MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Open Human-Robot Collaboration using Decentralized Inverse Reinforcement Learning

Suresh, Prasanth; Jain, Siddarth; Doshi, Prashant; Romeres, Diego

TR2024-135 October 02, 2024

Abstract

The growing interest in human-robot collaboration (HRC), where humans and robots cooperate towards shared goals, has seen significant advancements over the past decade. While previous research has addressed various challenges, several key issues remain unresolved. Many domains within HRC involve activities that do not necessarily require human presence throughout the entire task. Existing literature typically models HRC as a closed system, where all agents are present for the entire duration of the task. In contrast, an open model offers flexibility by allowing an agent to enter and exit the collaboration as needed, enabling them to concurrently manage other tasks. In this paper, we introduce a novel multiagent framework called oDec-MDP, designed specifically to model open HRC scenarios where agents can join or leave tasks flexibly during execution. We generalize a recent multiagent inverse reinforcement learning method - Dec-AIRL to learn from open systems modeled using the oDec-MDP. Our method is validated through experiments conducted in both a simplified toy firefighting domain and a realistic dyadic human-robot collaborative assembly. Results show that our framework and learning method improves upon its closed system counterpart.

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2024)

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Open Human-Robot Collaboration using Decentralized Inverse Reinforcement Learning

Prasanth Sengadu Suresh[1,2], Siddarth Jain[2], Prashant Doshi[1], Diego Romeres[2]

***Abstract*— The growing interest in human-robot collaboration (HRC), where humans and robots cooperate towards shared goals, has seen significant advancements over the past decade. While previous research has addressed various challenges, several key issues remain unresolved. Many domains within HRC involve activities that do not necessarily require human presence throughout the entire task. Existing literature typically models HRC as a *closed* system, where all agents are present for the entire duration of the task. In contrast, an *open* model offers flexibility by allowing an agent to enter and exit the collaboration as needed, enabling them to concurrently manage other tasks. In this paper, we introduce a novel multiagent framework called oDec-MDP, designed specifically to model open HRC scenarios where agents can join or leave tasks flexibly during execution. We generalize a recent multiagent inverse reinforcement learning method - Dec-AIRL to learn from *open* systems modeled using the oDec-MDP. Our method is validated through experiments conducted in both a simplified toy firefighting domain and a realistic dyadic human-robot collaborative assembly. Results show that our framework and learning method improves upon its closed system counterpart.**

I. INTRODUCTION

As the landscape of artificial intelligence and robotics continues to evolve, the collaboration between humans and robots has gained significant importance. This partnership harnesses the unique and often complementary strengths of both humans and robots [1]. Robots, equipped with sensory perception and intelligent decision-making abilities, play a crucial role as collaborators in enhancing efficiency, precision, and creativity across diverse fields. The Human-Robot Collaboration (HRC) paradigm is not geared towards replacing human labor; rather, it focuses on augmenting human capabilities, empowering individuals to tackle challenges that were once considered insurmountable. Previous studies in HRC have explored human and robotic agent interactions using multiagent decision-making frameworks [2]. However, many existing multiagent models typically operate within a *closed system* framework, where a fixed group of human and robotic agents collaborate from initiation to completion of a task. This closed system approach lacks the flexibility provided by an *open* system, where agents can dynamically join or leave the task at various stages as needed. This characteristic of openness is referred to as *agent openness* [3].

In this work, we consider domains where only a subset of tasks necessitate human collaboration. Recognizing human limitations in time and energy, an *open* system permits humans to seamlessly join and collaborate with robots when their input is essential. Such a framework is termed as an *open-HRC system (OHRCS)*. Despite its effectiveness, OHRCS introduces several challenges. For instance, in collaborative table assembly tasks with numerous components (Fig. 3), there can exist multiple valid sequences for assembly, with only a small subset of tasks requiring human intervention. Subsequently, there may be a particular order that minimizes the time and effort of the human. In such situations, the primary difficulty lies in crafting a model that is sufficiently sophisticated to encompass a wide range of potential behaviors. This multiagent model needs to accurately represent the behaviors of both the existing team of agents and any new agents that join, as well as the nature of the task itself. Furthermore, given that many real-world scenarios are decentralized [4]—meaning each agent may lack complete information about the others–the model must account for these system dynamics. The second major challenge involves reward engineering. The reward function must be intricate enough to encourage behaviors that effectively solve the task optimally, while also balancing the increased cost incurred by utilizing human assistance. Such reward shaping is non-trivial.

In this paper, we describe our inverse reinforcement learning (IRL) based approach to the aforementioned OHRCS challenges and make two key contributions. First, we present oDec-MDP, a novel multiagent decision-making framework to model agent openness in OHRCS. Second, we develop a novel IRL technique - oDec-AIRL that generalizes a recent decentralized IRL method - Dec-AIRL [5], to learn the underlying reward function and its corresponding vector of policies using the oDec-MDP as the behavioral model. We validate our contributions on two domains: a simulated Urban Firefighting [3], [6]–[8] scenario and a realistic physical human-robot collaborative furniture assembly.

II. BACKGROUND

Multiagent IRL typically models the expert using multiagent generalizations of the Markov decision process (MDP) such as multiagent MDP (MMDP), Markov game, or decentralized MDP (Dec-MDP). Due to the nature of HRC being decentralized and collaborative, a Dec-MDP is appropriate to model the collaborative expert team. A two-agent Dec-MDP can be formally defined as a tuple

$$\mathcal{DM} \triangleq \langle S, A, T, R \rangle$$

where the global 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, which

[1]School of Computing, University of Georgia, Athens, GA, USA. This research was completed during P. Suresh's internship at MERL. {ps32611,pdoshi}@uga.edu

[2]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA. {sjain,romeres}@merl.com

when combined yield the complete global 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 multi-agent system; and $R : S \times A \rightarrow \mathbb{R}$ is the common reward function [1]. In IRL, the latter is unknown, whereas the rest of the elements are usually known. As such, the agents know their local state and any common task attributes; agents act independently while optimizing a task-centric reward function [10]. Let $\mathcal{X}^E$ be the set of expert demonstrations and a complete trajectory $X^E \in \mathcal{X}^E$ is given by, $X^E = (\langle s_i^0, s_j^0\rangle, \langle a_i^0, a_j^0\rangle, \langle s_i^1, s_j^1\rangle, \langle a_i^1, a_j^1\rangle, ..., \langle s_i^{\mathcal{T}}, s_j^{\mathcal{T}}\rangle, \langle a_i^{\mathcal{T}}, a_j^{\mathcal{T}}\rangle)$.

