Influencing Leading and Following in Human-Robot Teams

Mengxi Li* · **Minae Kwon*** · **Dorsa Sadigh**

Abstract Roles such as leading and following can emerge naturally in human groups. However, in human-robot teams, such roles are often predefined due to the difficulty of scalably learning and adapting to them. In this work, we enable a robot to efficiently learn how group dynamics emerge and evolve in human teams and we leverage this understanding to plan for influencing actions for autonomous robots that guide the team toward achieving a common goal. We first develop an effective and concise representation of group dynamics, such as leading and following, by enforcing a graph structure while learning the weights of the edges corresponding to one-to-one relationships between the agents. We then develop an optimization-based robot policy that leverages this graph representation to attain an objective by influencing a human team. We apply our framework to two types of group dynamics, leading-following and predator-prey, and show that our structured representation is scalable with different human team sizes and also generalizable across different tasks. We also show that robots that utilize this representation are able to successfully influence a group to achieve various goals compared to robots that do not have access to these graph representations. [1]

Keywords human-robot teaming · human modeling · multiagent systems

1 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. 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 autonomous and human-driven cars.

In these collaborative teamwork scenarios, an important challenge for robots is to understand and interact with human agents seamlessly and even further influence a human team to achieve a desired goal. For instance, imagine a mixed human-robot search and rescue mission with no direct communication capabilities similar to Fig. 1. When a quadcopter senses valuable information from the environment how should the quadcopter direct the rest of its human teammates toward the desired goal?

One common solution is to assign leading and following roles to the team a priori before starting the search and rescue mission. Many current human-robot interactions determine leader-follower roles beforehand [29, 44, 55, 80, 84, 33, 72]. This include tasks that require learning from demonstrations or preferences, where the human is considered as the leader and the robot is the follower [19, 3, 1, 87, 23, 62, 13, 68], or assistive tasks where the robot teaches or assists human users [70, 41, 57, 39, 51]. However, assigning leadership roles a priori is not always feasible in dynamically changing environments or long-term interactions.

There has also been significant prior work on how we can construct intelligent robot policies that induce desired behaviors from people [76, 35, 63, 64, 12, 85, 56]. However, all of these works optimize for robot policies that influence only a single human in one-on-one interactions. These works are able to successfully produce

M. Li, M. Kwon, and D. Sadigh
Computer Science Department, Stanford University, E-mail: {mengxili,mnkwon, dorsa}@stanford.edu

[1] Parts of this work has been published at Robotics: Science and Systems (RSS) [47].

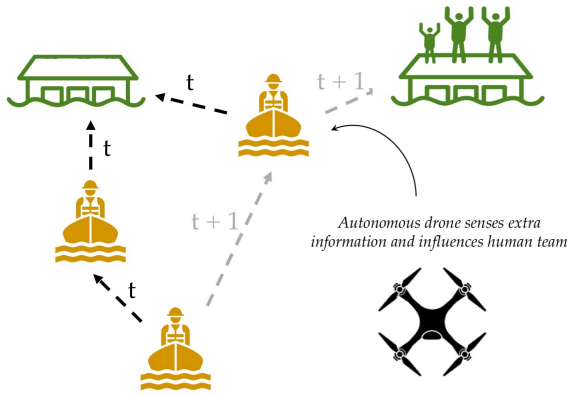


Fig. 1: A search and rescue example, where a team of humans intend to rescue people from two islands shown in green. The quadcopter collects more information and determines that the team should head towards the island on the right. It guides the human team toward the island on the right using a graph representation that models the human team. We estimate leading and following relationships in human teams (denoted by the arrows), and use this to create influential robot policies. The black arrows represent intended human leading and following behaviors whereas the grey arrows represent updated leading and following behaviors after the influencing robot action.

influencing behaviors by keeping an estimate of the human's state and optimize for actions based on the estimation, which is often computationally intractable 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. Similar to individuals, teams exhibit behavioral patterns and structures that robots can use to create intelligent influencing policies. One particular feature of human teams we will focus on in this work is *leading and following relationships.*

> *Our key insight is that there exists an underlying graphical structure encoding the larger and more complex interactions between humans in team settings.*

In this paper, we develop a scalable approach to extract meaningful latent structures in teams of humans that represent their leading and following behaviors. We extract an underlying graph, *leader-follower graph (LFG)*, to represent the *global* pattern of leader-follower dynamics using information from *local*, pairwise leader-follower interactions that we learn using supervised learning techniques. This structure provides a concise and informative representation of the current state of the team and can be used in planning. We then develop novel strategies for robots who join the human team to efficiently estimate the leader-follower graph and further influence this structure to more efficiently achieve the team's goals. For instance, as shown in Fig. 1, there is an underlying team structure between the humans who are collaboratively navigating towards the left goal. However, a robot capable of estimating this underlying structure through the leader-follower graph can follow strategies that collectively influence the team to instead navigate the team towards the right goal, which could lead to a more desirable outcome.

We demonstrate the generalizability of our approach by applying our framework to a second type of group dynamics: predator-prey relationships. We show that we are able to successfully model predator-prey relationships using leader-follower graphs (LFGs). We also demonstrate that a robot using this LFG model is able to influence predator-prey dynamics.

Our contributions in this paper are as follows:

- Formalizing and learning a graphical structure that captures complex relationships between members in human teams.
- Developing optimization-based robot strategies that leverage the graph representation to influence the team towards a more efficient objective.
- Providing simulation experiments in a pursuit-evasion game demonstrating the robot's influencing strategies to reverse a leader-follower relationship, distract a team, and lead a team towards an optimal goal based on its learned leader-follower graph.
- Generalizing our framework to a predator-prey domain and showing that our framework can still successfully model group dynamics, scalably deal with different group sizes, and can be used to design influencing policies.

In the rest of this paper, we first discuss relevant work on modeling teams, influencing teams, and ad hoc teamwork in Section 2. We then describe our formalism and algorithm for learning graphical representations of human teams in the leader-follower domain (Sections 3-5) followed by the predator-prey domain(Sections 6-8). Finally, we describe our experiments in the leader-follower domain (Section 9) and the predator-prey domain (Section 10) followed by a discussion of limitations and future works.

2 Related Work

2.1 Modeling Teams

Finding computationally efficient ways to model human teams is an important part of this work. These models can be used to design intelligent policies that allow an agent to influence or coordinate with the team. We review ways in which prior works have modeled groups of agents.

Flocks and Swarms. Many works model flocks and swarms inspired by animal flocking behavior [32, 22, 71, 81]. These models describe how groups reach consensus in orientation when navigating a space. They generally assume that all agents are homogenous and that they follow the same, relatively simple, update rule. Important components of this update rule include aligning orientation with their neighbors, positional attraction and repulsion towards neighbors, and some noise [32]. For example, Cristiani and Piccoli are able to replicate many self-organized patterns found in nature by modeling long-range cohesion, short-range repulsion, and the agents' visual fields [22]. Rosenthal et al. show that all agents are not equally susceptible to being influenced. They show that individuals with relatively few strongly connected neighbors are both more socially influential and susceptible to being influenced [71]. While these models are computationally efficient, they are too simplistic to be able to capture social dynamics that occur in human teams.

Attention and Graph Neural Networks. Recently, graph neural networks that use attention have become popular for modeling agent interactions [36, 52, 37, 40]. Attention is generally used to learn edge weights between agents. Vertex Attention Interaction Network (VAIN) uses attention to capture local structure by allowing the network to determine which agents will share information [36]. Li et al. uses self-attention to find structure in a coordination graph and then uses graph neural networks to integrate information among all agents [52]. Jiang et al. uses multi-head dot product attention to extract relations among neighboring agents in order to increase agents' receptive fields. Latent features are then extracted from these enlarged receptive fields to learn cooperative policies [40]. Compared to our approach, attention-based methods generally have more parameters and thus require more data to train. However, using attention-based methods to model human teams could be promising future work.

Modeling Humans. While there are many works that model multiagent systems, the extent to which these models can generalize to groups of humans remains underexplored. Many works in cognitive science, psychology, and behavioral economics have created predictive models of humans by modeling their biases and suboptimalities. For instance, Ordonez and Benson III investigated how humans make decisions under time constraints [66]. Simon developed the concept of bounded rationality to reflect limited humans' limited cognitive resources [77]. Tversky and Kahneman developed Cumulative Prospect Theory to capture human-decision making under risk and uncertainty [83]. In robotics, being able to successfully predict human behavior has shown to improve performance on tasks such as assistive robotics [51, 57, 39, 25], autonomous driving [74, 75, 6], collaborative games [61], and motion planning [88, 59]. The noisy rational choice model has been an extremely popular choice due to its simplicity [14, 13, 11, 27, 7]. Other models include the adoption of Cumulative Prospect Theory for human-robot interaction [48], models of human driving [34, 53], as well as learning-based models [60, 67].

In addition to explicitly modeling human behavior, robots have also been able to infer human preferences through interactions using partially observable Markov decision processes (POMDPs) which allow reasoning over uncertainty on the humans' internal state or intent [17, 24, 50, 58, 38, 75]. Human's intent inference has also been achieved through human-robot cross-training [62] as well as various other approximations to POMDP solutions such as augmented MDPs, belief space planning, approximating reachable belief space, and decentralization [2, 45, 46, 65, 69, 73]. However, these methods usually focus on modeling a single human agent and do not capture social dynamics that occur among humans.

2.2 Influencing Teams

Given a model of a team, an important next question is how a more informed agent can use this model to coordinate with or influence the team.

