(s^t) - \hat{V}_t^{targ}\right)^2$ where $\hat{V}_t^{targ}$ is the per-episode discounted reward-to-go and $V^{\pi_\omega}(s^t)$ is the predicted value of joint state s^t and ω is the vector of policy parameters (weights). We consider a dyadic system with agents i and j, although our formalism and method conceptually scale to N agents and is not limited to a dyad. The policy loss of agent i is given by

$$L_i^{CLIP}(\omega) = \mathop{\mathbb{E}}_{\pi_{\omega,i}} \left[min\Big(\lambda_i \, A^{\boldsymbol{\pi}}, clip(\lambda_i, 1-\epsilon, 1+\epsilon) \, A^{\boldsymbol{\pi}}\Big)\right],$$

where $\lambda_i = \dfrac{\pi_{\omega,i}(a_i|s_i)}{\pi_{\omega,i}^{old}(a_i|s_i)}$ is the importance sampling ratio.

$L_i^{CLIP}(\omega)$ provides a pessimistic bound over the final objective by using a surrogate objective that picks the minimum of the clipped and unclipped objectives. By clipping the importance sampling ratio, the incentive of moving λ outside the interval $[1-\epsilon, 1+\epsilon]$ is reduced. Advantage $A^{\boldsymbol{\pi}}$ is calculated using the reward estimate $R_{\boldsymbol{\alpha}}(X)$ from Eq. 2. This clipped surrogate objective, combined with the policy entropy, handles the explore-exploit dilemma. The policy entropy loss and corresponding total loss are given as: $L_i^{ENT}(\omega) = \sigma H\Big[\pi_{\omega,i}(s_i)\Big]$, $L_i(\omega) = L_i^{CLIP}(\omega) + L_i^{ENT}(\omega)$, where H is the policy entropy and σ is the entropy hyperparameter. These loss functions apply analogously for agent j. On convergence, the discriminator and the generator return the learned common reward function and the vector of policies, respectively.

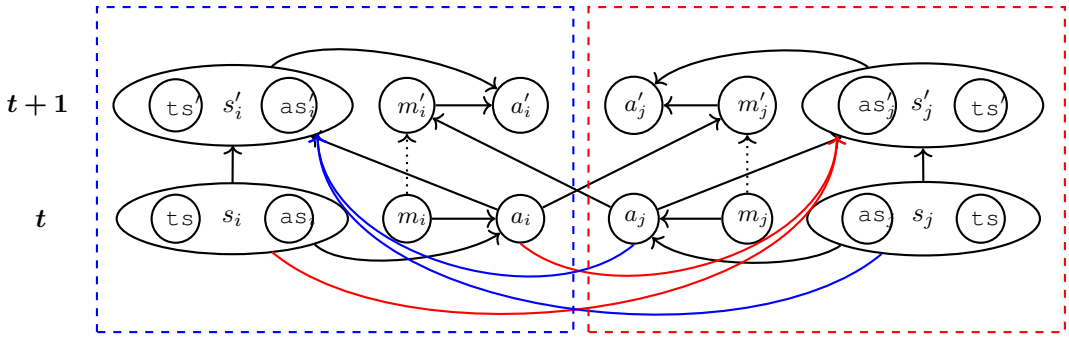


Figure 1: TB-DecMDP graphical model of a dyadic team with agents i (blue) and j (red), for two timesteps $\boldsymbol{t}$ and $\boldsymbol{t+1}$. Local state of agent i (s_i) is a combination of i's private attributes $\texttt{as}_i$ and common task attributes $\texttt{ts}$. Model m_i holds i's current belief over the others' type θ_j. The dotted link updates model m_i using the other agent's action at t. These apply analogously for agent j. All agents transition jointly to the next state, as indicated by the dependence (colored links) of each agent's next state on the other's previous state and action.

3 TYPE-BASED DECENTRALIZED AIRL FOR HUMAN-ROBOT TEAMING

In this section, we formalize a novel multiagent model, Type-Based Dec-MDP (TB-DecMDP), and present our novel IRL method, Type-Based Dec-AIRL (TB-DecAIRL) to learn a type-contingent reward function and the vector of policies (one for each agent), which reason about the other agents' behavioral changes and adapts accordingly.

We model realistic human-robot collaborative settings where agents adjust their behavior based on latent factors (e.g. trust, fatigue), and anticipate and adapt to others' changes to achieve a shared goal. Although we present this work in the context of HRC, it extends to domains where autonomous agents must adapt to changing human conditions (e.g. healthcare and smart homes). For instance, in healthcare, such agents provide personalized care by monitoring patient conditions and intervening as and when needed. In smart homes, agents adjust the air conditioning system according to human preferences, highlighting the need for continuous monitoring and adaptation.

3.1 TYPE-BASED COLLABORATION MODEL

In addition to shared task attributes, each agent i has its private attributes (which may include mental attributes and thus are not observable by other agents). Each agent also maintains a model of all other agents, including a belief about their joint types. Agent i's initial belief is based on prior knowledge of these types. The combination of i's private attributes and the common task attributes constitutes i's local state (s_i). Agent i makes decisions based on s_i and its belief about the other agents' types. At each timestep, agent i observes other agents' actions noisily and updates its belief accordingly (see Fig. 1). All agents collaborate in a decentralized manner to optimize a common task-centric reward. Since each agent has perfect access to its own local state and only maintains a belief about the other agents' types, our TB-DecMDP can be considered a decentralized variant of a mixed-observability MDP [Ong et al., 2010].

A TB-DecMDP is formally defined as:

$$\mathcal{TB}-\mathcal{DM} \triangleq \left\langle Ag, S, A, T, R, M \right\rangle$$

- Ag represents the set of all agent identifiers within the system. Each agent is uniquely identified, allowing for individual tracking and interaction within the environment. The total number of agents in the system is denoted by N, where $N = |Ag|$.
- The joint state space S is defined as the Cartesian product of the individual local states of all agents, i.e., $S = S_1 \times S_2 \times \ldots \times S_N$. The local state S_i is further decomposed into:
  - The *task-state* $\texttt{tS}$, which is common across all agents. It encapsulates the shared aspects of the environment or task that all agents are aware of and interact with.
  - The *agent-specific state* $\texttt{aS}_i$, containing private attributes unique to agent i. Note that $\texttt{aS}_i$ is not observable by others. This includes $\theta \in \Theta_i$: The *type* of agent i, representing its inherent characteristics, capabilities, or roles within the system.

  Then, $S_i = \texttt{tS} \times \texttt{aS}_i$, allowing each agent to maintain both shared and private information.
- The joint action space A is the Cartesian product of the individual action sets of all agents, expressed as $A = A_1 \times A_2 \times \ldots \times A_N$. Each agent i has a local action set A_i, which includes all actions available to that agent.
- The state transition function $T : S \times A \times S \rightarrow [0, 1]$ defines the probability of transitioning from one joint state to another, given a particular joint action.