A. Review of Decentralized Adversarial IRL

Decentralized adversarial IRL (Dec-AIRL) [5] generalizes the single-agent deep-IRL method - adversarial IRL [11] (that works on the principle of maximum causal entropy) to learn a common reward function for the team, from expert demonstrations. AIRL uses a discriminator $D_{\boldsymbol{\theta}}(X)$ to learn a function $f_{\boldsymbol{\theta}}(X)$ [11] which at convergence approximates the advantage function corresponding to the expert's policy. Dec-AIRL analytically represents the discriminator as: $D_{\boldsymbol{\theta}}(X) = \frac{e^{f_{\boldsymbol{\theta}}(X)}}{e^{f_{\boldsymbol{\theta}}(X)}+\pi(X)}$ and the reward update rule is given as

$$R_{\boldsymbol{\theta}}(X) \leftarrow \log D_{\boldsymbol{\theta}}(X) - \log(1 - D_{\boldsymbol{\theta}}(X)). \tag{1}$$

Eq. (1) when simplified yields: $f_{\boldsymbol{\theta}} - \log(\pi)$, which is the entropy-regularized reward formulation. In the underlying Dec-MDP [12], each agent only has access to their local state and some general task attributes. Dec-PPO - a decentralized generalization of the popular RL method - Proximal Policy Optimization [13], is used as Dec-AIRL's forward-rollout technique. Dec-PPO uses the centralized training, decentralized execution paradigm where the centralized critic network updates its value function as a squared-error loss:

$$L_t^{VF}(\omega) = (V^{\pi_\omega}(s^t) - \hat{V}_t^{targ})^2.$$

where $\hat{V}_t^{targ}$ is the per-episode discounted reward-to-go and $V^{\pi_\omega}(s^t)$ is the predicted value of global state s^t and ω is the policy weight vector. For a dyadic system with agents i and j, the policy loss of agent i is given by

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

where $\lambda_i^t = \frac{\pi_{\omega,i}(a_i^t|s_i^t)}{\pi_{\omega,i}^{old}(a_i^t|s_i^t)}$ is the importance sampling ratio. $L^{CLIP}(\omega)$ 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^t outside the interval $[1-\epsilon, 1+\epsilon]$ is reduced. $A_{\pi_\omega,i}^t$ is calculated with respect to the reward estimates $R_{\boldsymbol{\theta}}(X)$ from Eq. (1). This clipped surrogate objective, combined with the policy entropy, handles the explore-exploit dilemma. The policy entropy loss is given as:

$$L_i^{ENT}(\omega) = \sigma H[\pi_{\omega,i}(s_i^t)].$$

[1]This Dec-MDP describes a locally fully observable model whose local states when combined yield the fully observable global state [9].

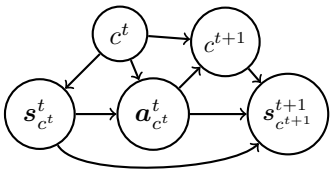


Fig. 1: The oDec-MDP graphical model for two timesteps t and $t+1$. Given the collab team ID c^t at timestep t, $\boldsymbol{s}^t_{c^t}$ is formed by combining the local states of all agents in c^t. All agents' local actions from c^t combined form $\boldsymbol{a}^t_{c^t}$, which leads to c^{t+1}, given c^t. c^{t+1}, $\boldsymbol{a}^t_{c^t}$ and $\boldsymbol{s}^t_{c^t}$ together lead to the next state $\boldsymbol{s}^{t+1}_{c^{t+1}}$ at time $t+1$.

where H is the policy entropy and σ is the entropy hyperparameter. The total loss is then given as:

$$L_i(\omega) = L_i^{CLIP}(\omega) + L_i^{ENT}(\omega). \tag{2}$$

The policy loss, entropy loss, and total loss apply analogously for agent j. At the end of training, the discriminator and generator return the learned common reward function and the converged vector of policies respectively.

III. Open Human-Robot Collaboration

In this section, we introduce the novel oDec-MDP and describe how oDec-AIRL uses it to model OHRC problems.

A. Open Collaboration Model

oDec-MDP generalizes Dec-MDP to model agent openness in a decentralized, collaborative setting. Formally,

$$\text{oDec-MDP} \triangleq \langle Ag, C, \boldsymbol{S}, \boldsymbol{A}, \Gamma, T, R, \rho\rangle$$

- Ag is the finite set of all agents and $|Ag| = N$ is the maximum number of agents;
- $C : \mathcal{P}(Ag) \rightarrow \mathbb{N}$ assigns a unique number identifier to each collaborating team and $\mathcal{P}$ denotes the powerset excluding the empty set. For convenience, we let C also denote the set of all assigned identifiers;
- Global state space $\boldsymbol{S} = \bigcup_{c=1}^{c=|C|} S_c$ where $c \in C$ and S_c denotes the set of states of the team identified by c;
- Global action space $\boldsymbol{A} = \bigcup_{c=1}^{c=|C|} A_c$ where $c \in C$ and A_c denotes the set of *joint* actions of the team identified by c. For instance, if team c involves agents i and j whose action sets are A_i and A_j, respectively, then $A_c = A_i \times A_j$;
- Team transition model $\Gamma : C \times \boldsymbol{A} \times C \rightarrow [0, 1]$ gives the distribution of the new teams given the current team and action letting agent(s) enter or exit;
- $T = \{T_c, T_c' \mid c = 1, 2, \ldots, |C|\}$, where *intra-team* state transition model $T_c : S_c \times A_c \times S_c \rightarrow [0, 1]$ gives the distribution over the team's next state and *inter-team* state transition model $T_c' : S_c \times C' \times S_{c'} \rightarrow [0, 1]$ gives the distribution over the next team's state. Both are available for all $c, c' \in C$;
- Common reward function shared by all agents in each team c, $R_c \triangleq R(S_c, A_c, c)$ and $R_c : S_c \times A_c \rightarrow \mathbb{R}$;
- Start state and team prior distribution $\rho : S \times C \rightarrow [0, 1]$.