Flocks and Swarms. Literature on influencing flocks and swarms looks at how informed agents can guide the group towards a preferred direction. This is similar to some of our evaluation tasks where the robot agent attempts to guide the human team towards a particular goal. The homogeneity and simple nature of agents in flocks and swarms allow for leader agents to implicitly influence the group. More specifically, implicit leadership algorithms allow a group of agents to reach consensus where each agent can observe their neighbors' states within a particular radius. As agents attempt to align their orientation with their neighbors', this empowers informed agents to lead [86, 31]. Prior work has

also examined properties that make a swarm more susceptible to influence. Couzin et al. show that in groups of animals, only a small proportion of informed agents are required, and the larger the group, a smaller the proportion of informed individuals are needed [21]. Celikkanat et al. study the extent to which informed individuals can lead a flock by varying three factors: (1) the weight of the direction of preference (2) the ratio of informed individuals and (3) the size of the flock. They find that a flock is easier to control when moderate weight is put on the direction of preference (2) larger flock sizes and (3) more agents attempt to align their states with neighboring agents' states [18]. It is difficult to apply these findings to human teams due to the simplicity of flock and swarm models.

Human-Swarm Interaction. There has also been considerable work on how humans can influence flocks and swarms. Tiwari et al. consider the problem of leader placement when steering a large robot swarm [82]. Robots can either be controlled by a human or behave according to a swarm model. The authors consider which robots are positionally best equipped to influence the swarm (front, middle, or periphery). Kerman et al. and Brown et al. also consider how humans can influence swarms by controlling a subset of them [43, 16]. They show that humans are able to lead the swarm to switch from torus to flock formations and vice versa. Our work tackles the reverse problem where a robot agent must influence a team of humans.

2.3 Ad Hoc Teaming

An autonomous ad hoc agent must both model and influence a team that it has never seen before [79]. The ad hoc setting is similar to ours in that we expect our robot agent to influence a human team that it has never worked with before. Ad hoc teaming has been studied in the multi-armed bandit setting where a teacher needs to trade off between teaching a new learning agent and exploitation [78, 10]. Role assignment in ad hoc teams have also been studied [15, 30]. Typically, an ad hoc agent needs to select a role such that it maximizes the team's utility. For instance, in Bowling and McCracken's work, teammates assign a role to the agent and the agent's job is to infer its role by simulating plays and selecting the one that is most similar to current teammate behavior [15]. Liemhetcharat models how well agents work together in ad hoc teams using a graph; nodes represent agents, their value represent the agent's capabilities, and agent synergy is determined by their capabilities and how far apart they are located from other agents in the graph [54]. Liemhetcharat describes how to learn this graph based on observations of team performance and then use this model to plan for creating effective ad hoc teams. Barrett et al. introduce model-based and model-free algorithms that allows ad hoc agents to collaborate with a variety of different teammates [9]. The algorithms either learn models about prior teammates or policies on how to collaborate with prior teammates, and uses this knowledge to interact with current teammates. Albrecht assumes that agents can be characterized into a set of policies drawn from some unknown distribution [4]. The author uses a Bayesian approach where agents update their posterior beliefs about types of other agents which can then be used for planning. While many of these ad hoc teaming works focus on modeling different types of potential teammates, in this work, we focus on modeling a specific type of *latent group dynamics* — leading and following graphs — in order to enable a robot to interact with an unknown team.

3 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*. The objective of the pursuers is to collaboratively capture the evaders (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, $A_i = \{$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, and a goal will be captured when all pursuers collide with it as shown in Fig. 2 (b).

Leaders and Followers. We define a set of goals $g \in G$, which 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, the goals informally correspond to the evaders that need to be captured by all the pursuers, i.e., all the agents (pursuers) need to reach a state corresponding to the goals (evaders) being captured. A goal in 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 the goals. To put this in the context of leading and

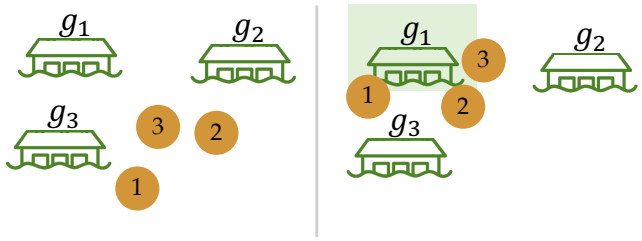


Fig. 2: Pursuit-evasion game. (Left) we demonstrate a pursuit-evasion game with three goals (green circles), and three pursuers (orange circles). The pursuers must jointly agree on moving toward a target. (Right) The three pursuers move to g_1 to capture it.

following, when agents capture a goal, *the goal can be thought of as being followed.*

Each agent $i \in I$ *follows* a goal or another agent, which we refer to 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. 3 (a), where agent 2 follows goal g_1 ($l_2 = g_1$) and agent 3 follows agent 2 ($l_3 = 2$).

Leader-Follower Graph. The set of leaders and followers form a directed *leader-follower graph* as shown in Fig. 3 (a). 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. 3 (a), $w_{3,2}$ 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, and we do not expect to observe cycling leading-following behaviors (Fig. 3 b). Other patterns that are likely include: chain patterns (Fig. 3 c) or patterns with multiple teams where multiple agents directly follow goals (Fig. 3 d). We describe how to construct a leader-follower graph that is scalable with the number of agents and avoids the undesirable patterns in Sec. 4.

Partial Observability. The leader of each agent, l_i, is a latent variable. We assume that agents cannot directly observe the 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 have full information on the observations of themselves and all other agents. (e.g. positions and velocities of agents).

4 Construction of a Leader-Follower Graph

In this section, we focus on constructing the leader-follower graph that emerges in collaborative teams. We will first focus on learning pairwise relationships between agents using a supervised learning approach. We then generalize our dyadic scoring to multi-player settings using graph theoretic algorithms. This combination of data-driven and graph-theoretic approaches allows the leader-follower graph to efficiently scale up with the number of agents. Our aim is to leverage this leader-follower graph to enable robot teammates to produce helpful leading behaviors.

4.1 Pairwise Leadership Scores

We first 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 develop 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 LSTM network with a softmax layer to predict each agent's most likely leader.

Data Collection. We collect labeled human data by asking participants to play a pursuit evasion game. 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 configurations of leaders and followers in a two-player setting (the configurations are shown in Fig. 7). We collected a total of 186 games.

Since human data is often noisy and difficult to collect in large amounts; we further augmented our dataset with synthetic data, where we had simulated humans play the game. We simulated humans based on a potential field path planner [8]. Agents at location q plan

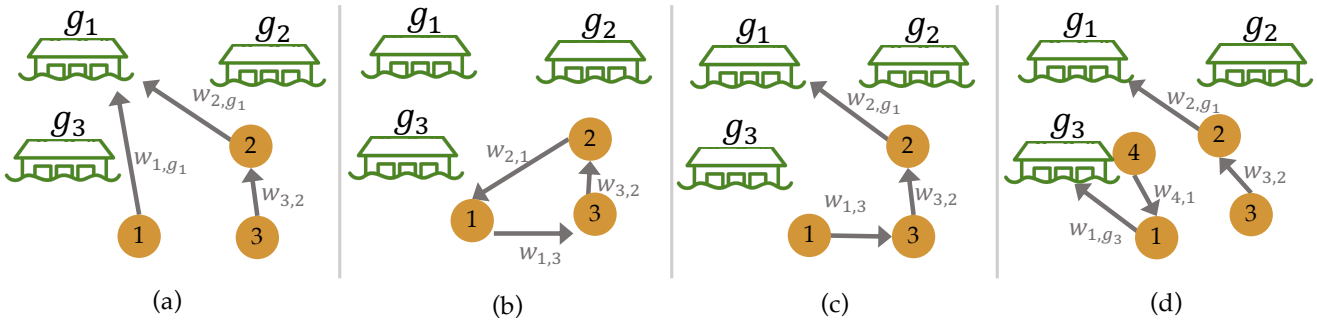


Fig. 3: (a) Leader-follower graph. Green islands are the goals that need to be captured. Orange circles are the pursuers. (b) Cyclic leader-follower graph. We design policies that avoid such cyclic behaviors. (c) Chain behavior in the leader-follower graph. (d) Multiple teams.

their path under the influence of an artificial potential field $U(q)$, which is constructed to reflect the environment. Agents moved toward their leaders by following an attractive potential field. Other agents and goals that are not their leaders are treated as obstacles that emit a repulsive potential field. In our game setting, the position of agent's assigned leader l_i is given an attractive potential field. The rest of the goals and agents are expressed as repulsive potentials.

Potential Field for Simulated Human Planning. We denote the set of attractions as $\mathcal{A}$, and the set of repulsive obstacles as $\mathcal{R}$. The overall potential field is a weighted sum of potential fields from all attractive and repulsive obstacles. θ_i is the weight for attractive potential field from $i \in \mathcal{A}$, and θ_j is the weight for repulsive potential field from $j \in \mathcal{R}$.

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

The optimal action a that an agent would take lies in the direction of the potential field gradient.

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

In our implementation, the attractive potential field increases as the distance to goal becomes larger to help the agent reach the goal. On the other hand, 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. Specifically, the attractive potential field of attraction i, denoted as $U^i_{\text{att}}(q)$, 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.

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

The repulsive potential field $U^j_{\text{rep}}(q)$ is used for obstacle avoidance. It usually has a limited effective radius since we do not want the obstacle to affect agents' planning if they are far way from each other. Our choice for $U^j_{\text{rep}}(q)$ 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. Thus, to compute the repulsive potential field to obstacle j at location q, we first identify the minimum distance $\gamma_j(q)$ between q and the obstacle j as in Eq.(2). 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.

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

In our experiments, we find that our simulations are good approximations of human behavior. The simple nature of the task given to humans (i.e., move directly toward your assigned leader l_i) is easily replicated in simulation.

