Influencing Leading and Following in Human-Robot Teams

Minae Kwon*
Stanford University

Mengxi Li*
Stanford University

Alexandre Bucquet
Stanford University

Dorsa Sadigh
Stanford University

***Abstract*—Recent efforts in human-robot interaction have been focused on modeling and interacting with single human agents. However, when modeling teams of humans, current models are not able to capture underlying emergent dynamics that define group behavior, such as leading and following. We introduce a mathematical framework that enables robots to influence human teams by modeling emergent leading and following behaviors. We tackle the task in two steps. First, we develop a scalable representation of latent leading-following structures by combining model-based methods and data-driven techniques. Second, we optimize for a robot policy that leverages this representation to influence a human team toward a desired outcome. We demonstrate our approach on three tasks where a robot optimizes for changing a leader-follower relationship, distracting a team, and leading a team towards an optimal goal. Our evaluations show that our representation is scalable with different human team sizes, generalizable across different tasks, and can be used to design meaningful robot policies.**

I. Introduction

Humans are capable of seamlessly interacting and collaborating with each other. They can easily form teams and decide if they should follow or lead to efficiently complete a task as a group [39, 41, 12]. This is apparent in sports teams, human driving behavior, or simply having two people move a table together. Similarly, humans and robots are expected to seamlessly interact with each other to achieve collaborative tasks. Examples include collaborative manufacturing, search and rescue missions, and in an implicit way, collaborating on roads shared by self-driving and human-driven cars.

These collaborative settings bring two important challenges for robots. First, robots need to understand the emergent leading and following dynamics that underlie seamless human coordination. Second, robots must be able to influence human teams to achieve a desired goal. For instance, imagine a mixed human-drone search-and-rescue team headed toward the island on the left as shown in Fig. 1. When a drone senses that the survivors are actually on the island on the right, how should it direct the rest of the human team toward the desired goal?

Using natural language or pre-defined, task-specific communication signals (e.g., blinking lights) are viable solutions [42, 5]. However, in this work, we focus on finding more general solutions that do not rely on the assumption that there will be direct channels of communication.

One common solution is to assign leading and following roles to the team a priori before starting the team's mission. Many current human-robot interactions determine leader-follower roles beforehand [16, 22, 26, 40, 43, 17, 34], which include learning-from-demonstration tasks where the human is the leader and the robot is the follower [10, 3, 1, 44, 13, 8, 28], assistive tasks where the robot leads by teaching and assisting human users [33, 20], or other interactions based on game theoretic formulations where the robot leads by influencing its human partner [37, 38]. However, assigning leadership roles a priori is not always feasible in dynamically changing environments or long-term interactions. For instance, the drone that collects new information should be able to dynamically become a leader.

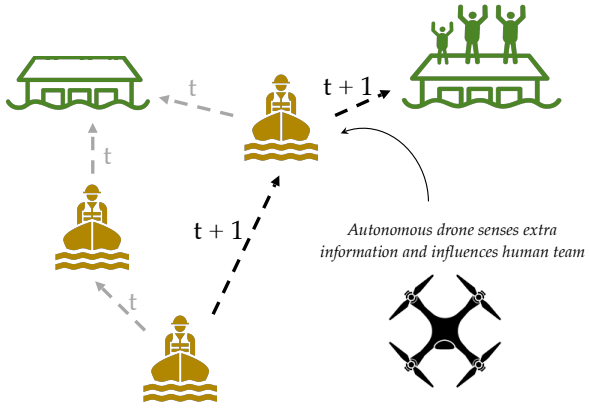


Fig. 1: We learn leading and following relationships in human teams (denoted by the arrows), and use this to create influential robot policies. The grey arrows represent intended human leading and following behaviors whereas the black arrows represent updated leading and following behaviors after the influencing robot action.

If leading and following roles are not assigned a priori, the robot must estimate human state and be equipped with policies that can influence teammates. In addition to the learning from demonstration work mentioned above, prior work has shown that robots have been able to infer human preferences through interactions using partially observable Markov decision processes (POMDPs) which allow reasoning over uncertainty on the human's internal state or intent [9, 14, 25, 27, 19, 36]. Human intent inference has also been achieved through human-robot cross-training [28] as well as various other approximations to POMDP solutions such as augmented MDPs, belief space planning, approximating reachable belief space, and decentralization [2, 23, 24, 31, 32, 35]. These approaches do not easily generalize to larger teams of humans due to computational intractability. Prior work has also studied

*Authors have contributed equally.

how we can construct intelligent robot policies that induce desired behaviors from people [38, 18, 29, 30, 7]. However, all of these works optimize for robot policies that influence only a single human. Human state estimation and optimization for actions based on the estimation will be computationally infeasible with larger groups of humans.

Instead of keeping track of each individual's state in a team, we propose a more scalable method that estimates the collective *team's* state. *Our key insight is that, similar to individuals, teams exhibit behavioral patterns that robots can use to create intelligent influencing policies.* One particular behavioral pattern we focus on are *leading and following relationships*.

In this paper, we introduce a framework for extracting underlying leading and following structures in human teams that is scalable with the number of human agents. We call this structure a *leader-follower graph*. This graph provides a concise and informative representation of the current state of the team and can be used when planning. We then use the leader-follower graph to optimize for robot policies that can influence a human team to achieve a goal.

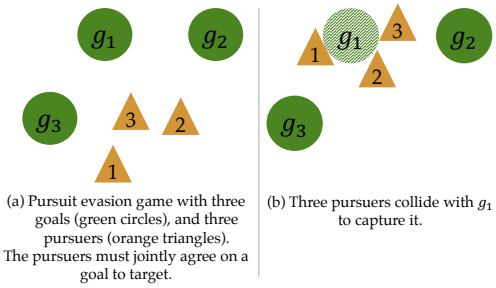
Our contributions in this paper are as follows:

- Formalizing and learning a scalable structure, *leader-follower graph*, that captures complex leading and following relationships between members in human teams.
- Developing optimization-based robot strategies that leverage the leader-follower graph to influence the team towards a more efficient objective.
- Providing simulation experiments in a pursuit-evasion game demonstrating the robot's influencing strategies to redirect a leader-follower relationship, distract a team, and lead a team towards an optimal goal based on its learned leader-follower graph.