An open teamwork trajectory of length $\mathcal{T}$ contains the collaborating team ID, team state, and team action at each time step:

$$X^E \triangleq \Big(\langle c, \boldsymbol{s}_c, \boldsymbol{a}_c\rangle^1, \langle c, \boldsymbol{s}_c, \boldsymbol{a}_c\rangle^2, \langle c', \boldsymbol{s}_{c'}, \boldsymbol{a}_{c'}\rangle^3 \ldots \langle c'', \boldsymbol{s}_{c''}, \boldsymbol{a}_c''\rangle^{\mathcal{T}}\Big).$$

Notice from the trajectory that the starting team with ID c persists for the first two timesteps followed by a change to team c'. If the team with ID c at time step $t = 1$ is a dyad with agents i and j, then the policy $\boldsymbol{\pi}$, the team state $\boldsymbol{s}_c^t$, and team action $\boldsymbol{a}_c^t$ are vectors of the two individual agents' policies, their partial states, and their actions respectively:

$$\boldsymbol{\pi}_c \triangleq \langle \pi_i, \pi_j \rangle;\ \boldsymbol{s}_c^t \triangleq \langle s_i^t, s_j^t \rangle;\ \text{and } \boldsymbol{a}_c^t \triangleq \langle a_i^t, a_j^t \rangle.$$

The likelihood of the first two time steps of X^E is obtained using the parameters of oDec-MDP as:

$$\begin{aligned}
&\Pr(c, \boldsymbol{s}_c^1, \boldsymbol{a}_c^1, c, \boldsymbol{s}_c^2, \boldsymbol{a}_c^2) \\
&= \Pr(c, \boldsymbol{s}_c^2, \boldsymbol{a}_c^2 | c, \boldsymbol{s}_c^1, \boldsymbol{a}_c^1) \Pr(\boldsymbol{a}_c^1 | c, \boldsymbol{s}_c^1) \Pr(c, \boldsymbol{s}_c^1) \\
&= \Pr(\boldsymbol{a}_c^2 | c, \boldsymbol{s}_c^2) \Pr(c | c, \boldsymbol{a}_c^1) \Pr(\boldsymbol{s}_c^2 | \boldsymbol{s}_c^1, \boldsymbol{a}_c^1) \Pr(\boldsymbol{a}_c^1 | c, \boldsymbol{s}_c^1) \Pr(c, \boldsymbol{s}) \\
&= \Pr(a_i^2 | c, s_i^2) \Pr(a_j^2 | c, s_j^2)\ \Gamma(c, \boldsymbol{a}_c^1, c)\ T_c(\boldsymbol{s}_c^1, \boldsymbol{a}_c^1, \boldsymbol{s}_c^2) \\
&\quad \times \Pr(a_i^1 | c, s_i^1) \Pr(a_j^1 | c, s_j^1) \Pr(c, \boldsymbol{s}_c^1) \\
&= \underbrace{\pi_i(a_i^2 | c, s_i^2)}_{\text{Policy of i at t=2}}\ \underbrace{\pi_j(a_j^2 | c, s_j^2)}_{\text{Policy of j at t=2}}\ \underbrace{\Gamma(c, \boldsymbol{a}_c^1, c)}_{\text{Team transition}}\ \underbrace{T_c(\boldsymbol{s}_c^1, \boldsymbol{a}_c^1, \boldsymbol{s}_c^2)}_{\text{intra-team state transition}} \\
&\quad \times\ \underbrace{\pi_i(a_i^1 | c, s_i^1)}_{\text{Policy of i at t=1}}\ \underbrace{\pi_j(a_j^1 | c, s_j^1)}_{\text{Policy of j at t=1}}\ \underbrace{\rho(c, \boldsymbol{s}_c^1)}_{\text{given prior}}. \qquad (3)
\end{aligned}$$

A locally fully observable Dec-MDP lets each agent's policy condition its action on the agent's partial view of the state. Next, we show how the likelihood is obtained for the second and third timesteps of X^E when the team changes. Its derivation proceeds analogously to the above steps:

$$\begin{aligned}
&\Pr(c, \boldsymbol{s}_c^2, \boldsymbol{a}_c^2, c', \boldsymbol{s}_{c'}^3, \boldsymbol{a}_{c'}^3) = \pi_i(a_i^3 | c', s_i^3)\ \pi_j(a_j^3 | c', s_j^3)\ \Gamma(c, \boldsymbol{a}_c^2, c') \\
&\quad \times\ T_c'(\boldsymbol{s}_c^2, c', \boldsymbol{s}_{c'}^3)\ \pi_i(a_i^2 | c, s_i^2)\ \pi_j(a_j^2 | c, s_j^2)\ \rho(c, \boldsymbol{s}_c^2). \qquad (4)
\end{aligned}$$