Training with a Scalable Network Architecture. Our network architecture consists of two LSTM submodules, one to predict player-player leader-follower

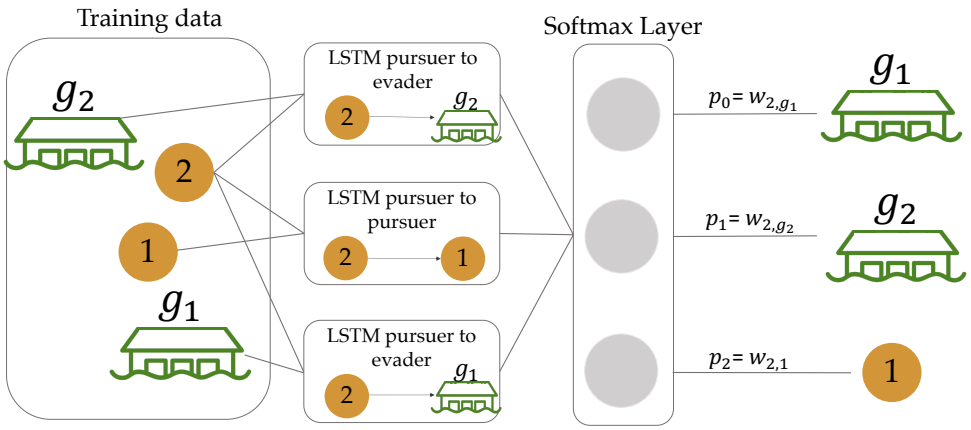


Fig. 4: Scalable neural network architecture. This example predicts the probability of another agent j being agent 2's leader, $w_{2,j}$. There are three LSTM submodules used because there are two possible evaders and one possible agent that could be agent 2's leader. This architecture demonstrates how one can select P-P and P-E modules and discover the leader-follower relationships in a more scalable and compositional manner.

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.

Evaluating Pairwise Scores. Our network trained on two-player simulated data successfully captured the pairwise leading-following relationship (training accuracy: 80%, validation accuracy: 83%). We also experimented with training with three-player simulated data as well as a combination of two-player simulated and human data (two-player mixed data) resulting in (training accuracy: 97%, validation accuracy: 75%).

Validation results are shown in Fig. 5. Our model trained with mixed two-player data was first trained on simulated data and then trained on human data. For this reason, we have represented the mixed-data model as a horizontal line in Fig. 5 demonstrating the final validation accuracy.

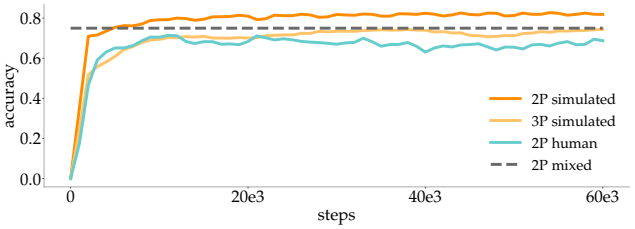


Fig. 5: Validation accuracy when calculating pairwise leadership scores trained on simulated, human, and mixed data (simulated & human), described in Sec. 4.1

4.2 Maximum Likelihood Leader-Follower Graph

To build a leader-follower graph in settings with more than two players, we compute pairwise weights $w_{i,j}$ of leader-follower relationships between all possible pairs of leaders i and followers j. The pairwise weights (leadership scores) can be computed based on the supervised learning approach described above, indicating the probability of one agent or goal being another agents' leader. After computing $w_{i,j}$ for all combinations of leaders and followers, we can 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 $w_{i,j}$. In addition, we add a special root node, where all the goals $g \in G$ have an outgoing edge to the root node. This produces a fully connected graph with each edge corresponding to the probability of one agent leading another, as shown in Fig. 6 (a).

Our model builds the graph based on the the pairwise scores, and thus can generalize to groups with different sizes. The computation increases quadratically with the size of the graph along with the number of pairs.

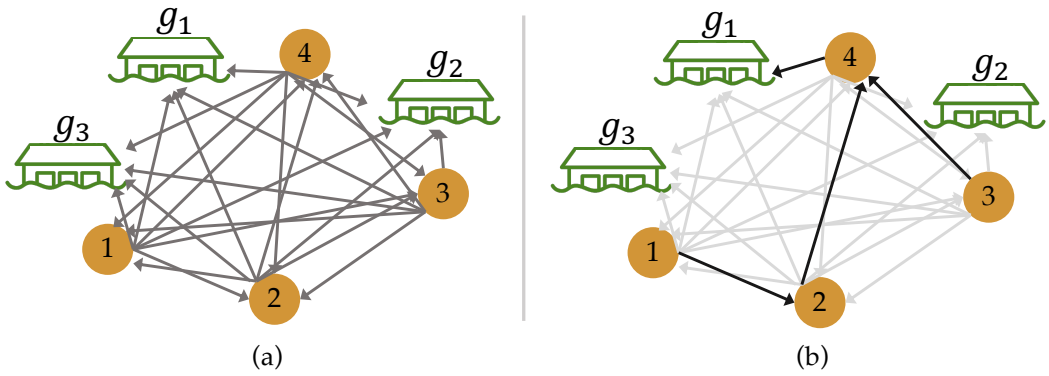


Fig. 6: (a) Graph $\mathcal{G}$. The directed edges represent pairwise likelihoods that the tail node is the head node's leader. (b) Maximum-likelihood leader-follower graph, $\mathcal{G}^*$. For each node, we select the outgoing edge that has the highest weight as shown by the bold edges.

To create a more useful graph, 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 as in Fig. 6 (b). 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 [26, 20], which finds a maximum weight arborescence [42] 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 since, compared to our approach, a maximum weight arborescence is more restrictive; it requires the resulting graph to be a tree.

Evaluating the Leader-Follower Graph. We evaluate how accurate our leader-follower graph with three or more agents is 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 1). We evaluated our leader-follower graph on simulated three, four, and five-player games, as well as two and three-player human games. 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 random guessing by a large margin. The random policy 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 then take the average of all success probabilities for all leader-follower graph configurations to compute the overall accuracy. As an example, for two-player games, there are in total three possible configurations as shown in Fig. 7. We compute the overall accuracy of the game by averaging $\frac{1}{2}$, $\frac{1}{2}$ and $\frac{1}{3}$, giving 0.44 (line 4, Table 1).

Table 1: Generalization accuracy (Acc) of leader-follower graph (LFG) trained and tested with various data sources.

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

In all experiments shown in Table 1, our trained model clearly outperforms the random policy. Most notably, the models trained on simulated data scale naturally to settings with large numbers of players as well as human data. We use the model trained on three-player simulated data for our experiments in Section 5.

5 Planning based on Inference over Leader-Follower Graphs

With a representation for latent leadership structures in human teams, we use a leader-follower graph $\mathcal{G}^*$ to positively influence human teams, i.e., move the team towards a more desirable outcome. We describe how a

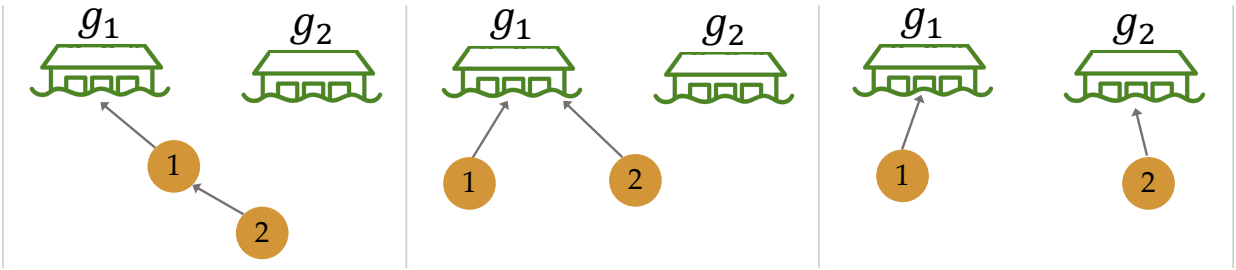


Fig. 7: All possible leader-follower graph configurations for two-player settings.

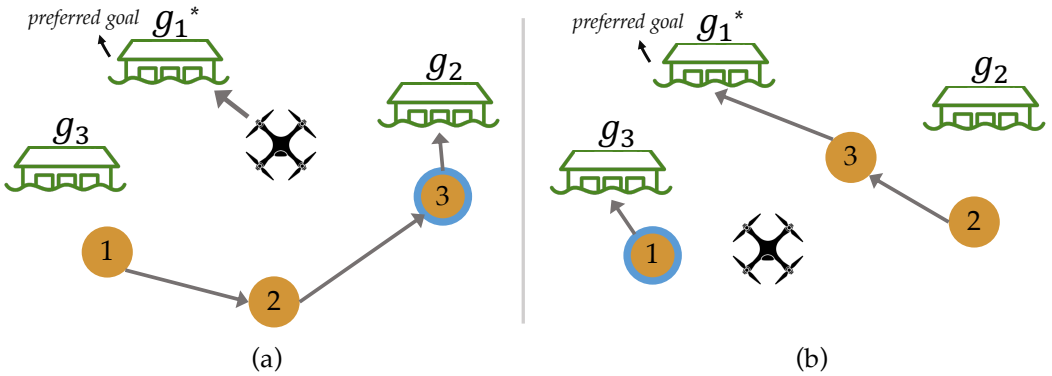


Fig. 8: (a) In this graph, the most influential leader is agent 2. (b) The most influential leader trivially becomes agent 1 since agents 2 and 3 are already targeting the optimal goal g_1^*.

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.

5.1 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 allow 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). A robot can then plan for a desired outcome by influencing or following these key goals and agents. 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. Being able to quickly infer the goal of multiple agents enables the robot to plan efficiently.

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 allows the robot to strategically influence a single teammate that also indirectly influences the other teammates that are following the most influential leader. For example, in Fig. 8 (a) and (b), 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. 8 (b).

5.2 Optimization Based on Leader-Follower Graph

The leader-follower graph allows the robot to single out key players and goals to follow or influence. A robot can then use this information to directly optimize for actions that help it achieve a desired outcome: Out-

comes such as following the crowd or influencing the crowd's decision through utilizing the leader-follower graph. For instance, the probability of the robot becoming an agent i's leader can be expressed as $w_{i,r}$. The probability of the robot following a goal g is $w_{r,g}$.