- It is important to note that an agent type $\theta \in \Theta_i$ transitions *exogenously*, meaning external factors govern its transition and this is not influenced by the state transition function T. This separation ensures that while agents can act and influence the environment, their inherent types remain consistent unless altered by external dynamics.
- The reward function $R : S \times A \rightarrow \mathbb{R}$ assigns a real-valued reward to each state-joint action pair. This reward is *common* to all agents, implying that it reflects a shared objective or goal that all agents are collectively trying to optimize.
- The set of agent models $M = M_1 \times M_2 \times \ldots \times M_N$ encapsulates the internal representations and beliefs each agent holds about the other agents.
- Each agent i's model M_i is defined as: $M_i = \Big(\Theta_{-i},\ b_i,\ F_i\Big)$. This includes:
  - $\Theta_{-i} = \prod_{j \neq i} \Theta_j$ represents the combined set of types for all agents except i.
  - $b_i \in \Delta(\Theta_{-i})$ denotes agent i's current belief distribution over the types of other agents.
  - $F_i : \Theta_{-i} \times A_{-i} \times \Theta_{-i} \rightarrow [0, 1]$ defines the probabilistic transition of others' types in the next time step given observed action and current types.

This formulation can be broadly seen as an *ex interim* expected utility formalism of Bayesian games [Shoham and Leyton-Brown, 2008] where each agent has perfect knowledge of its own type and has a mixed strategy of the others' type. For a dyadic team with agents i and j, the joint policy is given as $\boldsymbol{\pi} = \langle \pi_i, \pi_j \rangle$ where $\pi_i : S_i \times \Delta(\Theta_{-i}) \rightarrow A_i$.

3.2 IRL FROM TEAM DEMONSTRATIONS

The belief update equation for agent i in a dyad is given as:

$$\begin{aligned} b_i'(\theta_j' \mid a_j, b_i) &= \beta \sum_{\theta_j \in \Theta_j} \Pr(\theta_j' \mid a_j, \theta_j) \Pr(\theta_j) \\ &= \beta \sum_{\theta_j \in \Theta_j} F_i(\theta_j' \mid a_j, \theta_j)\, b_i(\theta_j). \end{aligned} \tag{3}$$

The experts' joint demonstrations contain state-action pairs as defined in (1) and a single belief trajectory corresponding to the expert trajectory:

$$\hat{b}^E_{\boldsymbol{\theta}} = (\langle \theta_i^0, \theta_j^0 \rangle, \langle \theta_i^1, \theta_j^1 \rangle, ..., \langle \theta_i^{\mathcal{T}}, \theta_j^{\mathcal{T}} \rangle \rangle). \tag{4}$$

The belief update of Eq. 3 and generation of the trajectory above occur in the belief module of TB-DecAIRL's architecture, which is shown in Fig. 2. For implementing this module, we use a GRU-cell [Cho et al., 2020] as it has been empirically demonstrated to have an equivalent representation as the analytical update.

The discriminator takes in the pooled states, pooled actions, and pooled types to distinguish expert and learned samples. This discriminator is then used to obtain the common task-centric reward as defined in Eq. 2. The (joint) discriminator optimization objective is now defined as:

$$\begin{aligned} &\mathcal{D}_{KL}(\rho_{\boldsymbol{\pi}_\omega}(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a}) \parallel \rho_{\mathcal{X}^E}(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a})) \approx \\ &\max_{\boldsymbol{\alpha}}\ \mathbb{E}_{(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a}) \sim \mathcal{X}^E, \hat{b}^E_{\boldsymbol{\theta}}} \Big[\log D_{\boldsymbol{\alpha}}(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a})\Big] + \\ &\mathbb{E}_{(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a}) \sim \boldsymbol{\pi}_\omega} \Big[\log\Big(1 - D_{\boldsymbol{\alpha}}(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a})\Big)\Big]. \end{aligned} \tag{5}$$

where ρ gives the occupancy measure as in AIRL. Note that divergence $\mathcal{D}_{KL}$ is dependent on the belief module parameters $\boldsymbol{\phi}$ through its dependence on the belief states.

Then, the IRL's objective is to optimize the policy vector, which is written as:

$$\min_{\omega} \mathcal{D}_{KL} \approx \min_{\omega} \max_{\boldsymbol{\alpha}} \mathbb{E}_{(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a}) \sim \mathcal{X}^E, \hat{b}^E_{\boldsymbol{\theta}}} \Big[\log D_{\boldsymbol{\alpha}}(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a})\Big] + \tag{6}$$

$$\mathbb{E}_{(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a}) \sim \boldsymbol{\pi}_\omega} \Big[\log\Big(1 - D_{\boldsymbol{\alpha}}(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a})\Big)\Big].$$

During the forward rollout (PPO) stage, the belief over other agents' type is obtained from the belief module at each timestep and factored into each agent's policy learning to learn a vector of policies that capture the expert's behavioral preferences. The gradient for policy optimization is given by:

$$\begin{aligned} \nabla_\omega \mathcal{D}_{KL} \approx \nabla_\omega \big[\mathbb{E}_{\boldsymbol{\pi}_\omega} \big(&\log D_{\boldsymbol{\alpha}}(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a}) \\ &- \log\left(1 - D_{\boldsymbol{\alpha}}(\boldsymbol{s}, b_{\boldsymbol{\theta}}, \boldsymbol{a})\right)\big)\big]. \end{aligned}$$

Given fixed belief parameters ($\boldsymbol{\phi}$), the required gradient for policy optimization for agent i is obtained as:

$$L_i^{CLIP}(\omega) = \mathop{\mathbb{E}}_{\pi_{\omega,i}} \Big[min\Big(\lambda_i\, A^{\boldsymbol{\pi}}, clip(\lambda_i, 1-\epsilon, 1+\epsilon)\, A^{\boldsymbol{\pi}}\Big)\Big]$$

where $\lambda_i^t = \dfrac{\pi_{\omega,i}(a_i^t | s_i^t, b_i^t)}{\pi^{old}_{\omega,i}(a_i^t | s_i^t, b_i^t)}$,

and analogously for agent j.