The key difference between Eqs. 3 and 4 is that the latter involves the inter-team transition function T_c' due to the change of team from time step $t = 2$ to $t = 3$. For the sake of completeness, we also show below how the value of a given team state at timestep t, $\boldsymbol{s}_c^t$, is obtained, which defines the value function of our oDec-MDP:

$$\begin{aligned}
V(\boldsymbol{s}_c^t) &= \max_{\boldsymbol{a}_c^t}\ \mathbb{E}_{c', \boldsymbol{s}_{c'}^{t+1}} \left[R(\boldsymbol{s}_c^t, \boldsymbol{a}_c^t, c) + \gamma V(\boldsymbol{s}_{c'}^{t+1}) | \boldsymbol{s}_c^t, c \right] \\
&= \max_{\boldsymbol{a}_c^t}\ R(\boldsymbol{s}_c^t, \boldsymbol{a}_c^t, c) + \gamma \sum_{C'} \sum_{\boldsymbol{S}_{c'}} \Pr(c', \boldsymbol{s}_{c'}^{t+1} | c, \boldsymbol{s}_c^t, \boldsymbol{a}_c^t)\ V(\boldsymbol{s}_{c'}^{t+1}) \\
&= \max_{\boldsymbol{a}_c^t}\ R_c + \gamma \sum_{C'} \sum_{\boldsymbol{S}_{c'}}\ \Gamma(c, \boldsymbol{a}_c^t, c')\ T_c'(\boldsymbol{s}_c^t, c', \boldsymbol{s}_{c'}^{t+1})\ V(\boldsymbol{s}_{c'}^{t+1}).
\end{aligned}$$

B. Method

The discriminator $D_{\boldsymbol{\theta}}(X)$ of oDec-AIRL learns a common reward function R_c contingent on $c, \boldsymbol{s}$, and $\boldsymbol{a}$. This common reward function is then used by oDec-PPO to learn a vector of policies (one for each agent). oDec-AIRL minimizes the reverse KL divergence between the learner's and expert's marginal teamID-state-action distribution $KL(\mathrm{P}_\pi(c, \boldsymbol{s}, \boldsymbol{a})\ ||\ \mathrm{P}_{exp}\ (c, \boldsymbol{s}, \boldsymbol{a}))$. The oDec-AIRL algorithm shown in Algorithm 1 takes the oDec-MDP ($\mathcal{ODM}$) without the reward and transition functions, and the expert trajectories $\mathcal{X}^E$, as input. The goal is to learn a common reward function R_c for the task based on $\mathcal{X}^E$, and the corresponding vector of learned policies.

The algorithm begins by initializing a random decentralized policy vector $\boldsymbol{\pi}_c$ (line 1), and a discriminator $D_{\boldsymbol{\theta}}$ with random weights $\boldsymbol{\theta}$. Learning continues until the end of training iterations (line 2-10). Each iteration, it generates joint trajectories $\hat{\mathcal{X}}$ using the current policy vector $\boldsymbol{\pi}_c$. It then samples minibatches of $\langle c, \boldsymbol{s}, \boldsymbol{a} \rangle$ from $\hat{\mathcal{X}}$ and $\mathcal{X}^E$ to yield $\hat{\mathcal{Y}}$ and $\mathcal{Y}^E$ respectively (line 4). It trains $D_{\boldsymbol{\theta}}$ using $\hat{\mathcal{Y}}$ and $\mathcal{Y}^E$ to minimize the reverse KL divergence between the expert and learned distributions (line 6). Using $D_{\boldsymbol{\theta}}$'s confusion it extracts an updated reward R_c (line 7). R_c is then provided as an input to the generator $G(R_c)$ using oDec-PPO which learns the forward rollout vector of policies (line 10). Finally, the learned reward function R_c and policy vector $\boldsymbol{\pi}_c$ are returned (line 11).

Algorithm 1: oDec-AIRL

Input: $\mathcal{ODM}$ sans R_c, Γ and T; Exp trajs $\mathcal{X}^E$.
Output: Learned common reward function R_c.

1 Initialize decentralized policy $\boldsymbol{\pi}_c$, Discriminator $D_{\boldsymbol{\theta}}$.
2 **for** $iter \leftarrow 0$ **to** $train\_iters$ **do**
3 Generate joint trajectories $\hat{\mathcal{X}}$ using $\boldsymbol{\pi}_c$.
4 Sample joint $\langle c, \boldsymbol{s}, \boldsymbol{a} \rangle$ minibatches $\hat{\mathcal{Y}}$ and $\mathcal{Y}^E$ from $\hat{\mathcal{X}}$ and $\mathcal{X}^E$, respectively.
5 **for** $ep \leftarrow 0$ **to** $discriminator\_epochs$ **do**
6 Train discriminator $D_{\boldsymbol{\theta}}$ to minimize $KL(\mathrm{P}_\pi(c, \boldsymbol{s}, \boldsymbol{a})\ ||\ \mathrm{P}_{exp}(c, \boldsymbol{s}, \boldsymbol{a}))$.
7 Update reward $R_c \leftarrow \log D_{\boldsymbol{\theta}}(X) - \log(1 - D_{\boldsymbol{\theta}}(X))$.
8 **for** $ep \leftarrow 0$ **to** $generator\_epochs$ **do**
9 Train generator $G(R_c) \leftarrow$ oDec-PPO.
10 Get updated policy $\boldsymbol{\pi}_c \leftarrow G(R_c)$.
11 **return** R_c, $\boldsymbol{\pi}_c$.

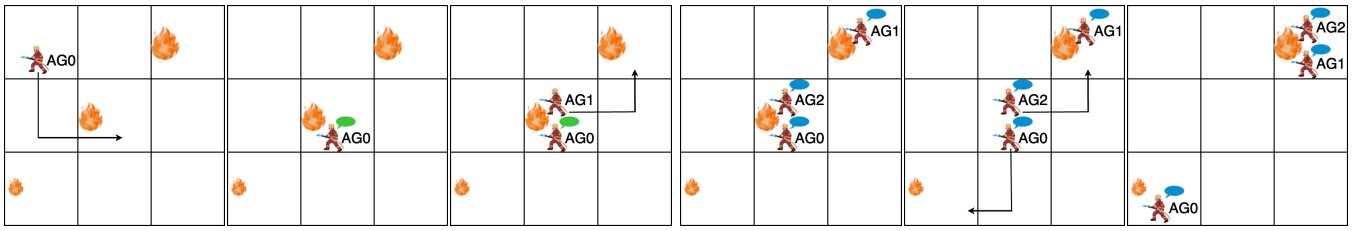
IV. Experiments

In this section, we consider two domains, a popular simulated toy domain - Urban Firefighting [3], [6]–[8], and a realistic dyadic collaborative human-robot furniture assembly domain. In both domains, we focus on the agent being called in at the right time. Nonetheless, our method applies analogously to agents exiting the task. To establish the efficacy of an open system over a closed one, we compare the learned behavior of oDec-AIRL with an ablation study using Dec-AIRL, on both domains.