II. Formalism for Modeling Leading and Following in Human Teams

Running Example: Pursuit-Evasion Game. We define a multi-player pursuit-evasion game on a 2D plane as our main running example. In this game, each pursuer is an agent in the set of agents I that can take actions in the 2D space to navigate. There are a number of stationary evaders, which we refer to as *goals*, G. The objective of the pursuers is to collaboratively capture the goals. Fig. 2 shows an example of a game with three pursuers, shown in orange, and three goals, shown in green. The action space of each agent is identical, $A_i = \{\text{move up, move down, move left, move right, stay still}\}$; the action spaces of all agents collectively define the joint action space A. All pursuers must jointly and implicitly agree on a goal to target without directly communicating with one another. A goal will be captured when all pursuers collide with it as shown in Fig. 2b.

Leaders and Followers. In order to collectively capture a goal, each agent $i \in I$ must decide to *follow* a goal or another agent. We refer to the goal or agent being followed as a *leader*. Formally we let $l_i \in G \cup I$, where l_i is either an agent or a fixed goal g who is the leader of agent i (agent i follows l_i). This is shown in Fig. 3a, where agent 2 follows goal g_1 ($l_2 = g_1$) and agent 3 follows agent 2 ($l_3 = 2$).



(a) Pursuit evasion game with three goals (green circles), and three pursuers (orange triangles). The pursuers must jointly agree on a goal to target.

(b) Three pursuers collide with g_1 to capture it.

Fig. 2: Pursuit-evasion game.

The set of goals G abstracts the idea of the agents reaching a set of states in order to fully optimize the joint reward function. For instance, in a pursuit-evasion game, all the agents need to reach a state corresponding to the goals being captured. An individual goal g intuitively signifies a way for the agents to coordinate strategies with each other. For instance, in a pursuit-evasion game, the agents should collaboratively plan on actions that capture each goal.

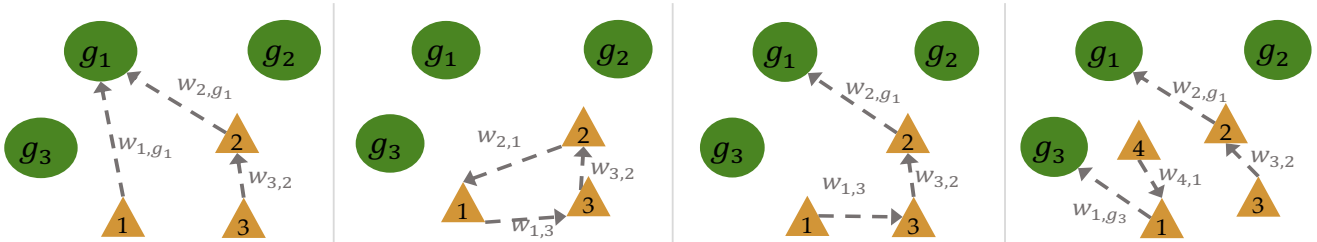
Leader-Follower Graph. The set of leaders and followers form a directed *leader-follower graph* as shown in Fig. 3a. Each node represents an agent $i \in I$ or goal $g \in G$. The directed edges represent leading-following relationships, where there is an outgoing edge from a follower to its leader. The weights on the edges represent a *leadership score*, which is the probability that the tail node is the head node's leader. For instance, in Fig. 3a, $w_{3,2}$ represents the probability that 2 is 3's leader. The leader-follower graph is dynamic in that agents can decide to change their leaders at any time. We assume that there could be an implicit transitivity in a leader-follower graph, i.e., if an agent i follows an agent j, implicitly it could be following the agent j's believed ultimate goal.

Some patterns are not desirable in a leader-follower graph. For instance, an agent would never follow itself, or we do not expect to observe cycling leading-following behaviors (Fig. 3b). Other patterns that are likely include: chain patterns (Fig. 3c) or patterns with multiple teams where multiple agents directly follow goals (Fig. 3d). We describe how to construct a leader-follower graph that is scalable with the number of agents and avoids the undesirable patterns in Sec. III.

Partial Observability. The leader of each agent, l_i, is a latent variable. We assume that agents cannot directly observe the one-on-one leading and following dynamics of other agents. Thus, constructing leader-follower graphs can help robot teammates predict who will follow whom, allowing them to strategically influence teammates to adapt roles. We assume agents are homogeneous and have full information on the observations of themselves and all other agents. (e.g. positions and velocities of agents).

III. Construction of a Leader-Follower Graph

In this section, we focus on constructing the leader-follower graph that emerges in collaborative teams using a combination of data-driven and graph-theoretic approaches. Our aim is to



(a) Leader-follower graph. Green circles are the goals that need to be captured. Orange triangles are the pursuers.

(b) Cyclic leader-follower graph. We design policies that avoid such cyclic behaviors.

(c) Chain behavior in the leader-follower graph.

(d) Multiple teams.

Fig. 3: Variations of the leader-follower graph

leverage this leader-follower graph to enable robot teammates to produce helpful leading behaviors. We will first focus on learning pairwise relationships between agents using a supervised learning approach. We then generalize our ideas to multi-player settings using algorithms from graph-theory. Our combination of data-driven and graph-theoretic approaches allows the leader-follower graph to efficiently scale with the number of agents.

A. *Pairwise Leadership Scores*

We first will focus on learning the probability of any agent i following any goal or agent $j \in G \cup I$. The pairwise probabilities help us estimate the leadership score $w_{i,j}$, i.e., the weight of the edge (i, j) in the leader-follower graph.

We propose a general framework of estimating the leadership scores using a supervised learning approach. Consider a two-player setting where $I = \{i, j\}$, we collect labeled data where agent i is asked to follow j, and agent j is asked to optimize for the joint reward function assuming it is leading i, i.e., following a fixed goal g in the pursuit-evasion game ($l_i = j$ and $l_j = g$). We then train a Long Short-Term Memory (LSTM) network with a softmax layer to predict each agent's most likely leader.

Training with a Scalable Network Architecture. Our network architecture consists of two LSTM submodules, one to predict player-player leader-follower relationships (P-P LSTM) and one to predict player-evader relationships (P-E LSTM). We use a softmax output layer with a cross-entropy loss function to get a probability distribution over j and all goals $g \in G$ of being i's leader. We take the leader (an agent or a goal) with the highest probability and assign this as the leadership score. The P-P and P-E submodules allow us to scale training to a game of any number of players and evaders as we can add or remove P-P and P-E submodules depending on the number of players and evaders in a game. An example of our scalable network architecture is illustrated in Fig. 4.