The advantage function is computed as $A^{\boldsymbol{\pi}_\omega} = Q^{\boldsymbol{\pi}_\omega} - V^{\boldsymbol{\pi}_\omega}$ with the action value-function $Q^{\boldsymbol{\pi}_\omega}$ given by $Q^{\boldsymbol{\pi}_\omega} = \mathbb{E}_{\boldsymbol{\pi}_\omega}\Big[\sum_{t'=t}^{\infty} \gamma^{t'-t}\Big(\log D_{\boldsymbol{\alpha}}(\boldsymbol{s}^{t'}, b^{t'}_{\boldsymbol{\theta}}, \boldsymbol{a}^{t'}) - \log\Big(1 - D_{\boldsymbol{\alpha}}(\boldsymbol{s}^{t'}, b^{t'}_{\boldsymbol{\theta}}, \boldsymbol{a}^{t'})\Big)\Big)\Big]$ and the state-value function $V^{\boldsymbol{\pi}_\omega}$ is given as $V^{\boldsymbol{\pi}_\omega}(b) = V(\boldsymbol{s}, b_{\boldsymbol{\theta}}) = \max_{\alpha \in \Gamma_{\boldsymbol{\theta}}(s)} (\alpha \times b_{\boldsymbol{\theta}})$ where $\alpha = \langle V(\boldsymbol{s}, \theta_1), V(\boldsymbol{s}, \theta_2), V(\boldsymbol{s}, \theta_3) \ldots V(\boldsymbol{s}, \theta_n) \rangle$.

3.3 ALGORITHM

The TB-DecAIRL algorithm (Algorithm 1) uses the new model $\mathcal{TB} - \mathcal{DM}$ without the reward and transition functions (as TB-DecAIRL is model-free) and the expert trajectories $\mathcal{X}^E$ (see Eq. 1) to learn the task's common reward function R. It starts by generating a random decentralized policy vector $\boldsymbol{\pi^L}$ with generator G_ω (line 1), loading the

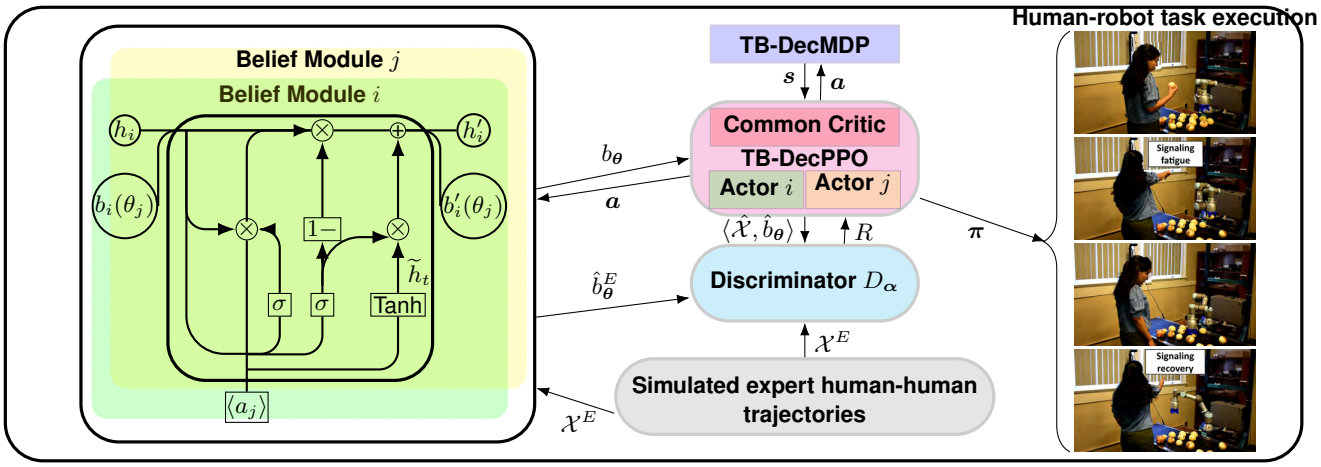


Figure 2: The TB-DecAIRL architecture for a dyadic team with agents i and j, and simulated human-human expert trajectories ($\mathcal{X}^E$). Agent i's belief module uses agent j's actions to generate belief states of j's type, and vice versa, creating joint $\hat{b}^E_{\boldsymbol{\theta}}$ trajectories. Similarly, TB-DecPPO interacts with TB-DecMDP, the type-based reward function R, and the belief module to generate $\langle \hat{\mathcal{X}}, \hat{b}_{\boldsymbol{\theta}} \rangle$. Both $\langle \mathcal{X}^E, \hat{b}^E_{\boldsymbol{\theta}} \rangle$ and $\langle \hat{\mathcal{X}}, \hat{b}_{\boldsymbol{\theta}} \rangle$ train the discriminator $D_{\boldsymbol{\alpha}}$ to update R until convergence. The learned policy $\boldsymbol{\pi}$ is then applied in HRC, where the robot follows its learned policy and the human continues to perform as previously in the demonstration. Here, h denotes the hidden state of the GRU and b denotes the normalized belief.

pre-trained belief module $B_{\boldsymbol{\phi}}$, and initializing the discriminator $D_{\boldsymbol{\alpha}}$ with random weights $\boldsymbol{\phi}$ and $\boldsymbol{\alpha}$ (see Fig. 3). $\hat{b}^E_{\boldsymbol{\theta}}$ is then obtained by passing X^E through $B_{\boldsymbol{\phi}}$ to obtain expert belief trajectories (see (4)). This is equivalent to performing a belief update at each timestep as per Eq. 3 using the expert state-action trajectories to obtain the expert's belief states at each timestep.

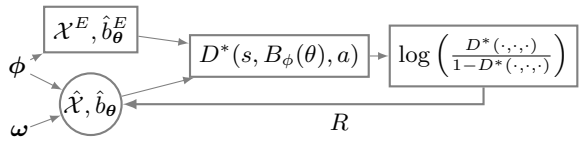


Figure 3: Stochastic computation graph for the expectation: $\mathbb{E}_{\mathcal{X}^E, \hat{b}^E_{\boldsymbol{\theta}}}\left[\log D^*\right] - \mathbb{E}_{\boldsymbol{\pi}, \hat{b}_{\boldsymbol{\theta}}}\left[\log(1 - D^*)\right]$ where D^* represents the maximum of $D_{\boldsymbol{\alpha}}$. Notice that both the policy parameters ($\boldsymbol{\omega}$) and the belief parameters ($\boldsymbol{\phi}$) influence the joint-state, joint-action trajectories and belief trajectories ($\hat{b}_{\boldsymbol{\theta}}$) through environment interaction. Circles represent stochastic nodes, rectangles represent deterministic nodes.

The algorithm iterates through updates until training concludes (line 2). In each iteration, it generates joint trajectories $\langle \hat{\mathcal{X}}, \hat{b}_{\boldsymbol{\theta}} \rangle$ using the current policy vector $\boldsymbol{\pi}$ and belief module $B_{\boldsymbol{\phi}}$ (line 3). (State, belief-state, action)-tuples are then sampled from these trajectories and from $\langle \mathcal{X}^E, \hat{b}^E_{\boldsymbol{\theta}} \rangle$ (line 5). The discriminator is trained to distinguish between expert and learned samples using BCE loss (line 7). The updated reward R is extracted from the trained discriminator (line 8). The generator $G_{\boldsymbol{\omega}}(R)$ is then trained with a centralized critic and decentralized actors using TB-DecPPO to produce the policy rollout vector. Finally, the learned reward function R and the converged policy $\hat{\boldsymbol{\pi}}^L$ are returned.