A. Urban Firefighting

In this domain, the goal is for the agent(s) to extinguish fires within a 3x3 grid as efficiently as possible. There are three active fires: a large, medium, and small fire as shown in Fig. 2 with intensities 0.9, 0.6, and 0.3 respectively at the beginning. Each firefighter has 4 cardinal direction actions, a CallAgent action, and an Extinguish action. A CallAgent action calls one agent at a time (up to 3 agents) and the Extinguish action at a fire location reduces its intensity by 0.1. Each agent's local state contains their location, the fire locations and intensities, and the number of teammates at their location. Agents need to decide whether to call another agent to extinguish the fires effectively. The challenge in this domain is that the learned reward function must maintain a delicate balance between the cost incurred by having additional agents and the reward of extinguishing fires sooner.

Results: To provide expert data for training, we generated simulated trajectories based on human preferences, of 10^6



(a) (b) (c) (d) (e) (f)

Fig. 2: A complete episode of the Urban Firefighting domain with learned oDec-AIRL policies. Colored bubbles green and blue represent CallAgent and Extinguish actions respectively. In Figure 2a, Agent 0 heads towards the medium fire, and Figure 2b, calls Agent 1 to join. Subsequently, Agent 1 moves to the large fire in Figure 2c, while Agent 0 calls Agent 2. In Figure 2d, all agents extinguish fires at their locations. After extinguishing the medium fire, Agent 0 moves to the small fire, and Agent 2 assists Agent 1 with the large fire in Figure 2e. Figure 2f shows agents at their final locations performing the Extinguish action. This visualization was built using PyGame [14].

TABLE I: Urban Firefighting learned policies comparison

Average of 10000 timesteps					
Max Num Agents	Method	Num Steps Active Per Eps			Episode Reward
		Agent0	Agent1	Agent2	
2	**oDec-AIRL**	$\mathbf{18.06 \pm 1.13}$	$\mathbf{13.06 \pm 0.61}$	-	$\mathbf{32.22 \pm 0.59}$
	Dec-AIRL	16.87 ± 0.09	16.87 ± 0.09	-	30.37 ± 0.07
3	**oDec-AIRL**	$\mathbf{12.36 \pm 0.57}$	$\mathbf{9.27 \pm 0.12}$	$\mathbf{8.27 \pm 0.02}$	$\mathbf{37.86 \pm 0.13}$
	Dec-AIRL	10.33 ± 1.29	10.33 ± 1.29	10.33 ± 1.29	35.73 ± 0.70

total timesteps for both oDec-AIRL and Dec-AIRL, trained both methods for 10^7 timesteps, and compared their best-learned behaviors using average episode reward and the average number of timesteps each agent is active per episode. As can be observed from Table I, in the behavior learned by Dec-AIRL, all agents are present throughout the task duration as expected, and in turn incur a higher step-cost. In comparison, oDec-AIRL policies only engage the second and third agents for 9.27 and 8.27 steps, respectively, in the three-agent case and accrue a higher average episodic reward. Note that the total number of steps per episode is slightly greater in an open system compared to a closed one since agents are being called in one at a time as needed.

B. Collaborative Furniture Assembly

In this realistic dyadic HRC furniture assembly task, the goal is to assemble a table consisting of multiple parts: base, leg-support1, leg-support2, leg1, leg2, two screws for each leg-support to connect them to the base, and two screws for each leg to screw them into their respective supports, as shown in Fig. 3. The task can be completed in multiple ways. One may position both leg-supports on the base and screw them in before positioning their corresponding legs and screwing the legs into their respective supports. Alternatively, one may position leg-support1, screw it into the base, place the leg1, screw it into the leg-support1, and analogously repeat the sequence for the other parts to complete the assembly. Notice that the positioning actions can be done independently by the robot, while the screwing action requires the assistance of a human. While the speed of assembly could be increased by having the human position parts in parallel from the beginning, the step-cost incurred due to the human's presence would be quite high. This means that the learned reward function must optimize completing the assembly sooner and the step-cost due to the human's presence. In other words, the optimal behavior must only call the human into the task when imperative.

Each agent has 8 discrete actions: *ChooseTask* - This randomly assigns a valid next task to perform, *Pick* - Agent picks up the current part, *Place* - Agent places the current part at the goal location, *HoldInPlace* - Agent holds the current part steadily at its current location, *ScrewIn* - Agent screws the current part into place, *CallAgent* - This calls the human into the task, *ResetTask* - Agent places the current part back to its original location, *NoOp* - No action. The local state of each agent in the expert's oDec-MDP consists of three discrete variables: *TaskName* - which takes a valid task name from eleven discrete values when a ChooseTask action is performed; *TaskStatus* - which describes the current status of the task through one of seven discrete values; *Collab* - which provides the current collaboration level between unavailable, partial and full collaboration. This domain was developed as a text-based discrete-state-action Gym environment on MA-Gym [15] based on domain knowledge of the assembly task. After completing a screwing subtask, the human can either stay idle (doing *NoOp*) until the robot needs help again or participate by positioning other parts in parallel. The upside to the latter is that the task is completed sooner and the team receives a better reward; the downside is that if the human is engaged in a different task while the robot requires help, the human must perform a *ResetTask* action before helping the robot, extending the task duration.