Data Collection. We collect labeled human data by asking participants to play a pursuit evasion game with pre-assigned leaders. We recruited pairs of humans and randomly assigned leaders l_i to them (i.e., another agent or a goal). Participants played the game in a web browser using their arrow keys and were asked to move toward their assigned leader, l_i. In order to create a balanced dataset, we collected data from all possible

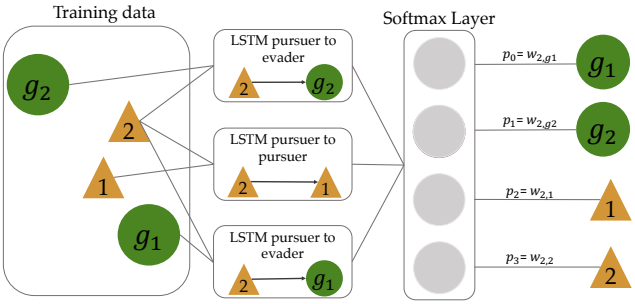


Fig. 4: Scalable neural network architecture. This example is predicting the probability of another agent j being agent 2's leader, $w_{2,j}$. There are three LSTM submodules used because there are two possible evaders and one possible agent that could be agent 2's leader.

configurations of leader and followers in a two-player setting. We collected a total of 186 games.

In addition, since human data is often difficult to collect in large amounts; we augmented our dataset with synthetic data. The synthetic data is generated by simulating humans playing the pursuit evasion game. We simulated humans based on the well-known potential field path planner model [6]. We find that the simple nature of the task given to humans (i.e., move directly toward your assigned leader l_i) is qualitatively easily replicated using a potential field path planner. Agents at location q plan their path under the influence of an artificial potential field $U(q)$, which is constructed to reflect the environment. Agents move toward their leaders l_i by following an attractive potential field. Other agents and goals that are not their leaders are treated as obstacles and emit a repulsive potential field.

We denote the set of attractions $i \in A$ and the set of repulsive obstacles $j \in R$. The overall potential field is weighted sum of potential field from all attractions and repulsive obstacles. θ_i is the weight for the attractive potential field from attraction $i \in A$, and θ_j is the weight for repulsive potential field from obstacle $j \in R$.

$$U(q) = \sum_{i \in A} \theta_i U^i_{att}(q) + \sum_{j \in R} \theta_j U^j_{rep}(q) \tag{1}$$

The optimal action a an agent would take lies in the direction

of the potential field gradient:

$$a = -\nabla U(q) = -\sum_{i \in A} \theta_i \nabla U^i_{att}(q) - \sum_{j \in R} \theta_j \nabla U^j_{rep}(q)$$

The attractive potential field increases as the distance to goal becomes larger. The repulsive potential field has a fixed effective range, within which the potential field increases as the distance to the obstacle decreases. The attractive and repulsive potential fields are constructed in the same way for all attractive and repulsive obstacles. We elaborate on details of the potential field function in the Appendix, Section VII.

Implementation Details. We simulated 15000 two-player games, equally representing the number of possible leader and follower configurations. Each game stored the position of each agent and goal at all timesteps. Before training, we pre-processed the data by shifting it to have a zero-centered mean, normalizing it, and down-sampling it. Each game was then fed into our network as a sequence. Based on our experiments, hyperparameters that worked well included a batch size of 250, learning rate of 0.0001 and hidden dimension size of 64. In addition, we used gradient clipping and layer normalization [4] to stabilize gradient updates. Using these hyperparameters, we trained our network for 100,000 steps.

Evaluating Pairwise Scores. Our network trained on two-player simulated data performed with a validation accuracy of 83%. We also experimented with training with three-player simulated data, two-player human data, as well as a combination of two-player simulated and human data (two-player mixed data). Validation results for the first 60,000 steps are shown in Fig. 5. For our mixed two-player model, we first took a pre-trained model trained on two-player simulated data and then trained on two-player human data. The two-player mixed curve shows the model's accuracy as it is fined-tuned on human data. Our three-player model had a validation accuracy of 74% while our two player human model and our mixed two-player data model had a validation accuracy of 65%.

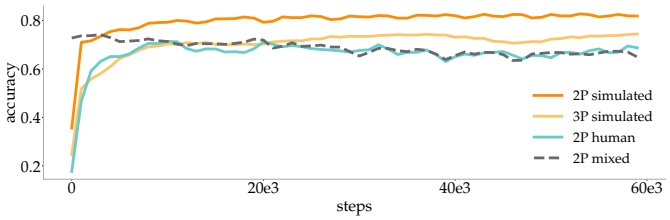


Fig. 5: Validation accuracy when calculating pairwise leadership scores, described in Sec. III-A.

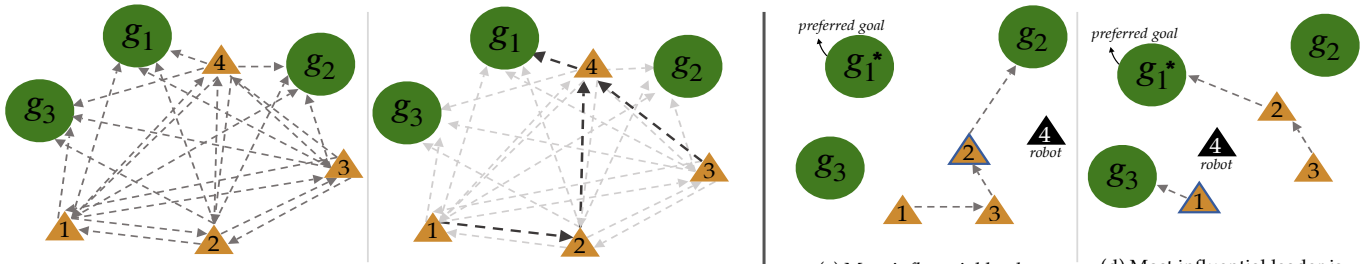
B. Maximum Likelihood Leader-Follower Graph in Teams