To select actions $a \in A$ that maximize an objective involving weights $w_{i,j}$ in the leader-follower graph, 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 that the robot is optimizing for (e.g., the robot becoming agent i's leader). We then optimize over the robot's actions to find the action a^*_t that maximizes a reward/outcome r that can be expressed in terms of $w^{t+k}_{i,j}$'s and $w^{t+k}_{i,g}$'s.

$$a^*_t = \underset{a_t \in A}{\text{argmax}}\ r\left(\{w^{t+k}_{i,j}(a_t)\}_{i,j\in I}, \{w^{t+k}_{i,g}(a_t)\}_{i\in I, g\in G}\right) \tag{3}$$

We describe three specific tasks that we will plan for using the optimization described in Eqn. (3).

Reversing a Leader-Follower Relationship. A robot can directly influence team dynamics by changing leader-follower relationships. Given a directed edge between agents i and j, the robot can use the optimization outlined in Eqn. (3) 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:

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

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. 3 (d)) by redirecting the edges of one sub-group to follow another. On the other hand, the robot can also redirect edges such that the team is dispersed, or reverse edges such that the edges form a cycle as shown in Fig. 3 (b).

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:

$$a^*_t = \underset{a_t \in A}{\text{argmax}}\quad w^{t+k}_{i^*r}(a_t) - w^{t+k}_{i^*g}(a_t),\ i^* \in \mathcal{I} \tag{4}$$

Distracting a team from reaching a collective goal can be useful in cases where the team is an adversary. For instance, a team of military drones masquerading as enemy drones may want to prevent the enemy team from 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^*:

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

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.

6 Modeling Predator-Prey Relationships

We test the generalizability of our framework by modeling a different type of group dynamics: predator-prey relationships. Each agent has either a prey that they are trying to capture, predators they are eluding, or both. Predator-prey relationships are different from leader-follower relationships in that agents are tasked with eluding their predator. In some cases, agents must simultaneously capture their prey while eluding their predator, giving way to more complex dynamics than leading and following. In human groups, predator-prey relationships can be found in games such as capture-the-flag. In capture-the flag, two teams guard regions that contain each team's flag. The goal of a team is to steal the other team's flag while protecting their own. Predator-prey relationships emerge when team members attempt to tag out members of the opposing team. An example is shown in Fig. 9 (a), where members of the blue team (agents 2 and 4) help their teammate (agent 1) escape from the opposing team.

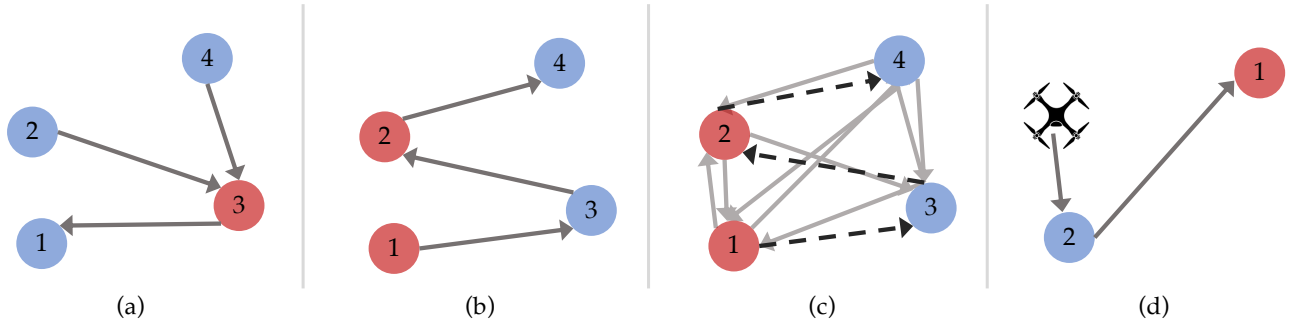


Fig. 9: (a) Example of a predator-prey dynamic in capture the flag. Red and blue circles represent agents on different teams. (b) Another example of a predator-prey dynamic between members of red and blue teams. (c) Predator-prey graph where the most likely edges are bolded. (d) The robot joins the game as agent 2's predator.

Predator-Prey Game. We modify the pursuit-evasion game setup described in Sec. 3. In the modified version, there are no stationary evaders (goals). Instead, each agent in the set of agents I acts as either a predator, prey, or both. Predator agents are assigned a prey and are required to capture it by colliding with them. Likewise, prey agents have assigned predators and their goal is to avoid being captured. The action space of each agent $i \in I$ is identical as in Sec. 3, $A_i =$ {move up, move down, move left, move right, stay still}.

Predator-Prey Graph. Using the set of predators and preys, we can form a directed predator-prey graph. Each node represents an agent $i \in I$. The directed edges represent predator-prey relationships where there is an outgoing edge from a predator to its prey. The weights on the edges represent the probability that the tail node is the head node's predator. Similar to leader-follower graphs, there can be many configurations of predator-prey graphs; two examples are shown in Figs. 9 (a) and (b). In this work, we experiment with various configurations of the predator-prey to validate that our framework can effectively capture these relationships between the agents in this predator-prey domain.

7 Construction of a Predator-Prey Graph

In this section, we describe how we learn these graphs. Similar to our leader-follower graphs, we use a supervised learning approach where we collect pairwise predator-prey data to train a predictive model. Like the leader-follower graph, our aim is to use this model to scalably construct a predator-prey graph for multi-agent settings. Ultimately, we hope to use this graph to build robot algorithms that can understand and influence predator-prey dynamics.

7.1 Pairwise Prey Scores

Data Collection. We recruited dyads to play the predator-prey game. We assumed a chain-structured predator-prey relationship. Predator and prey roles were randomly assigned to each partner. Participants played the game in the web browser where predators tried to collide with their prey as many times as possible within the time limit. We collected a total of 1.5 hours of data where we collected trajectories and scores of each participant.

We also generated synthetic human data using the same potential field simulator as described in Sec. 4.1). At the beginning of each game, each agent was randomly assigned one prey or no prey. Only configurations that contain no loops are considered valid. By randomly assigning preys, we effectively covered all possible valid configurations. In our simulator, predators moved towards their prey by following an attractive potential field and prey moved away following a negative potential field. We simulated 1000 three-agent predator prey games. We chose three-agent games for data collection because to have agents that being both a predator and a prey, the minimal number of agents in the game is three.

Training the Model. To test the generalization of our framework in this new domain, we use the same LSTM submodules as in Sec. 4 to predict player-player predator-prey relationships. For each agent $i \in I$ in a game, we train our model to predict agent i's prey by feeding their and their partner's trajectory data into our submodules. We add an additional submodule where the agent i's trajectories are fed in twice to represent the event that the agent does not have prey. We use a softmax output layer with a cross-entropy loss function to compute a probability distribution over all agents of being agent i's prey. Before training, we pre-processed

the data by normalizing it, shifting it to have a zero-centered mean, and down-sampled it. When training our network, we used the same hyperparameters as described in Sec. 4.

Evaluating Pairwise Scores. We evaluated the accuracy of our model on held out test sets of simulated and human data. Our network trained on three-player data performed with a validation accuracy of 95.51% in simulation with randomized predator-prey graph structure and 96.24% on human data with a chain structure. Both results indicate that our framework can accurately capture the predator-prey pairwise relationship.

7.2 Constructing and Evaluating the Predator-Prey Graph

In settings with more than three players, we construct a predator-prey graph based on the pairwise scores. For each agent i, we compute pairwise weights between agent i and all of its possible preys $j \in I, j \neq i$. In this Predator-Prey domain, we also compute an additional weight for agent i having no prey. With all of these scores, we then construct the graph as described in Sec. 4.

Evaluating the Predator-Prey Graph. We tested the generalization accuracy of the predator-prey graph by constructing the graph at each time step and comparing it against the ground truth labels. The results for testing on real human data and simulated data are demonstrated in Fig. 10 respectively. We found that the model trained with three agent data can successfully generalize to settings with more players with only a minor decrease on the accuracy. Similar patterns to Table 1 can also be observed here where the accuracy drops with larger numbers of agents. This is because as the number of agents increases, the task becomes more challenging and it becomes more difficult for the model to distinguish which agent is the prey.

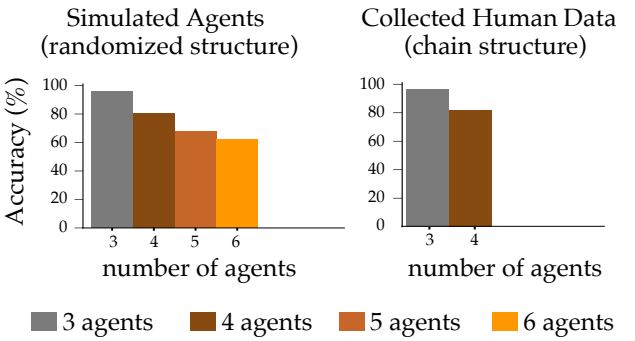


Fig. 10: Generalization accuracy of the predator-prey graph tested with simulated agents and human data. Both models are trained with three-agent data and we tested models in games that contain more agents.

8 Planning with the Predator-Prey Graphs

We now leverage the information from the predator-prey graph to plan for robot behaviors that can influence group dynamics. In this work, we focus on the task of becoming the only dominant predator among the chain-structured predator-prey group. Accomplishing this task requires two steps. First we need to identify which agent is the top predator, and then we want the robot to hunt for that identified top predator.

Inference Based on Predator-Prey Graph. One direct way to identify which agent is the top predator is to identify the agent that has no predator based on the estimated predator-prey graph. For example, as in Fig. 9 (b), agent 1 is the top predator. In this work, we experiment with a predator-prey chain, and therefore, there is only one top predator among the group.

Optimization Based on Predator-Prey Graph. After identifying the top predator, we can now again use the predator-prey graph for optimization. At each time step t, we infer the top predator X_t based on the current predator-prey graph. Then, similar to Sec. 5, we generate the predator-prey graph k time steps ahead $\mathcal{G}^{A_t}_{t+k}$, assuming the robot takes actions $A_t = (a_t, \dots, a_k)$ in the next k time steps. We then extract the weight w^{t+k}_{j,X_t} representing robot j being the predator of the identified top predator X_t from the graph $\mathcal{G}^{A}_{t+k}$. By maximizing this weight, we can compute the optimal robot actions:

$$A_t = (a_t, \dots, a_k) = \underset{A_t=(a_t,\dots,a_k)}{\operatorname{argmax}} \; w^{t+k}_{j,X_t} \tag{5}$$

We perform this optimization in a model predictive control fashion [28], where we find the optimal sequence of actions at time step t, and execute a_t. We then replan for a k time-step horizon at the next time step running the same optimization.

