Influencing Leading and Following in Human-Robot Teams

Minae Kwon*
Stanford University

Mengxi Li*
Stanford University

Alexandre Bucquet
Stanford University

Dorsa Sadigh
Stanford University

***Abstract*—Leading and following can emerge naturally in human teams. However, such roles are usually predefined in human-robot teams due to the difficulty of scalably learning and adapting to each agent's roles. Our goal is to enable a robot to learn how leader-follower dynamics emerge and evolve in human teams and to leverage this understanding to influence the team to achieve a goal. We focus on developing an algorithm for robot leaders that can influence human teams. To this end, we develop a mathematical framework that first models emergent leading and following behaviors and further leverages the learned predictive models to plan for intelligent robot teammates that influence the roles of other members for more efficient outcomes. We tackle the task in two steps. First, we develop an effective and concise team structure representation called a leader-follower graph by combining model-based methods and data-driven techniques to uncover the underlying human team latent structures. Then, we optimize for a robot policy that leverages the leader-follower graph to attain an objective by influencing a human team. Our structured representation is scalable with different human team sizes and also generalizable across different tasks. We demonstrate our algorithm on a simulated pursuit-evasion game, where a robot optimizes for three different tasks of reversing a leader-follower relationship, distracting a team, and leading a team towards an optimal goal based on its learned estimate of the leader-follower graph.**

I. Introduction

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

In these collaborative teamwork scenarios, an important challenge for robots is how they can understand and interact with other human agents seamlessly in complex multi-agent systems and may 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 where a drone senses valuable information from the environment. How should it 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 [13, 19, 23, 34, 35, 14, 31], which include learning-from-demonstration tasks where the human is the leader and the robot is the follower [8, 3, 1, 36, 10, 25], or assistive tasks where the robot teaches or assists human users [30, 17]. However, assigning leadership roles a priori is not always feasible in dynamically changing environments or long-term interactions.

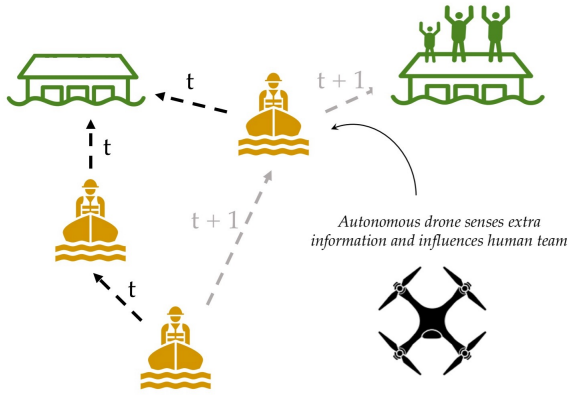


Fig. 1: We estimate leading and following relationships in human teams (denoted by the arrows), and use this to create influential robot policies. The grey arrows represent intended human leading and following behaviors whereas the solid arrows represent updated leading and following behaviors after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

There has also been a lot of prior work on how we can construct intelligent robot policies that induce desired behaviors from people [33, 15, 26, 27, 6]. 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 influencing behaviors by keeping an estimate of the human's state and optimize for actions based on the estimation, which will be computationally infeasible with larger groups of humans. In prior work, 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 [7, 11, 22, 24, 16]. Human's intent inference has also been achieved through human-robot cross-training [25] as well as various other approximations to POMDP solutions such as augmented MDPs, belief space planning, approximating reachable belief space, and decentralization [2, 20, 21, 28, 29, 32]. These approaches do not easily generalize to larger teams of humans due to computational intractability.

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 structure 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*. Teams of humans are able to fluidly take and give up leadership roles depending on the task and changing environment. *Our key insight is that there is an underlying structure encoding the larger and more complex interactions between humans in team settings.*

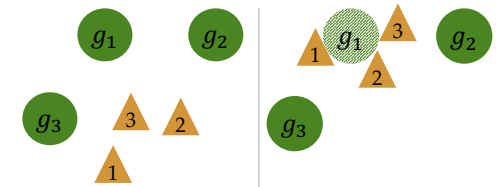
In this paper, we develop a scalable approach to extract meaningful latent structures in teams of humans that represent their leading and following behaviors. We extract an underlying graph, *leader-follower graph*, to represent the *global* pattern of leader-follower dynamics using information from *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 (LFG)* and further influence this structure to more efficiently achieve 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 first goal. However, a robot capable of estimating this underlying structure through the leader-follower graph, can follow strategies that collectively influences the team to instead navigate the team towards the second goal, which could lead to a more desirable outcome.