With a model that can predict pairwise relationships, we focus on building a model that can generalize beyond two players. We compute pairwise weights, or leadership scores, $w_{i,j}$ of leader-follower relationships between all possible pairs of leaders i and followers j. The pairwise weights can be computed based on the supervised learning approach described above, indicating the probability of one agent or goal being another agent's leader. After computing $w_{i,j}$ for all combinations of leaders and followers, we create a directed graph $\mathcal{G} = (V, E)$ where $V = I \cup G$ and $E = \{(i,j) | i \in I, j \in I \cup G, i \neq j\}$, and the weights on each edge (i,j) correspond to $w_{i,j}$. To create a more concise representation, we extract the maximum likelihood leader-follower graph $\mathcal{G}^*$ by pruning the edges of our constructed graph $\mathcal{G}$. We prune the graph by greedily selecting the outgoing edge with highest weight for each agent node. In other words, we select the edge associated with the agent or goal that has the highest probability of being agent i's leader, where the probabilities correspond to edge weights. When pruning, we make sure that no cycles are formed. If we find a cycle, we will choose the next probable edge. Our pruning approach is inspired by Edmonds' algorithm [15, 11], which finds a maximum weight arborescence [21] in a graph. An arborescence is an acyclic directed tree structure, where there is exactly one outgoing edge from a node to another. We use a modified version of Edmonds' algorithm to find the maximum likelihood leader-follower graph. Compared to our approach, a maximum weight arborescence is more restrictive since it requires the resulting graph to be a tree.

Evaluating the Leader-Follower Graph. We evaluate the accuracy of our leader-follower graph with three or more agents when trained on simulated two-player and three-player data, as well as a combination of simulated and human two-player data (shown in Table I). In each of these multi-player games, we extracted a leader-follower graph at each timestep and compared our leader-follower graph's predictions against the ground-truth labels. Our leader-follower graph performs better than a random policy, which selects a leader $l_i \in I \cup G$ for agent i at random, where $l_i \neq i$. The chance of being right is thus $\frac{1}{|G|+|I|-1}$. We take the average of all success probabilities for all leader-follower graph configurations to compute the overall accuracy. In all experiments shown in Table I, our trained model outperforms the random policy. Most notably, the models scale naturally to settings with large numbers of players as well as human data. We find that training with larger numbers of players helps with generalization. The tradeoff between training with larger numbers of players and collecting more data will depend on the problem domain. For our experiments (Section V), we use the model trained on three-player simulated data.

TABLE I: Generalization accuracy of leader-follower graph (LFG)

Training Data	Testing Data	LFG Accuracy	Random Accuracy
2 players, simulated	3 players, simulated	0.67	0.29
	4 players, simulated	0.45	0.23
	5 players, simulated	0.41	0.19
	2 players, human	0.68	0.44
	3 players, human	0.47	0.29
2 players, mixed	3 players, simulated	0.44	0.29
	4 players, simulated	0.38	0.23
	5 players, simulated	0.28	0.19
	2 players, human	0.69	0.44
	3 players, human	0.44	0.29
3 players, simulated	4 players, simulated	0.53	0.23
	5 players, simulated	0.50	0.19
	3 players, human	0.63	0.29



(a) Graph $\mathcal{G}$. The directed edges represent pairwise likelihoods that the tail node is the head node's leader.

(b) Maximum-likelihood leader-follower graph, $\mathcal{G}^*$shown in bold.

(c) Most influential leader is agent 2.

(d) Most influential leader is agent 1 since other human agents are targeting the preferred goal.

Fig. 6: On left: creating a maximum-likelihood leader-follower graph, $\mathcal{G}^*$. On right: examples of influential leaders in $\mathcal{G}^*$. The green circles are goals, orange triangles are human agents and black triangles are robot agents.

IV. Planning based on Inference on Leader-Follower Graphs

We so far have computed a graphical representation for latent leadership structures in human teams $\mathcal{G}^*$. In this section, we will use $\mathcal{G}^*$ to positively influence human teams, i.e., move the team towards a more desirable outcome. We first describe how a robot can use the leader-follower graph to infer useful team structures. We then describe how a robot can leverage these inferences to plan for a desired outcome.

A. Inference based on leader-follower graph

Leader-follower graphs enable a robot to infer useful information about a team such as agents' goals or who the most influential leader is. These pieces of information help the robot to identify key goals or agents that are useful in achieving a desired outcome (e.g., identifying shared goals in a collaborative task). Especially in multi-agent settings, following key goals or influencing key agents is a strategic move that allows the robot to plan for desired outcomes. We begin by describing different inferences a robot can perform on the leader-follower graph.

Goal inference in multiagent settings. One way a robot can use structure in the leader-follower graph is to perform goal inference. An agent's goal can be inferred by the outgoing edges from agents to goals. In the case where there is an outgoing edge from an agent to another agent (i.e., agent i follows agent j), we assume transitivity, where agent i can be implicitly following agent j's believed ultimate goal.

Influencing the most influential leader. In order to lead a team toward a desired goal, the robot can also leverage the leader-follower graph to predict who the *most influential leader* is. We define the most influential leader to be the agent $i^* \in I$ with the most number of followers. Identifying the most influential leader i^* allows the robot to strategically influence a single teammate that also indirectly influences the other teammates that are following i^*. For example, in Fig. 6c and Fig. 6d, we show two examples of identifying the most influential leader from $\mathcal{G}^*$. In the case where some of agents are already going for the preferred goal, the one that has the most followers among the remaining players becomes the most influential leader, as shown in Fig. 6d.

B. Optimization based on leader-follower graph

We leverage the leader-follower graph to design robot policies that optimize for an objective (e.g., leading the team towards a preferred goal). We introduce one such way we can use the leader-follower graph by directly incorporating inferences we make from it into a simple optimization.

At each timestep, we generate graphs $\mathcal{G}^{a_t}_{t+k}$ that simulate what the leader-follower graph would look like at timestep $t+k$ if the robot takes an action a_t at current timestep t. Over the next k steps, we assume human agents will continue along the current trajectory with constant velocity.

From each graph $\mathcal{G}^{a_t}_{t+k}$, we can obtain the weights $w^{t+k}_{i,j}$ corresponding to an objective r that the robot is optimizing for (e.g., the robot becoming agent i's leader). We then iterate over the robot's action, assuming it is finite and discrete, and choose the action a^*_t that maximizes the objective r. We note that r must be expressible in terms of $w^{t+k}_{i,j}$'s and $w^{t+k}_{i,g}$'s for $i, j \in \mathcal{I}$ and $g \in G$.

$$a^*_t = \underset{a_t \in \mathcal{A}}{\text{argmax}} \quad r\left(\{w^{t+k}_{i,j}(a_t)\}_{i,j\in I}, \{w^{t+k}_{i,g}(a_t)\}_{i\in I, g\in G}\right) \tag{2}$$

We describe three specific tasks that we plan for using the optimization described in Eqn. (2).