Other Tasks. In this work, we only demonstrate how to leverage the predator-prey graph for inference and optimization for the task of becoming the dominant predator in a chain-structured predator-prey group. However, like the leader-follower graph, we emphasize that the predator-prey graph is a general representation that can be combined with many other tasks as well, e.g., in more complex settings where the relationship is not limited to a chain structure. For example, the robot can help to protect another agent by identifying who its predators are and interfere with their actions.

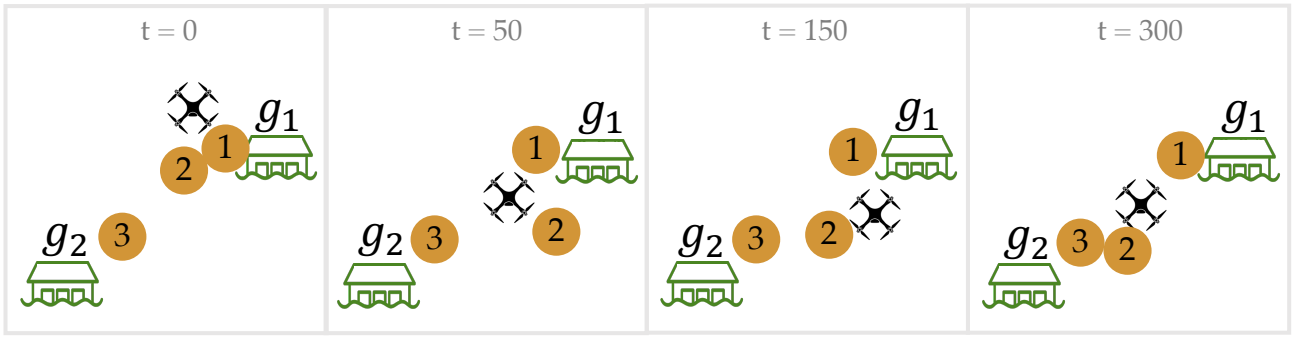


Fig. 11: Adversarial game snapshots for a 300 second horizon. The orange circles are human agents. (a) Agents 1 and 2 start very close to g_1. (b-c) The robot prevents agent 2 from converging on g_1. (d) The robot leads agent 2 to another goal, successfully extending the game time.

9 Experiments: Leading and Following

We first evaluate our framework in the leader and follower domain. Through our experiments, we demonstrate the efficacy of the leader-follower graph in representing the agents and further in enabling better planning and optimization for the robot actions.

We evaluate our LFG on three different tasks that involve influencing multiagent human teams. For each task, we compare task performance of robot policies that use the LFG against robot policies that do not have access to the LFG. Across all tasks, we find that robot policies that use the leader-follower graph perform better, showing that our graph can easily be 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. 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). We also specify weights associated with these attractions to be $\theta_g = 0.6$ and $\theta_c = 0.4$. In this way, a simulated human would trade off between following the crowd and moving toward a target goal.

In each iteration of the task, the initial position of agents and goals are randomized. For all of our experiments, game canvas is 500×500. At every time step, the human can move 1 unit in one of the four directions: up, down, left, right, or stay at its position. The robot's action space is the same but with larger move amount 5. We let the maximum game time limit be 1000.

Implementation Detail. We simulated 5000 games of each possible configuration, totaling 15000 games for the two-player setting (as shown in Fig. 7) and 35000 for the three-player setting. Each game stored the position of each agent and goal at all timesteps. Before training, we pre-processed the data by normalizing it, shifting it to have a zero-centered mean, and downsampled it. Each game was then fed into our network as a sequence. Based on our experiments, hyperparameters that worked well for our training were 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 [5] to stabilize gradient updates.

9.1 Reversing a Leader Follower Relationship

We evaluate a robot's ability to change an edge of a leader-follower graph. In this task, the end goal of the robot is not to affect the environment as some of the other tasks we describe below (e.g., influence humans toward a particular goal). Instead, this experiment 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 $\theta_g = 0.6$, $\theta_c = 0.4$, we created a robot policy that encouraged the human agent to follow the robot r instead. The robot optimized for the probability $w_{i,r}$ that it would become agent i's leader.

Metrics. We evaluated the performance of the robot based on the leadership scores, i.e., probabilities $w_{i,r}$, computed by the leader-follower graph.

Results. We show that the robot can influence a human agent to follow it. Fig. 12 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 method, which has an average performance of 26%, represented by the grey dashed line.

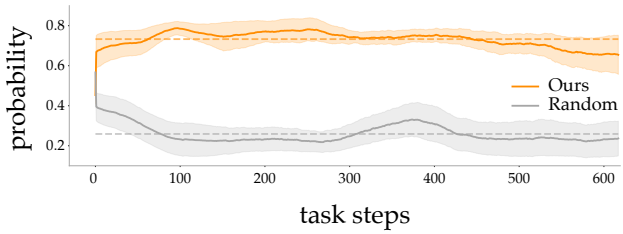


Fig. 12: Probabilities of a human agent following the robot over 10 tasks. The robot is successfully able to become the human agent's leader as the task progresses.

9.2 Adversarial Task: Distracting a Team

We now consider a task where the robot is an adversary that is trying to distract a team of humans from reaching a goal.

In this task, there are m goals and n players in the pursuit-evasion game. Among the n players, we have 1 robot agent and $n-1$ homogeneous human agents. $n-2$ human agents must collide with a goal at the same time to capture it, allowing 1 human to be absent. The game ends if all goals are captured or the game time exceeds the limit.

The adversarial robot's goal is to intentionally distract a team of human players so that they cannot converge to the same goal quickly and thus extending the game time. Note that simply blocking a single agent's way would not be a desirable solution, since we allow for an agent to be absent when capturing the goal.

Methods. We test our optimization methods based on the constructed leader-follower graph along with other baseline models.

We experimented with 3 baseline strategies without knowledge of LFG. In the *Random* strategy, the robot picks an action at each time step with uniform probability. *To One Pursuer* strategy is that the robot agent selects a random human agent and then goes towards it trying to block its way. The *To Farthest Goal* strategy selects the goal that the average distance to human players are largest and then goes to that goal in the hope that human agents would get influenced or may further change their goal by observing that some players are heading for another goal.

We also experimented with two optimization models based on the LFG. *LFG Closest Pursuer* involves the robot 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 described in Sec. 5 and then conducting the same optimization of maximizing the following probability, as shown in Eqn. (4).

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

Results. We conduct experiments with different game settings by varying n (number of players) and m (number of goals). For each specified game setting, we run the same 50 randomly initialized games for different robot strategy and compute the mean and standard deviation for game time over the 50 games. Across all the game settings we experimented with, our models based on LFG consistently outperforms methods without knowledge of LFG. The experimental results with varying number of players are summarized in Table 2. We also visualized the results of Table 2 in Fig. 13.

As shown in Fig. 13, average game time goes up as the number of players increases. This is because it is more challenging for more players to reach agreement on which goal to capture and thus takes longer time. The consistent advantageous performance suggests the effectiveness of LFG for inference and optimization in this scenario.

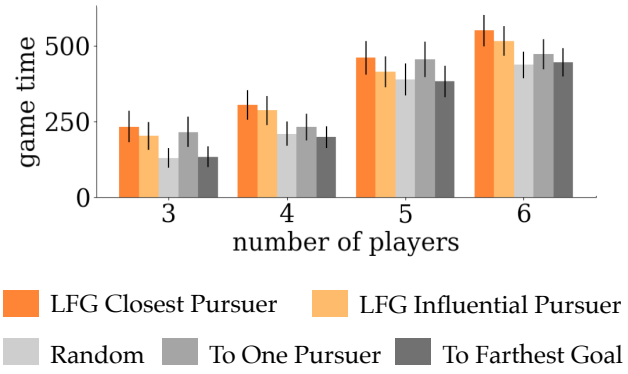


Fig. 13: Visualization of results in Table 2 For adversarial task, average game time over 50 games with 2 goals (as in Fig. 11) with different number of players across all baseline methods and our model.

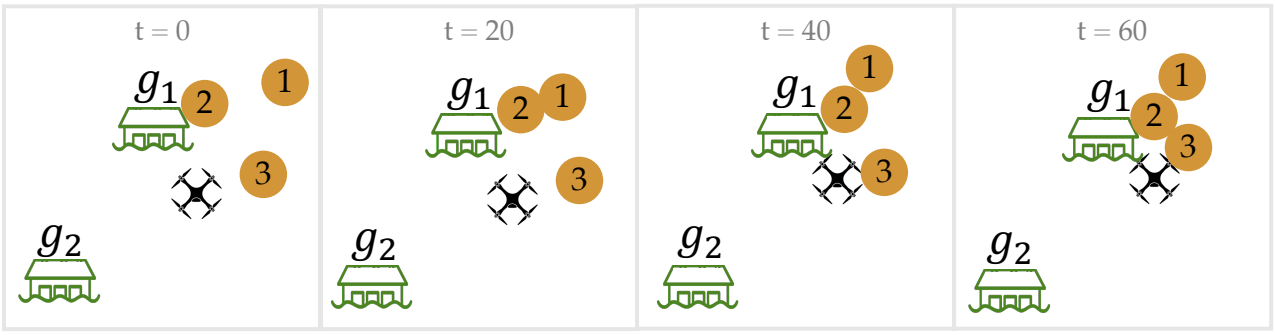
To demonstrate robot behavior in the adversarial game, we also took snapshots of one game as in Fig. 11. Player 1 and player 2 started very close to goal g_1 and thus it's very easy for them to capture it. The robot approached agent 2 and tried to block its way, leading it to another goal g_2. In this way, the robot successfully extended the game time.