Our contributions in this paper are as follows:

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

II. Formalism for Modeling Leading and Following in Human Teams

Running Example: Pursuit-Evasion Game. We define a multi-player pursuit-evasion game on a 2D plane as our main running example. In this game, each pursuer is an agent in the set of agents I that can take actions in the 2D space to navigate. There are a number of stationary evaders, which we refer to as *goals*. 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, $A_i = \{\text{move up, move down, move left, move right, stay still}\}$; the action spaces of all agents collectively define the joint action space A. All pursuers must jointly and implicitly agree on a goal to target, and a goal will be captured when all pursuers collide with it as shown in Fig. 2 b.



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

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

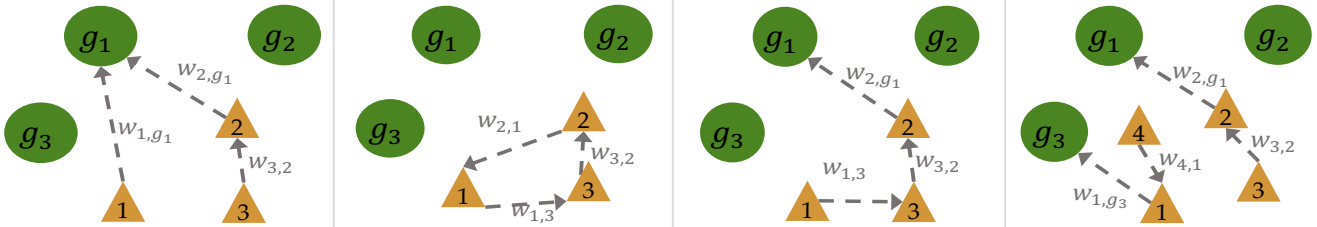
Fig. 2: Pursuit-evasion game.

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 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. 2 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, or 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. III.



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

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

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

(d) Multiple teams.

Fig. 3: Variations of the leader-follower graph

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 are homogeneous and have full information on the observations of themselves and all other agents. (e.g. positions and velocities of agents).

III. Construction of a Leader-Follower Graph

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

A. Pairwise Leadership Scores

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

We propose a general framework of estimating the leadership scores using a supervised learning approach. Consider a two-player setting where $I = \{i, j\}$, we collect labeled data where agent i is asked to follow j, and agent j is asked to optimize for the joint reward function assuming it is leading i, i.e., following a fixed goal g in the pursuit-evasion game ($l_i = j$ and $l_j = g$). We then train a LSTM network with a softmax layer to predicts 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 leader and followers in a two-player setting. We collected a total of 186 games.

Human data is often noisy and it is difficult to collect in large amounts; our model trained on human data did not perform as well as we hoped (validation accuracy = 65%). To get around this problem, we augmented our dataset with synthetic data where we simulated humans playing the pursuit evasion game. Like we did with our human experiments, we randomly assigned leaders for each agent to follow.

We simulated humans based on a potential field path planner [5]. Agents at location q plan their path under the influence of an artificial potential field $U(q)$, which is constructed to reflect the environment. Agents moved toward their leaders by following an attractive potential field. Other agents and goals that were 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.

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

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

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

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

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

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 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.

We simulated 5000 games of each of the three configurations for a two-player game setting, totaling 15000 games. 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 down-sampled 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 [4] to stabilize gradient updates.

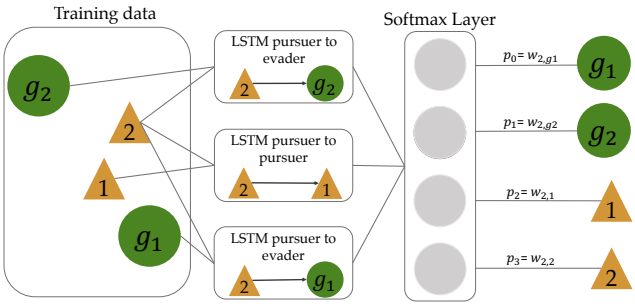


Fig. 4: Scalable neural network architecture. This example is predicting the probability of another agent j being agent 2's leader, $w_{2,j}$. There are three LSTM submodules used because there are two possible evaders and one possible agent that could be agent 2's leader.