Redirecting a leader-follower relationship. A robot can directly influence team dynamics by changing leader-follower relationships. For a given directed edge between agents i and j, the robot can use the optimization outlined in Eqn. (2) for actions that reverse an edge or direct the edge to a different agent. For instance, to reverse the direction of the edge from agent i to agent j, the robot will select actions that maximize the probability of agent j following agent i:

$$a^*_t = \underset{a_t \in \mathcal{A}}{\text{argmax}} \quad w^{t+k}_{j,i}(a_t), \;\; i, j \in I \tag{3}$$

The robot can also take actions to eliminate an edge between agents i and j by *minimizing* $w_{i,j}$. One might want to modify edges in the leader-follower graph when trying to change the leadership structure in a team. For instance, in a setting where agents must collectively decide on a goal, a robot can help unify a team with sub-groups (an example is shown in Fig. 3d) by re-directing the edges of one sub-group to follow another.

Distracting a team. In adversarial settings, a robot might want to prevent a team of humans from reaching a collective goal g. In order to stall the team, a robot can use the leader-follower graph to identify who the current most influential leader i^* is. The robot can then select actions that maximize the probability of the robot becoming the most influential leader's leader and minimize the probability of the most influential leader following the collective goal g:

$$a_t^* = \underset{a_t \in \mathcal{A}}{\text{argmax}} \quad w_{i^* r}^{t+k}(a_t), \; i^* \in \mathcal{I} \tag{4}$$

Distracting a team from reaching a collective goal can be useful in cases where the team is an adversary. For instance, a team of adversarial robots may want to prevent their opponents from collaboratively reaching a joint goal.

Leading a team towards the optimal goal. In collaborative settings where the team needs to agree on a goal $g \in G$, a robot that knows where the optimal goal $g^* \in G$ is should maximize joint utility by leading all of its teammates to reach g^*. To influence the team, the robot can use the leader-follower graph to infer who the current most influential leader i^* is. The robot can then select actions that maximize the probability of the most influential leader following the optimal goal g^*:

$$a_t^* = \underset{a_t \in \mathcal{A}}{\text{argmax}} \quad w_{i^* r}^{t+k}(a_t) + w_{r^* g^*}^{t+k}(a_t), \; i^* \in \mathcal{I} \tag{5}$$

Being able to lead a team of humans to a goal is useful in many real-life scenarios. For instance, in search-and-rescue missions, robots with more information about the location of survivors should be able to lead the team in the optimal direction.

V. Experiments

With an optimization framework that plans for robot actions using the leader-follower graph, we evaluate our approach on the three tasks described in Sec. IV. For each task, we compare task performance with robot policies that use the leader-follower graph against robot policies that do not. Across all tasks, we find that robot policies that use the leader-follower graph perform well compared to other policies, showing that our graph can be easily generalized to different settings.

Task setup. Our tasks take place in the pursuit-evasion domain. Within each task, we conduct experiments with simulated human behavior. Simulated humans move along a potential field as shown in Eqn. 1, where there are two sources of attraction: the agent's goal (a_g) and the crowd center (a_c), and simulated human agents would make trade offs between following the crowd and moving toward a target goal. The weights θ_g and θ_c encode how determined human players are about reaching the goal or moving towards the crowd. Higher θ_c means that players care more about being close to the whole team, while higher θ_g indicates more determined players who care about reaching their goals. This can make it harder for a robot to influence the humans. For validating our framework, we use weights θ_g = 0.6 and θ_c = 0.4, where going toward the goal is slightly more advantageous than moving with the whole team, setting moderate challenges for the robot to complete tasks. At the same time, we prevent agents from colliding with one another by making non-target agents and goals sources of repulsion with weight $\theta_j = 1$.

An important limitation is that our simulation does not capture the broad range of behaviors humans would show in response to a robot that is trying to influence them. We only offer one simple interpretation of how humans would react, based on the potential field.

In each iteration of the task, the initial position of agents and goals are randomized. For all of our experiments, the size of the game canvas is 500×500 pixels. At every time step, a simulated human can move 1 unit in one of the four directions: up, down, left, right, or stay at its position. The robot agent's action space is the same but can move 5 units at a time. The maximum game time limit for all our experiments is 1000 timesteps. For convenience, we will refer to *simulated human agents* as *human agents* for the rest of the section.

A. Changing edges of the leader-follower graph

We evaluate a robot's ability to change an edge of a leader-follower graph. Unlike the other tasks we described, the goal of this task is not to affect some larger outcome in the game (e.g., influence humans toward a particular goal). Instead, this task serves as a preliminary to others where we evaluate how well a robot is able to manipulate the leader-follower graph.

Methods. Given a human agent i who is predisposed to following a goal with weights $\theta_g = 0.6$, $\theta_a = 0.4$, we created a robot policy that encouraged the agent to follow the robot r instead. The robot optimized for the probability $w_{i,r}$ that it would become agent i's leader, as outlined in Eqn. (3). We compared our approach against a random baseline, where the robot chooses its next action at random.

Metrics. For both our approach and our baseline, we evaluated the performance of the robot based on the leadership scores, i.e., probabilities $w_{i,r}$, given by the leader-follower graph.

Results. We show that the robot can influence a human agent to follow it. Fig. 7 contains averaged probabilities over ten tasks. The probability of the robot being the human agent's leader $w_{i,r}$ increases over time, and averages to 73%, represented by the orange dashed line. Our approach performs well compared to the random baseline, which has an average performance of 26%, represented by the grey dashed line.

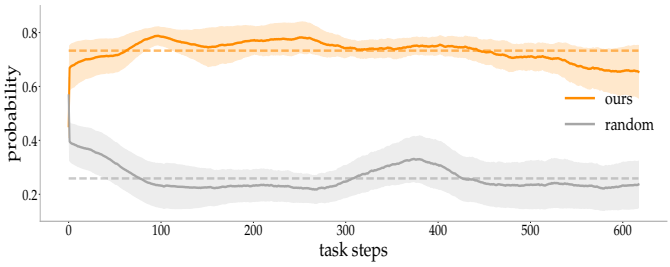


Fig. 7: Smoothed probabilities of a human agent following the robot over 10 tasks. The robot is quickly able to become the human agent's leader with an average of leadership score of 73%.