Simulation Experiments: We generated simulated expert trajectories of 10^6 total timesteps and trained both oDec-AIRL and Dec-AIRL for 10^7 timesteps. These simulated trajectories were generated based on human preferences and domain knowledge to generate team behavior that optimally completes the task while minimizing human time and effort. In these trajectories, the human agent on average intervenes

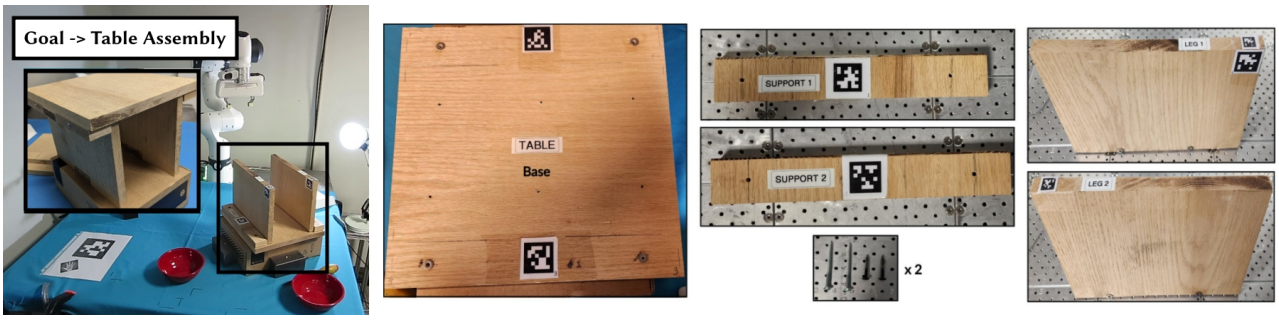


Fig. 3: A collaborative table assembly task that involves placing and screwing together various wooden components. *Left:* The assembled table. *Right:* The individual components required for assembly, including the table base, two supports, two legs, and the necessary screws.

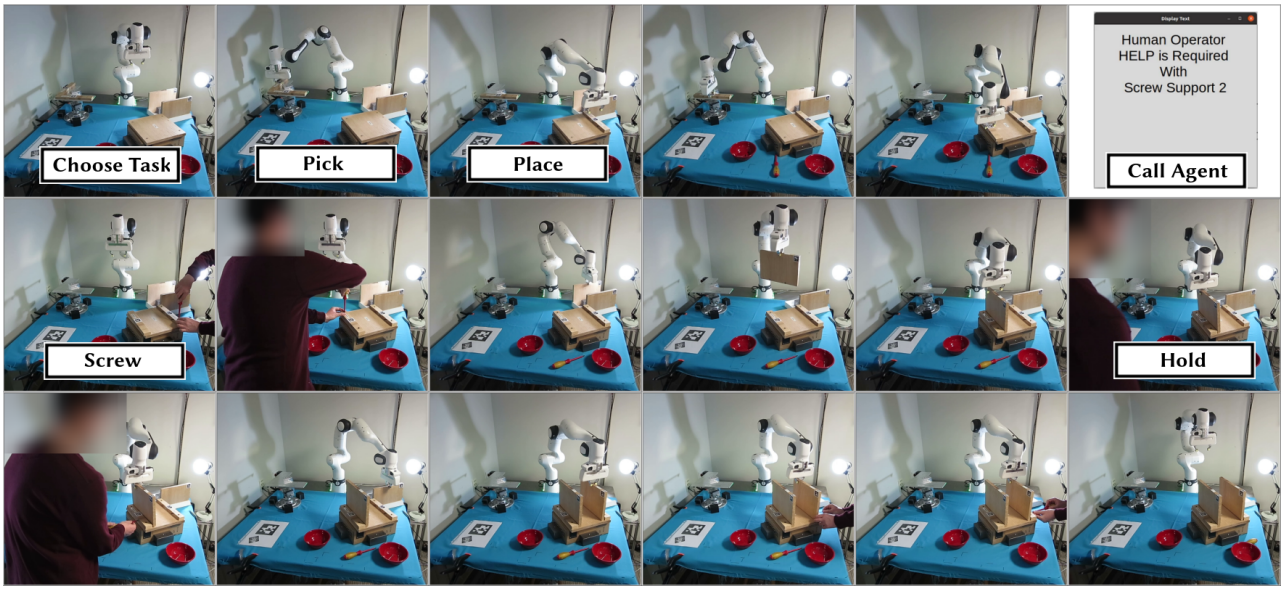


Fig. 4: Snapshots capturing key moments from a typical trial in the human-robot pilot study. At each step, the robotic agent executes actions based on its learned policy oDec-AIRL, while the human participant is encouraged to emulate the learned human policy. Examples of each action are illustrated upon their initial occurrence. Following the completion of the 'Place' action for Support1, the robot prompts for human assistance through a pop-up notification for the 'Call Agent' action.

TABLE II: **HRC Furniture Assembly learned policies comparison**

Average of 10000 timesteps				
Max Num Agents	Method	Num Steps Active Per Eps		Episode Reward
		Robot	Human	
2	**oDec-AIRL**	$\mathbf{19.61 \pm 0.94}$	$\mathbf{13.20 \pm 0.98}$	$\mathbf{4.06 \pm 0.58}$
	Dec-AIRL	16.34 ± 0.97	16.34 ± 0.97	1.74 ± 0.60

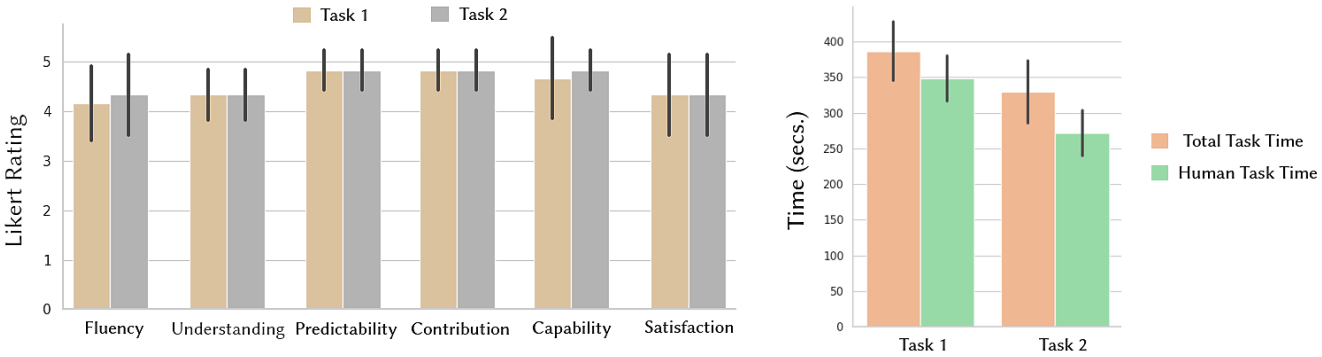


Fig. 5: *Left:* Findings for subjective measures on a 5-point scale. *Right:* The average total duration of tasks and the average time allocated to human agents starting from the Call Agent action.