Algorithm 1: TB-DecAIRL

Input: $\mathcal{TB} - \mathcal{DM}$ sans R and T; Exp trajs $\mathcal{X}^E$ sans other agent types.
Output: Learned joint type-based reward function R.

1 Initialize generator ($G_{\boldsymbol{\omega}}$) with policy vector $\boldsymbol{\pi}^{\boldsymbol{L}}$, Discriminator $D_{\boldsymbol{\alpha}}$, and pre-trained belief module $B_{\boldsymbol{\phi}}$.
2 **for** $iter \leftarrow 0$ **to** $train_iters$ **do**
3 Use $\boldsymbol{\pi}^{\boldsymbol{L}}$ and $B_{\boldsymbol{\phi}}$ to step through the environment and generate joint trajectories $\langle \hat{\mathcal{X}}, \hat{b}_{\boldsymbol{\theta}} \rangle$
4 Obtain expert state-action and belief trajectories tuple $\langle \mathcal{X}^E, \hat{b}^E_{\boldsymbol{\theta}} \rangle$ by passing $\mathcal{X}^E$ through $B_{\boldsymbol{\phi}}$
5 Sample joint $(\boldsymbol{s}, b_{\boldsymbol{\theta}}, a, \boldsymbol{s}', b'_{\boldsymbol{\theta}})$ pairs from $\langle \hat{\mathcal{X}}, \hat{b}_{\boldsymbol{\theta}} \rangle$ and $\langle \mathcal{X}^E, \hat{b}_{\boldsymbol{\theta}} \rangle$, respectively
6 **for** $ep \leftarrow 0$ **to** $discriminator_epochs$ **do**
7 Train discriminator $D_{\boldsymbol{\alpha}}$ via BCE loss to classify $\langle \mathcal{X}^E, \hat{b}^E_{\boldsymbol{\theta}} \rangle$ from $\langle \hat{\mathcal{X}}, \hat{b}_{\boldsymbol{\theta}} \rangle$
8 Update reward: $R \leftarrow \log(D_{\boldsymbol{\alpha}}(...)/(1 - D_{\boldsymbol{\alpha}}(...))$
9 **for** $ep \leftarrow 0$ **to** $generator_epochs$ **do**
10 Train generator $G_{\boldsymbol{\omega}}(R) \leftarrow$ TB-Dec-PPO.
11 Get updated policy $\boldsymbol{\pi}^{\boldsymbol{L}} \leftarrow G_{\boldsymbol{\omega}}(R)$.
12 **return** R, $\boldsymbol{\pi}^{\boldsymbol{L}}$

3.4 THEORETICAL ANALYSIS

Type transition kernel $F_i(\theta'_j | a_j, \theta_j)$ forms a Markov chain with state θ_j and edges guarded by a_j. Under the assumption that F_i is irreducible and aperiodic, the type distribution b_i

given by Eq. 3 will converge to a limiting distribution. Let the joint beliefs, $b^t(\boldsymbol{\theta}) = \prod_i b_i^t(\boldsymbol{\theta}_{-i})$. Then, after sufficient time t elapses, $\mathcal{D}_{TV}(b^{t+1}, b^t) \leq \delta \cdot \mathcal{D}_{TV}(b^t, b^{t-1})$ where $0 \leq \delta < 1$ and $\mathcal{D}_{TV}$ denotes the total variation distance. Then, the following holds (proof is in the Appendix):

Theorem 1. *If the i-th agent's discriminator error compared to the i-th expert is small, that is, $\|D_i^t - D_i^E\| \leq \epsilon \; \forall i = 1, \ldots, N$, then the difference in conditional log-likelihood (LL) of data is bounded:*

$$LL(\mathcal{X}|R^E) - LL(\mathcal{X}|R^t) \leq \frac{8N\epsilon}{1 - \gamma(1 - \delta/2)},$$

where R^E and R^t are the true (expert) and learned common *reward functions at iteration t.*

As adversarial inverse learning algorithms have a convergence rate of $\mathcal{O}\left(\frac{1}{(1-\gamma)^3\sqrt{t}}\right)$ [Guan et al., 2021], it follows that as ϵ decreases at that rate, the average error in log-likelihood approaches 0.

4 EXPERIMENTS

We implemented TB-DecAIRL in Python and evaluate its performance on a use-inspired human-robot collaborative onion sorting domain. Our implementation of the method is available at `https://github.com/thinclab/TB-Dec-AIRL`. The objective is to have a human and a UR3e cobot stand across each other and sort onions on a line conveyor. In this collaborative produce sorting domain, the optimal sorting behavior involves quickly assessing each onion on the conveyor. If it is blemished, it should be picked up and discarded into a bin. If it appears unblemished, it should be picked up for a closer inspection. If it is still seen as unblemished, it is returned to the conveyor; otherwise, it is discarded into the bin. Both the human and the cobot operate in a shared workspace and work in a decentralized manner, while the cobot must adapt to changes in the human's sortation behavior due to fatigue.

4.1 SIMULATION IN MA-GYM

The simulated environment for the collaborative sorting domain was developed as a discrete state-action domain in MA-Gym [Koul, 2019] based on domain knowledge [Suresh et al., 2023].[2] In this environment, each agent's state contains 5 discrete variables: *Onion location* (takes one of 4 values based on the current onion location); *End-effector location* (takes one of 4 values based on the current end-effector location); *Prediction* (takes one of 3 values: blemished, unblemished, or unknown prediction label of the onion in focus); *Self-type* (the subject agent's type); *Indication* (true if the subject agent's type change has been communicated otherwise false).

Each agent has 9 discrete actions: *No-op* (no operation), *Detect* (choose any onion on the conveyor), *Pick* (pick up the chosen onion from the conveyor), *Detect-pick* (combined action to choose any onion on the conveyor and immediately pick it up), *Inspect* (inspect the picked onion), *PlaceOnConveyor* (place the held onion back on the conveyor), and *PlaceInBin* (place the held onion in the discard bin). The last two actions for the human are *Thumbs-up* (gesture indicating unfatigued type to the robot), and *Thumbs-down* (indicating fatigued type to the robot). Similarly, the robot's last two actions are *Speed-up* (increase movement speed to maintain throughput and enter industrial mode) and *Slow-down* (reduce movement speed to return to regular collaborative mode).