B. Adversarial task

We consider a different task where the robot is an adversary that is trying to distract a team of humans from reaching a goal. There are m goals and n agents in the pursuit-evasion game. Among the n agents, we have 1 robot and $n-1$ humans. The robot's goal is to distract a team of humans so that they cannot converge to the same goal. In order to capture a goal, we enforce the constraint that $n-2$ agents must collide with it at the same time, allowing 1 agent to be absent. Since we allow one agent to be absent, this prevents the robot from adopting the simple strategy of blocking an agent throughout the game. The game ends when all goals are captured or when the game time exceeds the limit. Thus, another way to interpret the robot's objective is to extend game time as much as possible.

Methods. We compared our approach against 3 baseline heuristic strategies (random, to_one_player, to_farthest_goal) that do not use the leader-follower graph.

In the *Random* strategy, the robot picks an action at each timestep with uniform probability. In the *To_one_player* strategy, the robot randomly selects a human agent and then moves towards it to try to block its way. Finally, a robot enacting the *To_farthest_goal* strategy selects the goal that the average distance to human players are largest and then go to that goal in the hope that human agents would get influenced or may further change goal by observing that some players are heading for another goal.

We also experimented with two variations of our approach. *LFG_closest_pursuer* involves the robot agent selecting the closest pursuer and choosing an action to maximize the probability of the pursuer following it (as predicted by the LFG). Similarly, *LFG_influential_pursuer* strategy involves the robot targeting the most influential human agent predicted by the LFG and then conducting the same optimization of maximizing the following probability, as shown in Eqn. (4).

Finally, we explored different game settings by varying the number of players $n = \{3, \ldots, 6\}$ and the number of goals $m = \{1, \ldots, 4\}$.

Metrics. We evaluated the performance of the robot with game time as the metric. Longer game time indicates that the robot does well in distracting human players.

Results. We sampled 50 outcomes for every game setting and robot policy. Tables II and III show the mean and standard deviation of game time for every game setting and robot policy across 50 samples. Across game settings, our models based on the leader-follower graph consistently outperformed methods without knowledge of leader-follower graph. These experimental results are also visualized in Fig. 8.

As can be seen from Fig. 8, the average game time goes up as the number of players increases. This is because it is more challenging for more players to agree upon on which goal to capture. The consistent advantageous performance across all game settings suggests the effectiveness of the leader-follower graph for inference and optimization in this scenario.

We demonstrate an example of robot behavior in the adversarial game, shown in Fig. 9 (dashed lines represent trajectories). Player 1 and player 2 start very close to goal g_1, making g_1 the probable target. The robot then approaches agent 2, blocks its way, and leads it to another goal g_2. In this way, the robot successfully extends the game time.

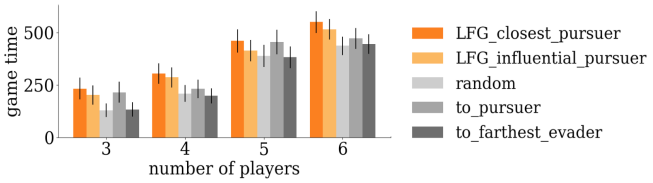


Fig. 8: Average game time over 50 games with a different number of players across all baseline methods and our model.

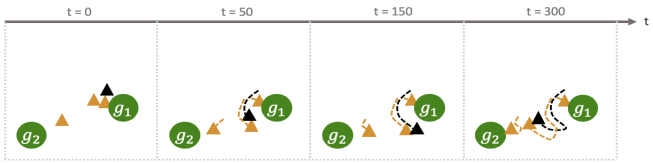


Fig. 9: Adversarial task snapshots. The orange triangles are the human agents and the black triangle is the robot agent.

C. Cooperative task

Finally, we evaluate the robot in a cooperative setting where the robot tries to be helpful for human teams. The robot aims to lead its teammates in a way that everyone can reach the target goal that gives the team the greatest joint utility $g^* \in G$. However, g^* is not immediately observable to all teammates. We assume a setting where only the robot knows where g^* is.

For consistency, the experiment setting is the same as the *Adversarial task* where $n-2$ human agents need to collide with a goal to capture it. In this scenario, the task is considered successful if the goal with greatest joint utility g^* is captured. The task is unsuccessful if any other suboptimal goal is captured or if the game time exceeds the limit.

Methods. Similar to the *Adversarial task*, we explore two variations of our approach where the robot chooses to influence its closest human agent or the most influential agent predicted by the leader-follower graph. The variation where the robot targets the most influential leader is outlined in Eqn. (5). Different from the *Adversarial task*, here, the robot is optimizing the probability of both becoming the target agent's leader and going toward the target goal. We also experimented with three baseline strategies. The *Random* strategy involves the robot taking actions at random. The *To_target_goal* strategy has the robot move directly to the optimal goal g^*, and stay there, trying to attract other human agents. The *To_goal_farthest_player* strategy involves the robot going to the player that is farthest away from g^* in an attempt to indirectly influence the player to move back to g^*. Finally, we experimented with game settings by varying the number of goals $m = \{2, \ldots, 6\}$.

Metrics. We evaluated the performance of the robot strategy using the game success rate over 100 games. We calculate the success rate by dividing the number of successful games over number of total games.

TABLE II: Average Game time over 50 adversarial games with varying number of players

number of goals (m=2)				
Model	n=3	n=4	n=5	n=6
LFG_closest_pursuer (**ours**)	**233.04±51.82**	**305.08±49.48**	**461.18±55.73**	**550.88±51.67**
LFG_influential_pursuer (**ours**)	201.94±45.15	286.44±48.54	414.78±50.98	515.92±48.80
random	129.2±32.66	209.40±39.86	388.92±53.24	437.16±43.17
to_one_pursuer	215.04±50.00	231.42±44.69	455.16±58.35	472.36±49.75
to_farthest_goal	132.84±34.22	198.5±36.14	382.08±52.59	445.64±46.77

TABLE III: Average Game time over 50 adversarial games with varying number of goals

number of players (n=4)				
Model	m=1	m=2	m=3	m=4
LFG_closest_pursuer (**ours**)	210.94±33.23	**305.08±49.48**	**289.22±52.99**	**343.00±55.90**
LFG_influential_pursuer (**ours**)	**239.04±39.73**	286.44±48.54	219.56±41.00	301.80±52.00
random	155.94±21.42	209.40±39.86	205.74±43.05	294.62±54.01
to_one_pursuer	123.58±9.56	231.42±44.69	225.52±41.47	317.92±54.75
to_farthest_goal	213.36±34.83	198.5±36.14	218.68±43.67	258.30±50.64