4 times per episode to perform the screwing action. When the human is called in depends on the task order, which is randomly selected in each episode to maximize policy exploration. We compared their best-learned behaviors using average episode reward and the average number of timesteps the human and robot are present per episode. Notice that by following the Dec-AIRL policies, the human is active for the entire task duration, while with oDec-AIRL policies, the human is only active for 13.2 timesteps per episode as shown in Table II. Naturally, oDec-AIRL policies accrue a better episode reward than the baseline policies.

TABLE III: Subjective Measures

Fluency The robot and I collaborated fluently to accomplish the task.
Understanding I feel the robot had a good understanding of the task.
Predictability I was never surprised by the robot's actions.
Contribution The robot called for my assistance at the appropriate time.
Capability I feel that my time and effort were valued by the robot.
Satisfaction I feel satisfied with the performance of the system.

Real-world HRC experiment: In our pilot study, we conducted experiments where human subjects collaborated in assembling a table using a Franka Emika Research-3 7-DoF robotic arm. The robot's control stack was implemented as a finite state machine to manage and monitor its states and actions. Initially trained neural network policies were exported to a CSV file, which was then incorporated into the finite state machine to guide the robot's actions at each state. We recruited 6 participants under a within-subjects design. Participants received instructions on the assembly task, the robot's action capabilities, and its objectives. They were tasked to participate in the assembly and cooperate with the robot, especially when it requested assistance via a 'CallAgent' action displayed on an on-screen graphical interface popup. To aid in marker-based state estimation, we deployed an externally mounted Intel Realsense D435 camera with April Tags on the assembly components. Additionally, we integrated human hand tracking using a deep learning model [16], which was meticulously calibrated within the experimental setup. An AprilTag positioned just outside the workspace was used to signal the completion of actions, which participants activated after each task step.

For manipulated variables, we shuffled task sequences and selected two variations of the table assembly task. Consequently, each participant performed the assembly twice. Task 1 entailed positioning leg-support2 and screwing it into the base, then positioning leg2 and securing it into its support, and analogously for leg-support1 and leg1 to complete the task. Conversely, task 2 involved positioning leg-supports 2 and 1, screwing them into the base, positioning leg1, securing it into its support, and repeating the process for leg2 to complete the assembly. Our focus was on examining the framework for real-world open human-robot collaboration and determining the optimal timing of the 'CallAgent' action based on task variations. For other measures, we measured task completion times and the duration spent by the human across both tasks (Fig. 5, right). We created six statements for subjective evaluation (Table III), and asked participants to rate their level of agreement on a 5-point Likert scale (inspired by the Godspeed Series Questionnaire [17]).

All trials of the study were successful. Fig. 4 shows the Franka Research-3 robot assembling the table as per task order 2, calling the human for assistance at the appropriate time using the on-screen popup, after which the human and the robot collaboratively complete the rest of the assembly. Fig. 5 (left), shows the subjective ratings of the real-world experiments. Fig. 5 (right), shows the quantitative measures of the real-world experiments. Task 1 takes an average of 386.76 ± 41.19 secs for completion, while Task 2 takes 348.42 ± 32.28 secs. Through the 'Call Agent' action, on average, human agents only spend 329.49 ± 43.98 secs on Task 1 and 271.82 ± 31.55 secs on Task 2 demonstrating successful OHRC through an average time saving of approximately 18.39% for the human across both tasks.

V. Related Work

This work marks the initial endeavor within HRC to introduce a learn-from-observation approach for contexts involving agent openness (AO). Whereas methods tackling planning, modeling, and learning challenges in open-agent systems exist, we outline their limitations for OHRCS.

Previous methods addressing HRC tasks adopt different modeling approaches. Nikolaidis et al. [18] and Chen et al. [19] use a single-agent POMDP, while Giacomuzzo et al. [20] use a single-agent Descrete-Event MDP to represent domain. In later work, Nikolaidis et al. [21] employ a two-player Markov game to characterize HR teams, where humans adjust their behavior based on evolving perceptions of the robot's capabilities. Seo and Unhelkar [22] divided scenarios into an agent model and a task model. A centralized task model captures the task attributes while the agent Markov model represents the mental states of the other interacting agents. Wang et al. [23] employ an MMDP to simulate a handover and combined human-robot manipulation task, learning joint policies. More recent work [5], [24] utilizes a Dec-MDP to model HRC scenarios, where the human and robot are aware only of their local states. The robot policy is evaluated on a real-world collaborative produce sorting task. It is worth noting that these studies assumed closed systems for HRC, which limits their applicability to OHRCS.

More generally, Eck, Doshi and Soh [3] define open agent systems as a multiagent scenario where some elements of the system dynamically change over time. They outline three primary types of openness: agent openness (AO), where agents can join or leave during the task; type openness (TO), involving dynamic changes in the frames of agents present; and task openness (TaO), which allows changes in current tasks. Accurately representing open systems with

uncertainties regarding tasks, active agents, and their characteristics is paramount. Framed by these forms of openness, Chen et al. [25] investigates openness from an ad hoc standpoint, concluding through simulations that AO enhances performance whereas TaO complicates learning. Another work [7] employs the IPOMDP-Lite framework [26] to simulate AO, validating it in simulated scenarios such as wildfire suppression. A representation close to our oDec-MDP framework is the Open Dec-POMDP [27] to model AO. A key distinction is that they incorporate AO via coalitions, where the coalition transition function depends on the previous coalition and disregards the actions of agents. This assumption may be impractical when coalition transitions depend on the policies of agents. Another distinction is that the Open Dec-POMDP does not model state transitions due to team member's actions and team transitions.