Evaluating Pairwise Scores. Our network trained on two-player simulated data performed with training accuracy of 80% and a validation accuracy of 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). Our mixed two-player data model had a training accuracy of 97% and a validation accuracy of 75%. Our model performed with a training accuracy of 72% and a validation accuracy of 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.

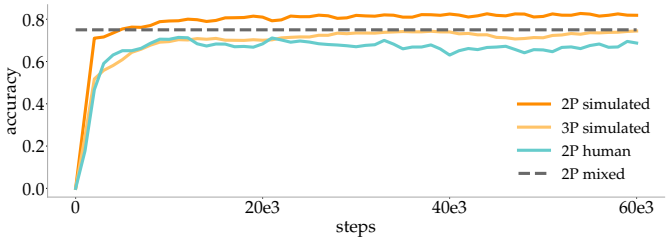


Fig. 5: Validation accuracy when calculating pairwise leadership scores trained on simulated, human and mixed simulated and human data, described in Sec. III-A

B. Maximum Likelihood Leader-Follower Graph in Teams

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 on the left.

In order 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. 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 from Edmonds' algorithm [12, 9], which finds a maximum weight arborescence [18] 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 I). 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. 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. 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 I). In all experiments shown in Table I, 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 V.

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

Training Data	Testing Data	LFG Accuracy	Random Accuracy
2 players, simulated	3 players, simulated	0.67	0.29
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

IV. Planning based on Inference on 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 robot can use the leader-follower graph to infer useful team structures. We then describe how a robot can leverage these inferences to plan for a desired outcome.

A. Inference based on leader-follower graph

Leader-follower graphs allow 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.

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. 6(c)(d), 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. 6(d).

B. 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: Outcomes 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 $w_{i,g}$.

In order to select actions $a \in \mathcal{A}$ 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 iterate over the robot's action space, assuming it is discrete and tractable, and choose 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.

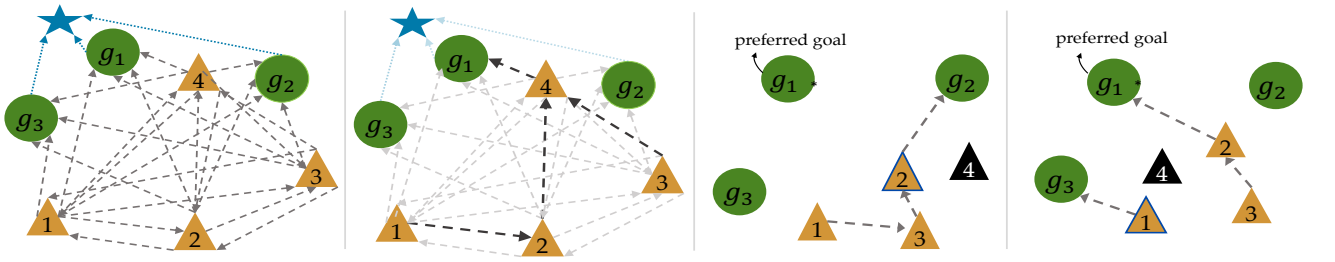
$$a^*_t = \underset{a_t \in \mathcal{A}}{\operatorname{argmax}} \quad r\left(\{w^{t+k}_{i,j}(a_t)\}_{i,j \in I}, \{w^{t+k}_{i,g}(a_t)\}_{i \in I, g \in G}\right) \tag{2}$$

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

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

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

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. 3d) by re-directing the edges of one sub-group to follow another.



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

(b) Maximum-likelihood leader-follower graph, $\mathcal{G}^*$. Edmonds' algorithm prunes graph $\mathcal{G}$ to produce a maximum-weight arborescence, shown in the bolded edges.

(c) Most influential leader is agent 2

(d) Most influential leader trivially becomes agent 1 since agents 2 and 3 are already targeting the optimal goal, g_1*.

Fig. 6: On left: creating a maximum-likelihood leader-follower graph, $\mathcal{G}^*$. On right: examples of influential leaders in $\mathcal{G}^*$.

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. 3b.

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

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

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^*:

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

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.