Results. Our results are summarized in Table IV. We expected the *To_target_goal* baseline to be very strong, since moving towards the target goal directly influences other simulated agents to also move toward the target goal. We find that this baseline strategy is especially effective when the game is not complex, i.e., the number of goals is small. However, our model based on the leader-follower graph still demonstrates competitive performance compared to it. Especially when the number of goals increase, the advantage of leader-follower graph becomes clearer. This indicates that, in complex scenarios, brute force methods that do not have knowledge of human team hidden structure will not suffice. Another thing to note is that the difference between all of the strategies becomes smaller as the number of goals increases. This is because the difficulty of the game increases for all strategies, and thus whether a game would succeed depends more on the game initial conditions. We have shown snapshots of one game

TABLE IV: Success rate over 100 collaborative games with varying number of goals m.

number of players (n=4)					
Model	m=2	m=3	m=4	m=5	m=6
LFG_closest_pursuer	0.59	0.38	0.29	**0.27**	**0.22**
LFG_influential_pursuer	0.57	0.36	**0.32**	0.24	0.19
random	0.55	0.35	0.24	0.21	0.20
to_target_goal	**0.60**	**0.42**	0.28	0.24	0.21
to_goal_farthest_player	0.47	0.29	0.17	0.19	0.21

as in Fig. 10. In this game, the robot approaches other agents and the desired goal in the collaborative pattern, trying to help catch the goal g_1.

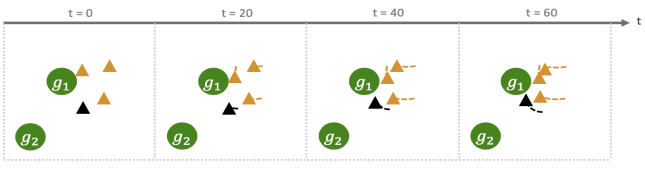


Fig. 10: Collaborative task snapshots. The orange triangles are the human agents and the black triangle is the robot agent.

VI. Discussion

Summary. We propose an approach for modeling leading and following behavior in multi-agent human teams. We use a combination of data-driven and graph-theoretic techniques to learn a leader-follower graph. This graph representation encoding human team hidden structure is scalable with the team size (number of agents), since we first learn *local*, pairwise relationships and combine them to create a *global* model. We demonstrate the effectiveness of the leader-follower graph by experimenting with optimization based robot policies that leverage the graph to influence human teams in different scenarios. Our policies are general and perform well across all tasks compared to other high-performing task-specific policies.

Limitations and Future Work. We view our work as a first step into modeling latent, dynamic human team structures. Perhaps our greatest limitation is the reliance on simulated human behavior to validate our framework. Further experiments with real human data are needed to support our framework's effectiveness for noisier human behavior understanding. Moreover, the robot policies that use the leader-follower graphs are fairly simple. Although this may be a limitation, it is also promising that simple policies were able to perform well using the leader-follower graph.

For future work, we plan to evaluate our model on large scale human-robot experiments in both simulation and navigation settings such as robot swarms to further improve our model's generalization capacity. We also plan on experimenting with combining the leader-follower graph with more advanced policy learning methods such as deep reinforcement learning. We think the leader-follower graph could contribute to multi-agent reinforcement learning in various ways such as reward design and state representation.

Acknowledgements. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. The authors would also like to acknowledge General Electric (GE) and Stanford School of Engineering Fellowship.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Ali-Akbar Agha-Mohammadi, Suman Chakravorty, and Nancy M Amato. Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 33(2):268–304, 2014.

[3] Baris Akgun, Maya Cakmak, Karl Jiang, and Andrea L Thomaz. Keyframe-based learning from demonstration. *International Journal of Social Robotics*, 4(4):343–355, 2012.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] Kim Baraka, Ana Paiva, and Manuela Veloso. Expressive lights for revealing mobile service robot state. In *Robot 2015: Second Iberian Robotics Conference*, pages 107–119. Springer, 2016.

[6] Jerome Barraquand, Bruno Langlois, and J-C Latombe. Numerical potential field techniques for robot path planning. *IEEE transactions on systems, man, and cybernetics*, 22(2):224–241, 1992.

[7] Aaron Bestick, Ruzena Bajcsy, and Anca D Dragan. Implicitly assisting humans to choose good grasps in robot to human handovers. In *International Symposium on Experimental Robotics*, pages 341–354. Springer, 2016.

[8] Erdem Biyik and Dorsa Sadigh. Batch active preference-based learning of reward functions. In *Conference on Robot Learning (CoRL)*, October 2018.

[9] Frank Broz, Illah Nourbakhsh, and Reid Simmons. Designing pomdp models of socially situated tasks. In *RO-MAN, 2011 IEEE*, pages 39–46. IEEE, 2011.

[10] Sonia Chernova and Andrea L Thomaz. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121, 2014.

[11] Yoeng-Jin Chu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.

[12] D Scott DeRue. Adaptive leadership theory: Leading and following as a complex adaptive process. *Research in organizational behavior*, 31:125–150, 2011.

[13] Anca D Dragan Dorsa Sadigh, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017.

[14] Finale Doshi and Nicholas Roy. Efficient model learning for dialog management. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 65–72. ACM, 2007.

[15] Jack Edmonds. Optimum branchings. *Mathematics and the Decision Sciences, Part*, 1(335-345):25, 1968.

[16] Katie Genter, Noa Agmon, and Peter Stone. Ad hoc teamwork for leading a flock. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 531–538. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[17] Matthew C Gombolay, Cindy Huang, and Julie A Shah. Coordination of human-robot teaming with human task preferences. In *AAAI Fall Symposium Series on AI-HRI*, volume 11, page 2015, 2015.

[18] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In *Advances in neural information processing systems*, pages 3909–3917, 2016.

[19] Shervin Javdani, Henny Admoni, Stefania Pellegrinelli, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared autonomy via hindsight optimization for teleoperation and teaming. *The International Journal of Robotics Research*, page 0278364918776060, 2018.