Table 2: 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 3: 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



(a) (b) (c) (d)

Fig. 14: Collaborative game snapshots for a 60 second horizon. The orange circles are human agents. The robot moves towards agent 3 in order to help all the agents converge on g_1.

9.3 Cooperative Task: Leading a Team toward the Optimal Goal

Finally, we evaluate the robot in a cooperative setting where the robot tries to be helpful for human teams. The goal of the robot is to lead its teammates so that everyone can reach the target goal that gives the team the largest joint reward $g^* \in G$. g^* is not immediately observable to all teammates. We assume a setting where only the robot knows where g^* is (e.g. due to its better sensing capabilities as in Fig. 1).

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 successfully completed if the goal with the largest joint reward g^* is captured, and it is considered failed if any other suboptimal goal is captured or the game time exceeds the maximum limit.

Methods. Similar to the case in *Adversarial Task*, we explore two models where the robot chooses to influence its closest human agent or the most influential agent predicted by the LFG. Different from the *Adversarial Task*, here, the robot is optimizing the probability of the target agent following itself and the probability of them going to the desired goal.

We also experimented with three baseline methods. *Random* strategy is taking random actions. *To Target Goal* strategy is that the robot agent goes directly to the optimal goal g^* and then stays there trying to attract other human agents. *To Goal Farthest Player* strategy is that the robot goes to the player that is farthest away from g^* in the hope that it can influence the target back to g^*.

Metrics. We evaluated the performance of the robot strategy using the game success rate over 100 games.

Results. We experimented with varying number of goals and the results are summarized in Table 4. In this scenario, going directly to the desired goal is a very strong method since it already conveys the message to other players that the robot is going for a specific goal. This method is especially effective when the game is not complex, i.e., the number of goals is small. However, our model based on the LFG still demonstrates competitive performance compared to it. Specially when the number of goal increases, the advantage of LFG gradually becomes dominant. This indicates that, in complex scenarios, brute force methods that do not have knowledge of human team hidden structure do not suffice. High-level understanding of human teams are necessary for better human-robot teaming in complex systems. Another thing to note is that the difference between all of the methods becomes smaller as the number of goals increases. This is because the game difficulty increases for all methods, and thus whether a game would succeed depends more on the game's initial conditions. We

Table 4: 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

took snapshots of one game as in Fig. 14. In this game, the robot approaches other agents and the desired goal in the collaborative pattern, trying to help catch the goal g_1.

10 Experiments: Predator-Prey

We next evaluate our framework in the predator-prey domain. We investigate whether our robot can utilize the predator-prey graph to insert itself into the game as the top predator.

Task Setup. We evaluate our approach with both simulated and real human agents in the modified pursuit evasion environment. In all of our experiments, agents follow a chain structure as shown in Fig. 15. We reference each (simulated and real) human agent by their ids $1 \dots n$, where n is the number of human agents. Each agent is instructed to capture the agent above it and run away from the agent below it. Thus agent 1 will always be the top predator and agent n will be the bottom-most prey. The robot's task is to join the game and become the top predator, i.e., capture agent 1.

An example of a 4 player game is shown in Fig. 15. Agent 1 is the top predator that tries to capture agent 2. Agent 2 aims to capture agent 3 but also tries to avoid being captured by agent 1. Agent 3 is at the bottom of the predator-prey chain and simply tries to avoid being captured. The robot joins and tries to become the top predator by capturing agent 1. Importantly, we do not inform the robot that its goal is to capture agent 1. The robot has to figure this out by relying on the learned graph structure. Similarly, we also do not explicitly inform the other agents about the robot's goal, i.e. the other agents will treat the robot neither as its predator nor prey.

The initial position of all agents are randomized. Our experiments are conducted on a canvas of size 500 x 500. At each time step, each agent can move 3 units in one of the cardinal directions or choose to stay in place.

Methods. In order to become the dominant predator, the robot first identifies the top predator. It then optimizes for the probability that it becomes that agent's predator, as described in Sec. 8. We compare against two methods with our predator-prey graph: a random agent (*Random*) and an agent that optimizes for moving towards the center of the other three agents (*Center*). *Center* encourages the robot agent to stay closer to the group without knowing which agent is the top predator. By including the *Center* method, we hope to verify that the robot is actually following agent 1 and not following other agents.

Metrics. We evaluated the performance of the robot based on the average number of time steps that the robot agent is in collision with agent 2. Longer collision time indicates that the robot performs well by capturing the top predator among the other agents.

Results with Simulated Humans. We conduct experiments with different settings by varying the number of agents. For each specified game setting, we run the same 100 randomly initialized games and compute the mean and standard error for the time the robot agent is capturing the top predator. The experimental results are summarized in Fig. 16. Across all the settings, the predator-prey graph that our method used was trained only with three-player data. We leverage this graph to optimize robot behavior in various multi-agent games. We can see that our method captures prey for a longer amount of time compared to both *Random* and *Center* methods in the three-agent and four-agent settings. When the number of agents gets

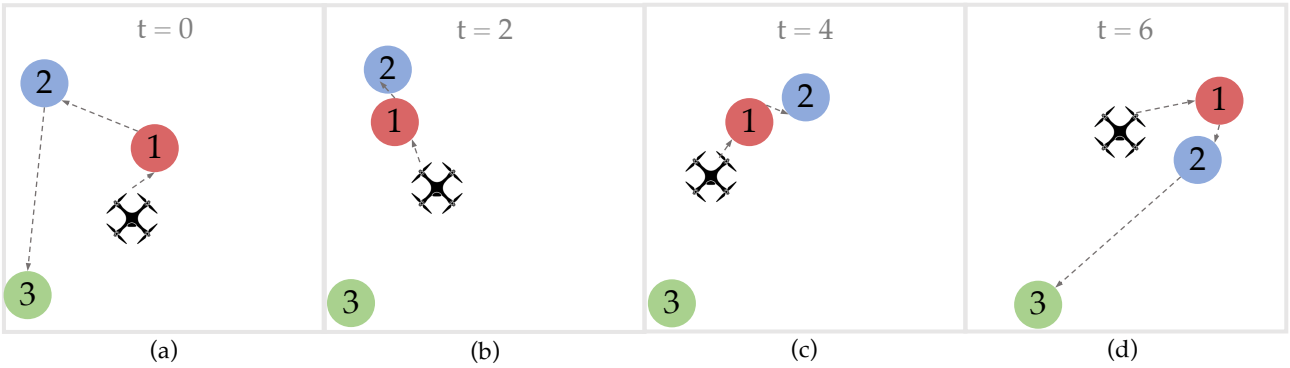


Fig. 15: Predator-prey game snapshots. The different colored circles represent agents in different teams. (a) Agents follow a chain structure in the predator-prey game. (b) As agent 2 attempts to capture agent 3, agent 1 intervenes and attempts to capture agent 2. (c) Agent 2 flees. (d) Agent 2 attempts to capture agent 3 again.

larger, e.g. when we have five agents, the performance degrades. This is because as the number of agents increases, the task becomes more challenging and correspondingly, the predator-prey generalization error also accumulates both in the inference and the optimization process.

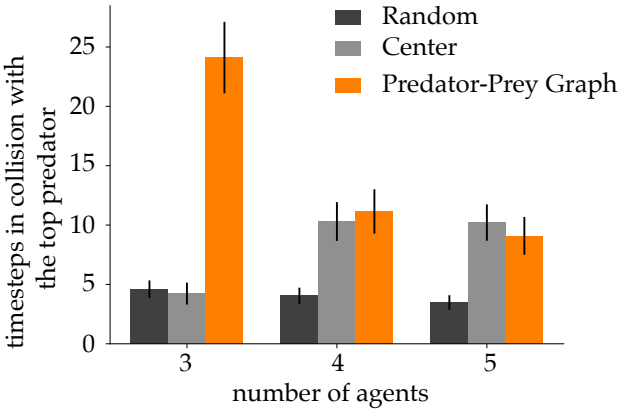


Fig. 16: Average time the robot agent capturing (in collision with) the top predator over 100 games with varying number of agents.

Results with Human Participants. To evaluate the effectiveness of our framework against real human users, We recruited human participants to play 3 player and 4 player versions of the game. We conducted 6 games with different groups of participants. Each group played the game three times with a robot following our algorithm as well as the *Center* and *Random* methods in a randomized order. Results are shown in Table 5.

In the 3 player setting, we report the mean time the robot and agent 1 were in collision with their prey as well as the ratio of the two means. We do not report agent 2's score because they had no assigned prey. The ratio highlights the effectiveness of the robot over the other human predator, and thus acts as a good metric for assessing the robot policy. Looking at the results, our method achieves a higher ratio than *Random* demonstrating the effectiveness of the robot policy when interacting with real humans. In 3 player settings, *Center* places the robot in between the two human agents, making it impossible for agent 1 to capture agent 2 without being captured by the robot. This makes the robot's job as a predator trivial. *Center* is therefore a special case of the 3 player setting. In practice, this often leads to stalemates where all agents remained far apart from each other, as shown by agent 1's 0 mean in Table 5. However, in two out of the six games, agent 1 came close enough to the robot, which explains the robot's large average and standard deviation for *Center*.

In the 4 player setting, we report the mean time the robot were in collision with their prey. With larger number of agents, our method demonstrates its advantage of capturing the group structure more and achieves highest performance. Compared to the special case in 3 player settings, the crowd's center was less correlated with its prey's position and thus *Center* demonstrates inferior performance compared to our method.

11 Discussion

Summary. We propose an approach for modeling group behavior in multi-agent human teams. We use a combination of data-driven and graph-theoretic techniques to learn a graph-based representation for leading-following and predator-prey dynamics. This graph representation encoding human team hidden structure is scal-

Table 5: Average number of time steps an agent is in collision with its prey over 6 games with 2 and 3 human participants.