Expert trajectories were recorded using a hand-coded policy vector derived from observing real human-human team demonstrations, which was run in the MA-Gym environment. In the demonstration, one of the humans in the human-human team rests every so often and the other human speeds up during this time to maintain throughput. Otherwise, both humans sort the onions simultaneously as described previously.[3] This allowed us to repeatedly generate a large number of trajectories from different start states. We used a total of 10^6 timesteps (same for the baseline) and trained the methods for 10^9 iterations. Rest of the hyperparameters and training error are provided in the Appendix.

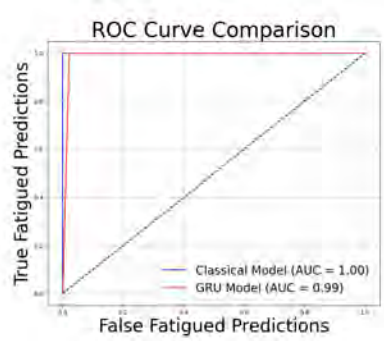
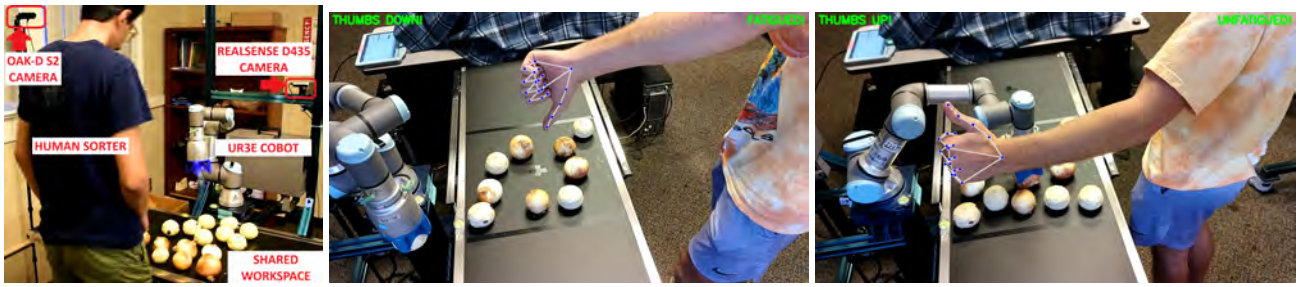


Figure 4: The receiver operating characteristic (ROC) plot comparing the pretrained GRU module predictions, a classical Bayesian belief module predictions, and the ground truth, for 100 episodes.

Each agent's belief update is represented using a pre-trained GRUcell, which is trained with ground-truth labels from the Gym environment and hand coded policy actions. It learns to predict what could be the next type of the other based on the observed action, not *when* the type will transition. We use

[2] It is available for download at `https://github.com/prasuchit/ma-gym/tree/master/ma_gym/envs/dec_huro_sorting`.

[3] These policies can be accesssed at `https://github.com/thinclab/TB-Dec-AIRL/blob/main/utils/tb_sorting_simulated_policy.py`.



(a) (b) (c)

Figure 5: (a) HRC sorting setup with a Realsense D435 behind the robot detecting objects on the conveyor and the OAK-D S2 camera to the left of the human monitoring their actions. (b) and (c) a human sorter signaling fatigue and recovery to the robot.

tanh activation, 64 hidden nodes, Adam optimizer, and cross-entropy loss for each GRUcell. We use batch training until convergence with data collected from complete episodes. Each episode is reset after an arbitrary limit of 100 timesteps. As shown in Fig. 4, upon convergence, the GRU model is near-identical to a classical Bayesian belief update.

Performance comparison with default. We use a reward function for evaluation which assigns a $+1$ reward for successfully sorting an onion and a -1 penalty for incorrect sorting (e.g., placing a bad onion back on the conveyor). Our baseline is a decentralized policy vector learned using a previous method Dec-AIRL [Suresh et al., 2023], which does not model types and therefore the cobot's policy is not type contingent. Dec-AIRL serves as an ablation of TB-DecAIRL for *disabled beliefs* and helps measure the benefit gained by TB-DecAIRL from the belief component alone.

Table 1: HRC performance comparison on MA-Gym simulation. Expert performance should be seen as an upper bound. Each episode lasts 100 timesteps.

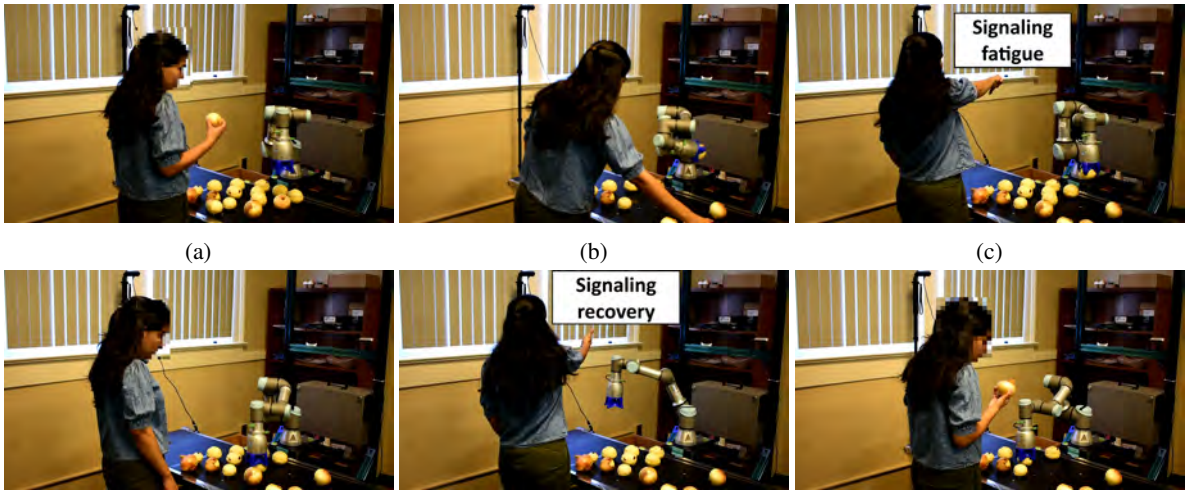
Average of $1,000$ episodes		
Method	**Onions Sorted Per Eps**	**Eps Reward**
Expert	64 ± 1	64 ± 1
TB-DecAIRL	$\mathbf{56 \pm 2}$	$\mathbf{55.7 \pm 0.78}$
Dec-AIRL	45 ± 2	43.2 ± 0.65

As depicted in Table 1, the decentralized policy vector learned by TB-DecAIRL scores a significantly higher average episode reward compared to the decentralized policies from Dec-AIRL and performs closer to expert behavior. TB-DecAIRL learns to use the belief over the human's type to adjust robot behavior accordingly. Notice that this adjusted behavior differs from the robot simply defaulting to a single sortation mode as in Dec-AIRL. The decision making learns that when the human is fatigued, fewer onions are sorted, thereby reducing the team's overall reward. This understanding makes the robot choose *Speed-up* action, which in turn increases the team's throughput. Although both methods operate within the same Dec-MDP framework for HRC, the baseline policy relies solely on an agent's local state attributes to decide actions. In contrast, TB-DecAIRL's policy accounts for both the agent's local state and their belief about the other agent(s)' type. This enables the cobot to use faster "industrial-mode" actions at the appropriate times leading to higher rewards.