To study AO in settings involving many agents, Eck et al. [8] enhances scalability by selectively modeling neighbors and extrapolating behavior from this modeling to others. A CI-POMDP model involving communication has also been utilized [6], incorporating agents' messages into belief updates validated via simulations. More recently, a RL-based decentralized actor-critic method based on the I-POMDP framework learns policies for open-organization challenges where employees may be hired or removed. Rahman et al. [28] proposes a partially observable open stochastic Bayesian game to model AO, employing graph-based policy learning evaluated across simulated domains. These approaches share a common drawback: they typically assess their methodologies using small, simulated toy environments, which may not effectively scale to real-world scenarios. Techniques relying on I-POMDPs or game theory may not be adequate for collaborative teamwork, while those for ad hoc teamwork overlook the dynamic nature of agents entering and exiting tasks. Moreover, the absence of evaluation in physical systems like real robots makes it challenging to gauge their efficacy in operational HRC contexts.

VI. Conclusions and Future Work

In this paper, we addressed agent openness (AO) in human-robot collaboration systems, and introduced a novel multiagent model, oDec-MDP, and a new IRL technique, oDec-AIRL, which uses expert demonstrations to learn a reward function for solving OHRC problems. Future work may relax the assumption of full observability to handle environmental occlusions or sensor noise and consider a larger set of agents to validate scalability. We also plan to explore type openness (TO) in OHRC, such as how fatigue affects human collaboration levels. OHRCS enhances seamless and efficient collaboration between human and robotic agents, opening numerous future research directions.

References

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

[2] A. Dahiya, A. M. Aroyo, K. Dautenhahn, and S. L. Smith, "A survey of multi-agent human–robot interaction systems," *Robotics and Autonomous Systems*, vol. 161, p. 104335, 2023.

[3] A. Eck, L.-K. Soh, and P. Doshi, "Decision making in open agent systems," *AI Magazine*, 2023.

[4] C. V. Goldman and S. Zilberstein, "Decentralized control of cooperative agents," *JAIR*, 2003.

[5] P. Sengadu Suresh, Y. Gui, and P. Doshi, "Dec-airl: Decentralized adversarial irl for human-robot teaming," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023, pp. 1116–1124.

[6] A. Kakarlapudi, G. Anil, A. Eck, P. Doshi, and L.-K. Soh, "Decision-theoretic planning with communication in open multiagent systems," in *Uncertainty in Artificial Intelligence*. PMLR, 2022, pp. 938–948.

[7] M. Chandrasekaran, A. Eck, P. Doshi, and L. Soh, "Individual planning in open and typed agent systems," in *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, 2016, pp. 82–91.

[8] A. Eck, M. Shah, P. Doshi, and L.-K. Soh, "Scalable decision-theoretic planning in open and typed multiagent systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020.

[9] C. V. Goldman and S. Zilberstein, "Optimizing information exchange in cooperative multi-agent systems," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '03. New York, NY, USA: ACM, 2003.

[10] F. S. Melo and M. Veloso, "Decentralized mdps with sparse interactions," *Artificial Intelligence*, vol. 175, no. 11, pp. 1757–1789, 2011.

[11] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *International Conference on Learning Representations*. unknown, 2018, pp. 1–10.

[12] C. V. Goldman and S. Zilberstein, "Decentralized control of cooperative systems: Categorization and complexity analysis," *Journal of artificial intelligence research*, vol. 22, pp. 143–174, 2004.

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

[14] W. McGugan, *Beginning game development with Python and Pygame: from novice to professional*. Apress, 2007.

[15] A. Koul, "ma-gym: Collection of multi-agent environments based on openai gym." https://github.com/koulanurag/ma-gym, 2019.

[16] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020.

[17] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi, "Godspeed questionnaire series," *International Journal of Social Robotics*, 2008.

[18] S. Nikolaidis and J. Shah, "Human-robot teaming using shared mental models," *ACM/IEEE HRI*, 2012.

[19] M. Chen, S. Nikolaidis, H. Soh, D. Hsu, and S. Srinivasa, "Trust-aware decision making for human-robot collaboration: Model learning and planning," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 9, no. 2, pp. 1–23, 2020.

[20] G. Giacomuzzo, M. Terreran, S. Jain, and D. Romeres, "Decaf: a discrete-event based collaborative human-robot framework for furniture assembly," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[21] S. Nikolaidis, S. Nath, A. D. Procaccia, and S. Srinivasa, "Game-theoretic modeling of human adaptation in human-robot collaboration," in *Proceedings of the 2017 ACM/IEEE international conference on human-robot interaction*, 2017, pp. 323–331.

[22] S. Seo and V. V. Unhelkar, "Semi-supervised imitation learning of team policies from suboptimal demonstrations," *arXiv preprint arXiv:2205.02959*, 2022.

[23] C. Wang, C. Pérez-D'Arpino, D. Xu, L. Fei-Fei, K. Liu, and S. Savarese, "Co-gail: Learning diverse strategies for human-robot collaboration," in *Conference on Robot Learning*. PMLR, 2022.

[24] P. S. Suresh and P. Doshi, "Marginal map estimation for inverse rl under occlusion with observer noise," in *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

[25] B. Chen, X. Chen, A. Timsina, and L.-K. Soh, "Considering agent and task openness in ad hoc team formation." in *AAMAS*, 2015.

[26] T. N. Hoang and K. H. Low, "Interactive pomdp lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents," *arXiv preprint arXiv:1304.5159*, 2013.

[27] J. Cohen, J.-S. Dibangoye, and A.-I. Mouaddib, "Open decentralized pomdps," in *IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2017.

[28] A. Rahman, I. Carlucho, N. Höpner, and S. V. Albrecht, "A general learning framework for open ad hoc teamwork using graph-based policy learning," *Journal of Machine Learning Research*, vol. 24, 2023.