Game	Agent	Ours	Center	Random
3 player game	Robot	229.67 ± 164.4	**657.44 ± 1365.04**	103.78 ± 100.94
	Human Agent 1	132.17 ± 265.9	0	438.89.83 ± 858.63
	Robot/Human Agent 1	**1.74**	N/A	0.24
4 player game	Robot	**230.83 ± 93.32**	136.64± 102.26	7.13 ± 44.59

able with the team size (number of agents) since we base the model on *local*, pairwise relationship prediction and combine them to create a *global* model. We demonstrate the effectiveness of our graph structure by testing 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.

There are several ways in which our framework can be applied to more complex real-world settings. First, we can extend our approach to partially observable settings. When human agent positions are partially observable (i.e., the robot can only access the positions of its nearest neighbors), our framework can still be applied locally. For instance, the robot can determine local leader-follower structures that can be updated as the robot moves around and gathers more information.

We include preliminary results on what running our framework on real robots might look like in Fig. 17. We use Zooids robots for our experiment, which is a collection of custom-designed wheeled micro tabletop robots and can be used for various tasks including swarm drawing, interactive swarm visualization [49]. Users control the movement of Zooids through a GUI on computers by dragging the zooids icons to the intended moving directions from the interface. The video can be found here: https://youtu.be/6t_IfJ82EvE. One robot (highlighted in blue) was controlled by our framework and the rest were controlled by human users. In this cooperative task, the team would only receive reward if all agents go to the same goal within the maximum game time limit. There are two goals in the game, goal 1 in the bottom right and goal 2 in the upper left as shown in Fig. 17. The robot agent knows that goal 1 has largest reward and tries to lead the team towards the optimal goal.

Limitations and Future Work. We view our work as a first step into modeling latent, dynamic human team structures. Although our framework is general to different group dynamics and can scale to various team sizes, we do recognize that the performance degrades when the task complexity increases. Examples include when the number of goals or number of agents becomes too large, as shown in Fig. 16. We observe that when increasing the number of agents, the assumption that group dynamics can be explained through local pairwise interactions weakens due to the complexity of interactions. For instance, when we recruited 5 humans to play the predator-prey game, "alliances" emerged where agents that were non-adjacent in the predator-prey chain would team up. These types of dynamics were not observed in two-player games. These types of scenarios are challenging for our models, and the prediction error also accumulates both in the inference process and the optimization process. Further exploration in these complex scenarios is needed to enable our model to be self-aware and corrective.

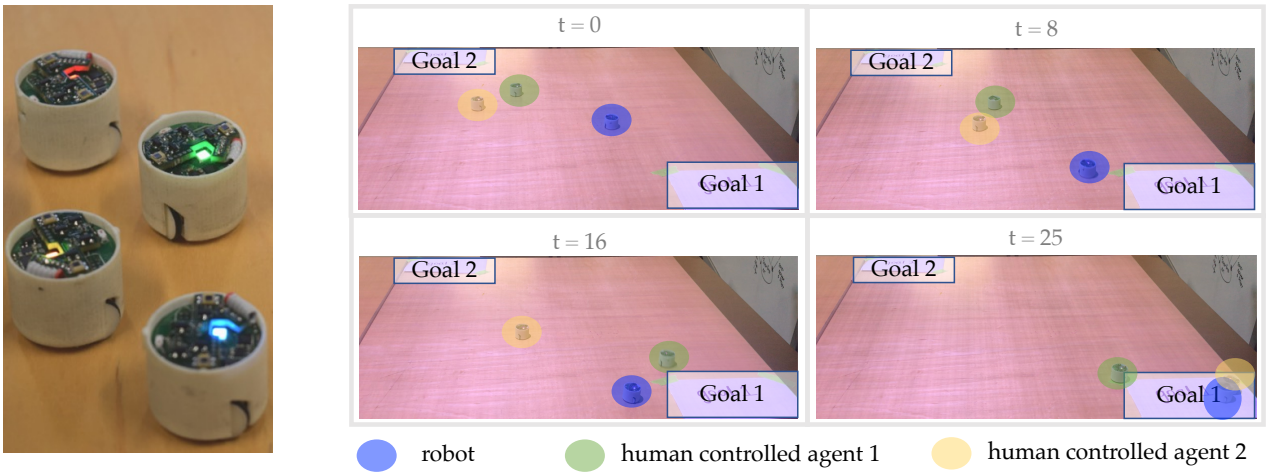
Another limitation is the reliance on simulated human behavior to test our framework. Further experiments with large-scale human data are needed to support our framework's effectiveness for understanding of noisier human behavior.

Finally, the robot policies that use the graph representation are fairly simple. Although this may be a limitation, it is also promising that simple policies were able to perform well using the proposed graph structures.

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

Acknowledgements

We acknowledge funding from the NSF Award #1941722 and Qualcomm Graduate Fellowship for supporting this work.



(a) (b)

Fig. 17: (a) The Zooids robots. (b) Cooperative Zooids robot game snapshots for a 25 second horizon. The highlighted blue robot is controlled by our framework, the rest were controlled by human users. There are two goals in the game, goal 1 in the bottom right and goal 2 in the upper left. The robot tries to aggressively to lead the human team towards the more optimal goal 1.

References

1. Abbeel P, Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the twenty-first international conference on Machine learning, ACM, p 1
2. Agha-Mohammadi AA, Chakravorty S, Amato NM (2014) Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. The International Journal of Robotics Research 33(2):268–304
3. Akgun B, Cakmak M, Jiang K, Thomaz AL (2012) Keyframe-based learning from demonstration. International Journal of Social Robotics 4(4):343–355
4. Albrecht SV (2015) Utilising policy types for effective ad hoc coordination in multiagent systems
5. Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. arXiv preprint arXiv:160706450
6. Bai H, Cai S, Ye N, Hsu D, Lee WS (2015) Intention-aware online pomdp planning for autonomous driving in a crowd. In: 2015 ieee international conference on robotics and automation (icra), IEEE, pp 454–460
7. Baker CL, Saxe R, Tenenbaum JB (2009) Action understanding as inverse planning. Cognition 113(3):329–349
8. Barraquand J, Langlois B, Latombe JC (1992) Numerical potential field techniques for robot path planning. IEEE transactions on systems, man, and cybernetics 22(2):224–241
9. Barrett S (2015) Making friends on the fly: advances in ad hoc teamwork, vol 603. Springer
10. Barrett S, Stone P (2011) Ad hoc teamwork modeled with multi-armed bandits: An extension to discounted infinite rewards. In: Proceedings of 2011 AAMAS Workshop on Adaptive and Learning Agents, pp 9–14
11. Basu C, Biyik E, He Z, Singhal M, Sadigh D (2019) Active learning of reward dynamics from hierarchical queries. In: IROS, pp 120–127
12. Bestick A, Bajcsy R, Dragan AD (2016) Implicitly assisting humans to choose good grasps in robot to human handovers. In: International Symposium on Experimental Robotics, Springer, pp 341–354
13. Biyik E, Palan M, Landolfi NC, Losey DP, Sadigh D (2019) Asking easy questions: A user-friendly approach to active reward learning. In: Proceedings of the 3rd Conference on Robot Learning (CoRL)
14. Bloem M, Bambos N (2014) Infinite time horizon maximum causal entropy inverse reinforcement learning. In: 53rd IEEE Conference on Decision and Control, IEEE, pp 4911–4916
15. Bowling M, McCracken P (2005) Coordination and adaptation in impromptu teams. In: AAAI, vol 5, pp 53–58
16. Brown DS, Kerman SC, Goodrich MA (2014) Human-swarm interactions based on managing attractors. In: Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction, pp 90–97