4.2 PHYSICAL HRC EXPERIMENTS

Human processing behaviors may diverge slightly from that observed in the demonstration and sim2real challenges exist for the cobot in this domain. Consequently, we validate the simulation results using physical human-cobot experiments with five different human sorters to account for any variability. To sort with the human, we utilize a Universal Robots UR3e 6-DOF cobot equipped with a Realsense D435 RGB-D camera for onion detection and an OAK-D S2 RGB-D camera for human action estimation (see Fig. 5). The raw RGB frames from the Realsense camera are processed using a pre-trained object detection model YOLOv7, which generates bounding boxes around the onions on the conveyor. By combining these bounding boxes with their corresponding depth information, we compute the real-world 3D locations of the onions using rigid transforms and a pinhole camera model. Concurrently, we employ a hand-tracking method [Zhang et al., 2020], fine-tuned for our application, on the RGB input from the OAK-D camera. This output is similarly processed to determine the 3D location of the human hand to assess the human's actions. Whereas TB-DecAIRL yields a policy for each agent, only the cobot uses one of the type-contingent policies in the vector to control its behavior. This learned policy, exported to a CSV file, is loaded into a finite-state machine which is used to control the cobot via ROS Noetic.

Validation of simulation results. Five University students (not involved in this paper) played the role of the human sorter in the collaboration with small natural variability, and engaged in six rounds of onion sorting each across from



(a) (b) (c)

(d) (e) (f)

Figure 6: Key frames from a human-robot collaborative sort. In Fig. 6a, the human and cobot begin sorting with the human inspecting an unblemished onion while the cobot attempts to pick an onion. Fig. 6b captures the the human placing an onion back on the conveyor while the cobot inspects the picked onion. In Fig. 6c, the human indicates fatigue with a thumbs-down gesture to the OAK-D camera. The cobot responds by entering industrial mode in Fig. 6d, moving faster while the human performs NoOp. Finally, in Eq. 6e, the human signals recovery with a thumbs-up gesture, and both agents collaboratively complete the sort in Fig. 6f.

the cobot. The team was required to sort fifteen randomly placed onions on a conveyor in each round while the human indicates fatigue no more than two times. During the first three rounds, the cobot followed the policy learned from TB-DecAIRL, and for the remaining three rounds, it adhered to the Dec-AIRL policy. All trials were successful with no cobot failure in any round (an example sortation is shown in Fig. 6). In Fig. 7, we show the average time taken per round, under the two conditions of the cobot using the TB-DecAIRL policy and the baseline policy. The HRC where the cobot is aware of human factors and adapts accordingly took significantly less time to sort the onions compared to one which is not cognizant.

The human sorter held his or her thumb up or thumb down until it was recognized by the hand tracking application. This recognition was not instantaneous and the humans held their thumbs for slightly varying amounts of time across the 6 rounds. Indeed, this variability in gesture recognition is responsible in part for the variability of the sortation times of the 6 human-robot teams shown in Fig. 7. However, the two gestures (thumb-up and thumb-down) are quite distinct and the hand tracking application did not make any mistakes in distinguishing them.

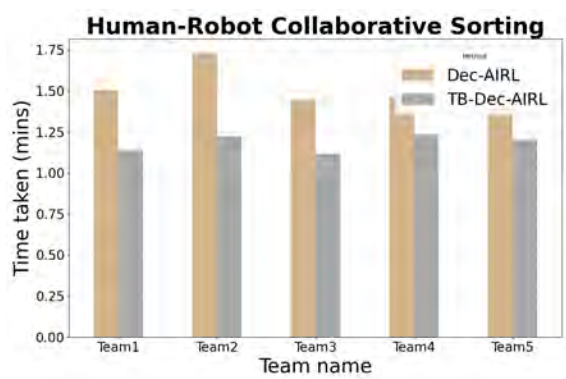


Figure 7: Average collaborative sort time for 15 onions with a UR3e cobot. Each human-robot team performed 3 rounds with the TB-DecAIRL policy and 3 rounds with the Dec-AIRL policy.

5 RELATED WORK

Modeling teammates to enhance decision making is common in fields such as game theory, multiagent planning, and multiagent reinforcement learning. Although previous work in *teammate modeling* has had success in toy simulated environments, e.g., SMAC [Samvelyan et al., 2019] and level-based foraging [Albrecht and Ramamoorthy, 2015], the body of research in this area is limited. A prominent approach in modeling other agents is interactive POMDPs (I-POMDPs) [Gmytrasiewicz and Doshi, 2005], which recursively updates beliefs about other agents' types while solving for their policies. A key distinction between I-POMDPs and game theory-based models as compared to ours is that the former is often tailored for non-cooperative contexts, not collaborative systems with shared goals.

Unhelkar [2020] uses an agent Markov model (AMM) to capture an agent's mental states, allowing learners to infer the AMM through variational Bayesian inference. Following up on this, recently Seo and Unhelkar [2024] learns the expert's policy and hidden intent (both part of the AMM) using expectation-maximization and IQ-Learn [Garg et al., 2021]. Unlike our multiagent collaborative HRC scenario, this method models the human as a single-agent MDP and whose type does not change.

Nikolaidis et al. [2015b] treats different expert behaviors as partially observable variables and uses expectation-maximization to cluster demonstrations, leading to different reward functions for each expert type through single-agent IRL. However, the model assumes that agent types remain constant throughout task execution, whereas our scenario accounts for *dynamic* types. Another prior thread uses a bounded-memory model combined with a mixed-observability MDP to estimate human adaptability [Nikolaidis et al., 2015a, 2017a,b], enabling the robot to adjust its actions accordingly. In contrast, TB-DecMDP adopts an objective perspective to capture the nuances of collaboration, allowing agents to adjust their behaviors as needed, to maximize team rewards. Peternel et al. [2018] develops a method to adapt the robot's physical behavior online to account for human motor fatigue, using techniques like dynamical movement primitives and adaptive frequency oscillators.

Another prior technique uses Trust-POMDP [Chen et al., 2020] model that maintains a belief over trust as a latent variable, allowing the robot to adapt its policy in response to human interruptions based on their trust in the robot's abilities. This work aligns more closely with active learning [Settles, 2009] than with traditional HRC. Recent studies by Yuan et al. [2022] and Jiang et al. [2024] apply variational inference models with mutual information maximization in decentralized settings to learn a random variable z that informs each agent about others. These works are similar to Wang et al. [2022], in that they assume a shared latent strategy space that remains constant throughout execution, whereas the TB-DecMDP allows for diverse and dynamic agent types during task execution.