[20] Takayuki Kanda, Takayuki Hirano, Daniel Eaton, and Hiroshi Ishiguro. Interactive robots as social partners and peer tutors for children: A field trial. *Human-computer interaction*, 19(1):61–84, 2004.

[21] Richard M Karp. A simple derivation of edmonds' algorithm for optimum branchings. *Networks*, 1(3):265–272, 1971.

[22] Piyush Khandelwal, Samuel Barrett, and Peter Stone. Leading the way: An efficient multi-robot guidance system. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, pages 1625–1633. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[23] Mykel J Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

[24] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland., 2008.

[25] Oliver Lemon. Conversational interfaces. In *Data-Driven Methods for Adaptive Spoken Dialogue Systems*, pages 1–4. Springer, 2012.

[26] Michael L Littman and Peter Stone. Leading best-response strategies in repeated games. In *In Seventeenth Annual International Joint Conference on Artificial Intelligence Workshop on Economic Agents, Models, and Mechanisms*. Citeseer, 2001.

[27] Owen Macindoe, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Pomcop: Belief space planning for sidekicks in cooperative games. In *AIIDE*, 2012.

[28] Stefanos Nikolaidis and Julie Shah. Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 33–40. IEEE Press, 2013.

[29] Stefanos Nikolaidis, Anton Kuznetsov, David Hsu, and Siddharta Srinivasa. Formalizing human-robot mutual adaptation: A bounded memory model. In *The Eleventh ACM/IEEE International Conference on Human Robot*

Interaction, pages 75–82. IEEE Press, 2016.

[30] Stefanos Nikolaidis, Swaprava Nath, Ariel D Procaccia, and Siddhartha Srinivasa. Game-theoretic modeling of human adaptation in human-robot collaboration. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 323–331. ACM, 2017.

[31] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Christopher Amato, and Jonathan P How. Decentralized control of partially observable markov decision processes using belief space macro-actions. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5962–5969. IEEE, 2015.

[32] Robert Platt Jr, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. 2010.

[33] Ben Robins, Kerstin Dautenhahn, Rene Te Boekhorst, and Aude Billard. Effects of repeated exposure to a humanoid robot on children with autism. *Designing a more inclusive world*, pages 225–236, 2004.

[34] Michael Rubenstein, Adrian Cabrera, Justin Werfel, Golnaz Habibi, James McLurkin, and Radhika Nagpal. Collective transport of complex objects by simple robots: theory and experiments. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 47–54. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[35] Dorsa Sadigh. *Safe and Interactive Autonomy: Control, Learning, and Verification*. PhD thesis, University of California, Berkeley, 2017.

[36] Dorsa Sadigh, S. Shankar Sastry, Sanjit A. Seshia, and Anca Dragan. Information gathering actions over human internal state. In *Proceedings of the IEEE, /RSJ, International Conference on Intelligent Robots and Systems (IROS)*, pages 66–73. IEEE, October 2016. doi: 10.1109/IROS.2016.7759036.

[37] Dorsa Sadigh, S. Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *Proceedings of Robotics: Science and Systems (RSS)*, June 2016. doi: 10.15607/RSS.2016.XII.029.

[38] Dorsa Sadigh, Nick Landolfi, Shankar S Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state. *Autonomous Robots*, 42(7): 1405–1426, 2018.

[39] Michaéla C Schippers, Deanne N Den Hartog, Paul L Koopman, and Daan van Knippenberg. The role of transformational leadership in enhancing team reflexivity. *Human Relations*, 61(11):1593–1616, 2008.

[40] Peter Stone, Gal A Kaminka, Sarit Kraus, Jeffrey S Rosenschein, and Noa Agmon. Teaching and leading an ad hoc teammate: Collaboration without pre-coordination. *Artificial Intelligence*, 203:35–65, 2013.

[41] Fay Sudweeks and Simeon J Simoff. Leading conversations: Communication behaviours of emergent leaders in virtual teams. In *proceedings of the 38th annual Hawaii international conference on system sciences*, pages 108a–108a. IEEE, 2005.

[42] Daniel Szafir, Bilge Mutlu, and Terrence Fong. Communicating directionality in flying robots. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 19–26. IEEE, 2015.

[43] Zijian Wang and Mac Schwager. Force-amplifying n-robot transport system (force-ants) for cooperative planar manipulation without communication. *The International Journal of Robotics Research*, 35(13):1564–1586, 2016.

[44] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

VII. APPENDIX

A. Potential Field for Simulated Human Planning

In our implementation, the attractive potential field of attraction i, denoted as $U^i_{att}(q)$, is constructed as the square of the Euclidean distance $\rho_i(q)$ between agent at location q and attraction i at location q_i. In this way, the attraction increases as the distance to goal becomes larger. ϵ is the hyper-parameter for controlling how strong the attraction is and has consistent value for all attractions.

$$\rho_i(q) = \|q - q_i\|$$
$$U^i_{att}(q) = \tfrac{1}{2}\epsilon\rho_i(q)^2$$
$$-\nabla U^i_{att}(q) = -\epsilon\rho_i(q)(\nabla\rho_i(q))$$

The repulsive potential field $U^j_{rep}(q)$ is used for obstacle avoidance. It usually has a limited effective radius since we don't want the obstacle to affect agents' planning if they are far way from each other. Our choice for $U^j_{rep}(q)$ has a limited range γ_0, where the value is zero outside the range. Within distance γ_0, the repulsive potential field increases as the agent approaches the obstacle. Here, we denote the minimum distance from the agent to the obstacle j as $\gamma_j(q)$. Coefficient η and range γ_0 are the hyper-parameters for controlling how conservative we want our collision avoidance to be and is consistent for all obstacles. Larger values of η and γ_0 mean that we are more conservative with collision avoidance and want the agent to keep a larger distance to obstacles.

$$\gamma_j(q) = \min_{q' \in obs_j} \|q - q'\|$$
$$U^j_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{\gamma_j(q)} - \frac{1}{\gamma_0}) & \gamma_j(q) < \gamma_0 \\ 0 & \gamma_j(q) > \gamma_0 \end{cases}$$
$$\nabla U^j_{rep}(q) = \begin{cases} \eta(\frac{1}{\gamma_j(q)} - \frac{1}{\gamma_0})(\frac{1}{\gamma_j(q)^2})\nabla\gamma(q) & \gamma(q)_j < \gamma_0 \\ 0 & \gamma_j(q) > \gamma_0 \end{cases}$$