17. Broz F, Nourbakhsh I, Simmons R (2011) Designing pomdp models of socially situated tasks. In: RO-MAN, 2011 IEEE, IEEE, pp 39–46
18. Çelikkanat H, Şahin E (2010) Steering self-organized robot flocks through externally guided individuals. Neural Computing and Applications 19(6):849–865
19. Chernova S, Thomaz AL (2014) Robot learning from human teachers. Synthesis Lectures on Artificial Intelligence and Machine Learning 8(3):1–121
20. Chu YJ (1965) On the shortest arborescence of a directed graph. Science Sinica 14:1396–1400
21. Couzin ID, Krause J, Franks NR, Levin SA (2005) Effective leadership and decision-making in animal groups on the move. Nature 433(7025):513–516
22. Cristiani E, Piccoli B (2009) A unifying model for the structure of animal groups on the move. arXiv preprint arXiv:09034056
23. Dorsa Sadigh ADD, Sastry S, Seshia SA (2017) Active preference-based learning of reward functions. In: Robotics: Science and Systems (RSS)
24. Doshi F, Roy N (2007) Efficient model learning for dialog management. In: Proceedings of the ACM/IEEE international conference on Human-robot interaction, ACM, pp 65–72
25. Dragan AD, Srinivasa SS (2012) Formalizing assistive teleoperation. MIT Press, July
26. Edmonds J (1968) Optimum branchings. Mathematics and the Decision Sciences, Part 1(335-345):25
27. Finn C, Levine S, Abbeel P (2016) Guided cost learning: Deep inverse optimal control via policy optimization. In: International conference on machine learning, pp 49–58
28. Garcia CE, Prett DM, Morari M (1989) Model predictive control: Theory and practice—a survey. Automatica 25(3):335–348
29. Genter K, Agmon N, Stone P (2013) Ad hoc teamwork for leading a flock. In: Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, International Foundation for Autonomous Agents and Multiagent Systems, pp 531–538
30. Genter KL, Agmon N, Stone P (2011) Role-based ad hoc teamwork. In: Plan, Activity, and Intent Recognition, Citeseer, pp 1782–1783
31. Genter KL, et al. (2017) Fly with me: algorithms and methods for influencing a flock. PhD thesis
32. Giardina I (2008) Collective behavior in animal groups: theoretical models and empirical studies. HFSP journal 2(4):205–219
33. Gombolay MC, Huang C, Shah JA (2015) Coordination of human-robot teaming with human task preferences. In: AAAI Fall Symposium Series on AI-HRI, vol 11, p 2015
34. Gray A, Gao Y, Hedrick JK, Borrelli F (2013) Robust predictive control for semi-autonomous vehicles with an uncertain driver model. In: 2013 IEEE intelligent vehicles symposium (IV), IEEE, pp 208–213
35. Hadfield-Menell D, Russell SJ, Abbeel P, Dragan A (2016) Cooperative inverse reinforcement learning. In: Advances in neural information processing systems, pp 3909–3917
36. Hoshen Y (2017) Vain: Attentional multi-agent predictive modeling. In: Advances in Neural Information Processing Systems, pp 2701–2711
37. Iqbal S, Sha F (2019) Actor-attention-critic for multi-agent reinforcement learning. In: International Conference on Machine Learning, PMLR, pp 2961–2970
38. Javdani S, Admoni H, Pellegrinelli S, Srinivasa SS, Bagnell JA (2018) Shared autonomy via hindsight optimization for teleoperation and teaming. The International Journal of Robotics Research p 0278364918776060
39. Jeon HJ, Losey D, Sadigh D (2020) Shared autonomy with learned latent actions. In: Proceedings of Robotics: Science and Systems (RSS), DOI 10.15607/rss.2020.xvi.011
40. Jiang J, Dun C, Huang T, Lu Z (2018) Graph convolutional reinforcement learning. arXiv preprint arXiv:181009202
41. Kanda T, Hirano T, Eaton D, Ishiguro H (2004) Interactive robots as social partners and peer tutors for children: A field trial. Human-computer interaction 19(1):61–84
42. Karp RM (1971) A simple derivation of edmonds' algorithm for optimum branchings. Networks 1(3):265–272
43. Kerman S, Brown D, Goodrich MA (2012) Supporting human interaction with robust robot swarms. In: 2012 5th International Symposium on Resilient Control Systems, IEEE, pp 197–202
44. Khandelwal P, Barrett S, Stone P (2015) Leading the way: An efficient multi-robot guidance system. In: Proceedings of the 2015 international conference on autonomous agents and multiagent systems, International Foundation for Autonomous Agents and Multiagent Systems, pp 1625–1633
45. Kochenderfer MJ (2015) Decision making under uncertainty: theory and application. MIT press
46. Kurniawati H, Hsu D, Lee WS (2008) Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In: Robotics: Science and systems, Zurich, Switzerland., vol 2008

47. Kwon M, Li M, Bucquet A, Sadigh D (2019) Influencing leading and following in human-robot teams. In: Proceedings of Robotics: Science and Systems (RSS), DOI 10.15607/rss.2019.xv.075
48. Kwon M, Biyik E, Talati A, Bhasin K, Losey DP, Sadigh D (2020) When humans aren't optimal: Robots that collaborate with risk-aware humans. In: Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction, pp 43–52
49. Le Goc M, Kim LH, Parsaei A, Fekete JD, Dragicevic P, Follmer S (2016) Zooids: Building blocks for swarm user interfaces. In: Proceedings of the 29th Annual Symposium on User Interface Software and Technology, pp 97–109
50. Lemon O (2012) Conversational interfaces. In: Data-Driven Methods for Adaptive Spoken Dialogue Systems, Springer, pp 1–4
51. Li M, Losey DP, Bohg J, Sadigh D (2020) Learning user-preferred mappings for intuitive robot control. arXiv preprint arXiv:200711627
52. Li S, Gupta JK, Morales P, Allen R, Kochenderfer MJ (2020) Deep implicit coordination graphs for multi-agent reinforcement learning. arXiv preprint arXiv:200611438
53. Liebner M, Baumann M, Klanner F, Stiller C (2012) Driver intent inference at urban intersections using the intelligent driver model. In: 2012 IEEE Intelligent Vehicles Symposium, IEEE, pp 1162–1167
54. Liemhetcharat S (2013) Representation, planning, and learning of dynamic ad hoc robot teams
55. Littman ML, Stone P (2001) 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
56. Losey DP, Li M, Bohg J, Sadigh D (2019) Learning from my partner's actions: Roles in decentralized robot teams. In: Proceedings of the 3rd Conference on Robot Learning (CoRL)
57. Losey DP, Srinivasan K, Mandlekar A, Garg A, Sadigh D (2020) Controlling assistive robots with learned latent actions. In: International Conference on Robotics and Automation (ICRA), DOI 10.1109/ICRA40945.2020.9197197
58. Macindoe O, Kaelbling LP, Lozano-Pérez T (2012) Pomcop: Belief space planning for sidekicks in cooperative games. In: AIIDE
59. Mavrogiannis CI, Knepper RA (2020) Decentralized multi-agent navigation planning with braids. In: Algorithmic Foundations of Robotics XII, Springer, pp 880–895
60. Mottini A, Acuna-Agost R (2017) Deep choice model using pointer networks for airline itinerary prediction. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 1575–1583
61. Nguyen THD, Hsu D, Lee WS, Leong TY, Kaelbling LP, Lozano-Perez T, Grant AH (2012) Capir: Collaborative action planning with intention recognition. arXiv preprint arXiv:12065928
62. Nikolaidis S, Shah J (2013) 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, IEEE Press, pp 33–40
63. Nikolaidis S, Kuznetsov A, Hsu D, Srinivasa S (2016) Formalizing human-robot mutual adaptation: A bounded memory model. In: The Eleventh ACM/IEEE International Conference on Human Robot Interaction, IEEE Press, pp 75–82
64. Nikolaidis S, Nath S, Procaccia AD, Srinivasa S (2017) Game-theoretic modeling of human adaptation in human-robot collaboration. In: Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, ACM, pp 323–331
65. Omidshafiei S, Agha-Mohammadi AA, Amato C, How JP (2015) Decentralized control of partially observable markov decision processes using belief space macro-actions. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE, pp 5962–5969
66. Ordonez L, Benson III L (1997) Decisions under time pressure: How time constraint affects risky decision making. Organizational Behavior and Human Decision Processes 71(2):121–140
67. Osogami T, Otsuka M (2014) Restricted boltzmann machines modeling human choice. In: Advances in Neural Information Processing Systems, pp 73–81
68. Palan M, Landolfi NC, Shevchuk G, Sadigh D (2019) Learning reward functions by integrating human demonstrations and preferences. In: Proceedings of Robotics: Science and Systems (RSS), DOI 10.15607/rss.2019.xv.023
69. Platt Jr R, Tedrake R, Kaelbling L, Lozano-Perez T (2010) Belief space planning assuming maximum likelihood observations
70. Robins B, Dautenhahn K, Te Boekhorst R, Billard A (2004) Effects of repeated exposure to a humanoid robot on children with autism. Designing a more inclusive world pp 225–236
71. Rosenthal SB, Twomey CR, Hartnett AT, Wu HS, Couzin ID (2015) Revealing the hidden networks of interaction in mobile animal groups allows predic-

tion of complex behavioral contagion. Proceedings of the National Academy of Sciences 112(15):4690–4695
72. Rubenstein M, Cabrera A, Werfel J, Habibi G, McLurkin J, Nagpal R (2013) 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, International Foundation for Autonomous Agents and Multiagent Systems, pp 47–54
73. Sadigh D (2017) Safe and interactive autonomy: Control, learning, and verification. PhD thesis, University of California, Berkeley
74. Sadigh D, Sastry S, Seshia SA, Dragan AD (2016) Planning for autonomous cars that leverage effects on human actions. In: Robotics: Science and Systems, Ann Arbor, MI, USA, vol 2
75. Sadigh D, Sastry SS, Seshia SA, Dragan A (2016) Information gathering actions over human internal state. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp 66–73
76. Sadigh D, Landolfi N, Sastry SS, Seshia SA, Dragan AD (2018) 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
77. Simon HA (1972) Theories of bounded rationality. Decision and organization 1(1):161–176
78. Stone P, Kraus S (2010) To teach or not to teach?: decision making under uncertainty in ad hoc teams. In: AAMAS, pp 117–124
79. Stone P, Kaminka GA, Kraus S, Rosenschein JS, et al. (2010) Ad hoc autonomous agent teams: Collaboration without pre-coordination. In: AAAI, p 6
80. Stone P, Kaminka GA, Kraus S, Rosenschein JS, Agmon N (2013) Teaching and leading an ad hoc teammate: Collaboration without pre-coordination. Artificial Intelligence 203:35–65
81. Strandburg-Peshkin A, Twomey CR, Bode NW, Kao AB, Katz Y, Ioannou CC, Rosenthal SB, Torney CJ, Wu HS, Levin SA, et al. (2013) Visual sensory networks and effective information transfer in animal groups. Current Biology 23(17):R709–R711
82. Tiwari R, Jain P, Butail S, Baliyarasimhuni SP, Goodrich MA (2017) Effect of leader placement on robotic swarm control. In: AAMAS, pp 1387–1394
83. Tversky A, Kahneman D (1992) Advances in prospect theory: Cumulative representation of uncertainty. Journal of Risk and uncertainty 5(4):297–323
84. Wang Z, Schwager M (2016) Force-amplifying n-robot transport system (force-ants) for cooperative planar manipulation without communication. The International Journal of Robotics Research 35(13):1564–1586
85. Xie A, Losey D, Tolsma R, Finn C, Sadigh D (2020) Learning latent representations to influence multi-agent interaction. In: Proceedings of the 4th Conference on Robot Learning (CoRL)
86. Yu CH, Werfel JK, Nagpal R (2010) Collective decision-making in multi-agent systems by implicit leadership
87. Ziebart BD, Maas AL, Bagnell JA, Dey AK (2008) Maximum entropy inverse reinforcement learning. In: AAAI, Chicago, IL, USA, vol 8, pp 1433–1438
88. Ziebart BD, Ratliff N, Gallagher G, Mertz C, Peterson K, Bagnell JA, Hebert M, Dey AK, Srinivasa S (2009) Planning-based prediction for pedestrians. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp 3931–3936