More recently van der Spaa et al. [2024] focuses on learning both the human's hidden intent and task preferences for effective human-robot collaboration using maximum entropy IRL. They employ a multi-agent MDP model akin to Wang et al. [2022] to derive the marginalized policy, transition function, and reward function necessary for the robot to maintain a belief over human intent. This uses a centralized framework where all agents are aware of the joint state, and hence is not applicable to realistic scenarios that tend to be naturally decentralized. Prasad et al. [2024] utilizes a mixture of Gaussians to model the interactions between humans and robots and to learn the latent space of robot actions. However, the policy is conditioned solely on human movements, meaning the robot reacts to human actions rather than actively collaborating to solve tasks as part of a team.

6 CONCLUDING REMARKS

Motivated by the understanding that human behavior changes over time due to latent decision-making factors, we introduced a novel multi-agent model and an associated IRL method. This approach learns from human-human team demonstrations, enabling a collaborative robot (cobot) to work effectively with a human by accommodating changes in the human's type, thereby enhancing collaboration within a shared workspace. Supported by a conditional bound on performance loss, our experimental results indicate that TB-DecAIRL outperforms a previous decentralized IRL method in which the cobot does not reason about human types. Although TB-DecMDP is designed for HRC, it generally applies to any multiagent decision making in similar problem settings. TB-DecAIRL brings robot learning to HRC and its deployment a step closer to real-world collaborative environments.

Limitations. We assume that expert trajectories are available as state-action pairs (a common assumption in IRL). While the task attributes of the state can be captured using sensors like RGB-D cameras, certain hidden attributes such as the agent's own type are mental states. We leveraged simulated human-human trajectories to seamlessly generate expert trajectories. However, future work may reframe the state definition to attributes that can be fully captured through observable traits, and investigate the impact (e.g., on throughput) of errors and noise in observed actions that could lead to poor belief estimation of others' types. Future studies could also test on larger teams and accommodate asynchronous, durative actions, e.g., using a variant of the Dec-semi-MDP [Sutton et al., 1999, Goldman and Zilberstein, 2008] model.

Acknowledgements

This research was supported in part by NSF grant #IIS-1830421, by a grant from the Georgia Research Alliance, and by an intramural grant from University of Georgia's Precision Agriculture Institute, to PD. All opinions expressed in this paper are those of the authors alone and do not reflect on the sponsors. We would also like to acknowledge useful discussions with Vaibhav Unhelkar from Rice University and constructive feedback from the anonymous reviewers, which have substantially improved this paper.

References

Stefano V Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method

for ad hoc coordination in multiagent systems. *arXiv preprint arXiv:1506.01170*, 2015.

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

Min Chen, Stefanos Nikolaidis, Harold Soh, David Hsu, and Siddhartha Srinivasa. Trust-aware decision making for human-robot collaboration: Model learning and planning. *ACM Transactions on Human-Robot Interaction (THRI)*, 9(2):1–23, 2020.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv 2014. *arXiv preprint arXiv:1406.1078*, 2020.

Justin Fu, Katie Luo, and Sergey Levine. Learning Robust Rewards with Adverserial Inverse Reinforcement Learning. In *International Conference on Learning Representations*, pages 1–10. unknown, 2018.

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. IQ-learn: Inverse soft-Q learning for imitation. *Advances in Neural Information Processing Systems*, 34:4028–4039, 2021.

Adam Gleave and Sam Toyer. A primer on maximum causal entropy inverse reinforcement learning. *arXiv preprint arXiv:2203.11409*, 1, 2022.

Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

Claudia V. Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, New York, NY, USA, 2003. ACM. ISBN 1-58113-683-8.

Claudia V Goldman and Shlomo Zilberstein. Communication-based decomposition mechanisms for decentralized MDPs. *Journal of Artificial Intelligence Research*, 32:169–202, 2008.

Ziwei Guan, Tengyu Xu, and Yingbin Liang. When will generative adversarial imitation learning algorithms attain global convergence. In *International Conference on Artificial Intelligence and Statistics*, pages 1117–1125. PMLR, 2021.

Rui Jiang, Xuetao Zhang, Yisha Liu, Yi Xu, Xuebo Zhang, and Yan Zhuang. Multi-agent cooperative strategy with explicit teammate modeling and targeted informative communication. *Neurocomputing*, 586:127638, 2024.

Anurag Koul. MA-Gym: Collection of multi-agent environments based on OpenAI gym. `https://github.com/koulanurag/ma-gym`, 2019.

Francisco S Melo and Manuela Veloso. Decentralized MDPs with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789, 2011.

Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 663–670. Morgan Kaufmann, 2000.

Stefanos Nikolaidis, Przemyslaw Lasota, Ramya Ramakrishnan, and Julie Shah. Improved human–robot team performance through cross-training, an approach inspired by human team training practices. *The International Journal of Robotics Research*, 34(14):1711–1730, 2015a.

Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 189–196. IEEE, 2015b.

Stefanos Nikolaidis, Jodi Forlizzi, David Hsu, Julie Shah, and Siddhartha Srinivasa. Mathematical models of adaptation in human-robot collaboration. *arXiv preprint arXiv:1707.02586*, 2017a.

Stefanos Nikolaidis, Yu Xiang Zhu, David Hsu, and Siddhartha Srinivasa. Human-robot mutual adaptation in shared autonomy. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 294–302, 2017b.

Stefanos Z Nikolaidis. *Computational formulation, modeling and evaluation of human-robot team training techniques*. PhD thesis, Massachusetts Institute of Technology, 2014.

Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068, 2010.

Luka Peternel, Nikos Tsagarakis, Darwin Caldwell, and Arash Ajoudani. Robot adaptation to human physical fatigue in human–robot co-manipulation. *Autonomous Robots*, 42:1011–1021, 2018.

Vignesh Prasad, Alap Kshirsagar, Dorothea Koert Ruth Stock-Homburg, Jan Peters, and Georgia Chalvatzaki. MoVEInt: Mixture of Variational Experts for Learning Human-Robot Interactions from Demonstrations. *IEEE Robotics and Automation Letters*, 2024.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994. ISBN 0471619779.

Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In *Conference on Robot Learning*, pages 1634–1644, None, 2022. PMLR, None.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sangwon Seo and Vaibhav Unhelkar. IDIL: Imitation Learning of Intent-Driven Expert Behavior. *arXiv preprint arXiv:2404.16989*, 2024.

Burr Settles. Active learning literature survey. *Active Learning*, 2009.

Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

Prasanth Sengadu Suresh, Yikang Gui, and Prashant Doshi. Dec-AIRL: Decentralized Adversarial IRL for Human-Robot Teaming. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 1116–1124, 2023.

Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Vaibhav Vasant Unhelkar. *Effective information sharing for human-robot collaboration*. PhD thesis, Massachusetts Institute of Technology, 2020.

Linda van der Spaa, Jens Kober, and Michael Gienger. Simultaneously learning intentions and preferences during physical human-robot cooperation. *Autonomous Robots*, 48(4):11, 2024.

Chen Wang, Claudia Pérez-D'Arpino, Danfei Xu, Li Fei-Fei, Karen Liu, and Silvio Savarese. Co-GAIL: Learning diverse strategies for human-robot collaboration. In *Conference on Robot Learning*. PMLR, 2022.

Lei Yuan, Jianhao Wang, Fuxiang Zhang, Chenghe Wang, Zongzhang Zhang, Yang Yu, and Chongjie Zhang. Multi-agent incentive communication via decentralized teammate modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9466–9474, 2022.

Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

Brian Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, December 2010.

APPENDIX

PROOF OF THEOREM 1

Proof. We use max-norms $\|.\|$ over the sets of states, actions and next states. We refer to joint beliefs $b^t(\ldots) = \prod_i b_i^t(\ldots)$. As in Dec-AIRL, we distinguish interactive states (S_I) from non-interactive states (S_{NI}) where the reward function is given as:

$$R(s, \mathbf{a}, s') = \begin{cases} R(s, \mathbf{a}, s'), \text{ if } s \in S_I \\ \sum_i R_i(s_i, a_i, s_i'), \text{ if } s \in S_{NI} \end{cases} \tag{7}$$

Type transition kernel $F_i(\theta_j'|a_j, \theta_j)$ forms a Markov chain with state θ_j and edges guarded by a_j. Under the assumption that F_i is irreducible and aperiodic, the type distribution b_i given by Eq. 3 will converge to a limiting distribution. Let the joint beliefs, $b^t(\boldsymbol{\theta}) = \prod_i b_i^t(\boldsymbol{\theta}_{-i})$. Then, after sufficient time t elapses,

$$\mathcal{D}_{TV}(b^{t+1}, b^t) \leq \delta \cdot \mathcal{D}_{TV}(b^t, b^{t-1}) \tag{8}$$

where $0 \leq \delta < 1$ and $\mathcal{D}_{TV}$ denotes the total variation distance.

First, we note the definition and upper-bound of $\mathcal{D}_{TV}(b^t, b^{t-1}) = \frac{1}{2}\sum_\theta |b^t(\theta|h^{t-1}) - b^{t-1}(\theta|h^{t-2})| \leq 1$, and the fact that if $\mathcal{D}_{TV}(P, Q) \leq \epsilon$ then for any event A, $P(A) \geq Q(A)(1 - \epsilon/2)$. Applying these to the property of belief updates (Eq. 8) we get

$$\begin{aligned} b^{t+1}(\theta|h^t) &\geq b^t(\theta|h^{t-1})(1 - \delta\mathcal{D}_{TV}(b^t, b^{t-1})/2) \\ &\geq b^t(\theta|h^{t-1})(1 - \delta/2) \end{aligned}$$

Therefore, $\frac{b^{t+1}(\theta|\ldots)}{b^t(\theta|\ldots)} \geq (1 - \frac{\delta}{2}), \forall \theta, t$ when t is sufficiently large. Note that this applies to both the expert and generated trajectories. Now, the estimates of joint rewards in non-interactive (NI) states is

$$\begin{aligned} R^t &= \sum_i R_i^t \\ &= \sum_i \log\left(\frac{D_i^t}{1 - D_i^t}\right) \end{aligned}$$

Similarly, the expert rewards in non-interactive states are $R^E = \sum_i \log\left(\frac{D_i^E}{1 - D_i^E}\right)$. Now, the i-th agent's discriminator error is $\|D_i^t - D_i^E\| \leq \epsilon$, and the i-th expert's discriminator value [4] is 0.5, therefore

$$\begin{aligned} \|R^t - R^E\| &= \|\sum_i \log\left(\frac{D_i^t}{1 - D_i^t}\right) - \sum_i \log\left(\frac{D_i^E}{1 - D_i^E}\right)\| \\ &= \|\sum_i \log\left(\frac{D_i^t}{D_i^E}\right) - \sum_i \log\left(\frac{1 - D_i^t}{1 - D_i^E}\right)\| \\ &\leq \left|\sum_i \log\left(\frac{1/2 + \epsilon}{1/2 - \epsilon}\right) - \sum_i \log\left(\frac{1/2 - \epsilon}{1/2 + \epsilon}\right)\right| \\ &= 2\sum_i \log\left(\frac{1/2 + \epsilon}{1/2 - \epsilon}\right) \\ &\leq 8N\epsilon \end{aligned} \tag{9}$$

when ϵ is small enough and N is the number of agents. For interactive states, the same arguments apply to the joint discriminator, but the bound corresponding to Eq. 9 would just be 8ϵ. Therefore, we use Eq. 9 as the dominant form of this bound.

The log-likelihood of a trajectory (X) given the reward estimate R^t is

$$\begin{aligned} \log P(\mathcal{X}|R^t) &\propto \sum_t \gamma^t \sum_\theta R^t(s^t, a^t, \theta) b^t(\theta|h^t) \\ &\geq \sum_t (\gamma(1 - \delta/2))^t \sum_\theta R^t(\ldots, \theta)\rho_1(\theta) \\ &\geq \sum_t (\gamma(1 - \delta/2))^t \sum_\theta \left|R^E(\ldots, \theta) - 8N\epsilon\right| \rho_1(\theta) \\ &\geq \log P(\mathcal{X}|R^E) - \sum_t (\gamma(1 - \delta/2))^t 8N\epsilon \end{aligned}$$

Here, h^t is the history of states-actions preceding step t and ρ_1 is the initial belief. Thus,

$$\begin{aligned} LL(\mathcal{X}|R^E) - LL(\mathcal{X}|R^t) &\leq \sum_t (\gamma(1 - \delta/2))^t 8N\epsilon \\ &\leq \frac{8N\epsilon}{1 - \gamma(1 - \delta/2)} \end{aligned}$$

□

HYPERPARAMETERS OF THE METHODS

For both TB-DecAIRL and Dec-AIRL, we used 256 hidden nodes for the 2 hidden layers in both the g and h networks within the discriminator, a batchsize of 128, learning rate for both the discriminator and the actors as 0.0003, discount factor and generalized advantage estimator 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 [Raffin et al., 2022] because it has been tuned and tested on multiple pybullet environments. During our training, we observed discriminator errors of about 14% at termination for TB-DecAIRL.

[4] An expert's discriminator attempts to distinguish between expert trajectories and those produced by the expert's policy, which are indistinguishable.