V. Experiments

With an optimization framework that plans for robot actions using the leader-follower graph, we put our learned leader-follower graph to the test. We evaluate our LFG on three different tasks that involve influencing multiagent human teams. For each task, we compare task performance with robot policies that use the LFG against robot policies that do not. Across all tasks, we find that robot policies that use the leader-follower graph perform well compared to other policies, showing that our graph can be easily generalized to different settings.

Task setup. Our tasks take place in the pursuit-evasion domain. Within each task, we conduct experiments with simulated human behavior. 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, 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, human can move 1 unit in one of the four directions: up, down, left, right, or stay at its position. Robot agent's action space is the same but with larger move amount 5. Maximum game time limit is 1000.

A. Changing edges of the leader-follower graph

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 $\theta_g = 0.6$, $\theta_a = 0.4$, 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 $w_{i,r}$, given by the leader-follower graph.

Results. We show that the robot can influence a human agent to follow it. Fig. 7 contains averaged probabilities over ten tasks. The probability of the robot being the human agent's leader $w_{i,r}$ increases over time, and converges to around 57%.

B. Adversarial task

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

In this task experiment setting, there are m goals and n players in the pursuit-evasion game. Among the n players, we

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

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

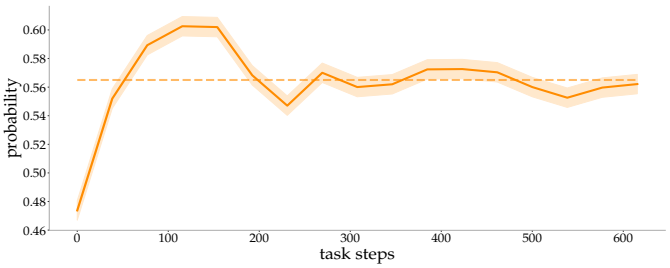


Fig. 7: Probabilities of a human agent following the robot over 10 tasks. The robot is successfully able to become the human agent's leader as the task progresses.

have 1 robot agent and $n-1$ human agents. All the human agents are homogeneous and predisposed in the same pattern as in task *obtaining a desired leader-follower graph*. $n-2$ 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 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 agent's way would not be a complete solution, since we allow for an agent to be absent of capturing an goal.

Methods. We test optimization methods based on constructed Leader-Follower Graph along with other baseline models.

We experimented with 3 baseline heuristics strategies without knowledge of LFG. *Random* strategy is that the robot agent just moves randomly. *To_one_player* strategy is that the robot agent selects a human agent and then goes towards it trying to block its way. *To_farthest_goal* strategy is that robot selects the goal that the average distance to human players are largest and then go to that goal in the hope that human agents would get influenced or may further change goal by observing that some players are heading for another goal.

We also experimented with two optimization models based on the LFG. *LFG_closest_pursuer* involves the robot agent selecting the closest pursuer and choosing an action to maximize the probability of the pursuer following it (as predicted by the LFG). Similarly, *LFG_influential_pursuer* strategy involves the robot targeting the most influential human agent predicted by the LFG described in Sec. IV and then conducting the same optimization of maximizing the following probability.

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 bssed on LFG consistently outperforms methods without knowledge of LFG. The experimental results with varying number of players are summarized in Table II and visualized in Fig. 8. As could be seen from Fig. 8, average game time goes up as the number of players increases. This is because it's more challenging for more players to reach agreement on which goal to capture and thus takes longer time. Experiment results with varying number of evaders are also summarized in Table III for reference. The consistent advantageous performance suggests the effectiveness of LFG for inference and optimization in this scenario.

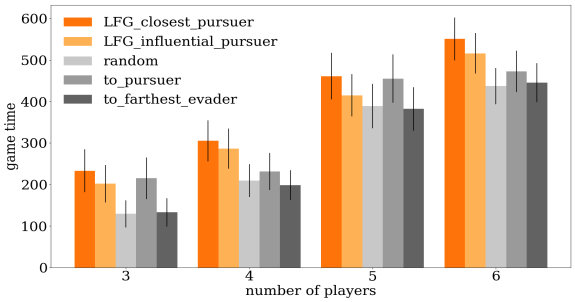


Fig. 8: Average time over 50 games in 2 goal games with different number of players across all baseline methods and our model

Fig. 9: adversarial game snapshots. (orange is human agent and black is the robot agent)

To demonstrate robot behavior in the adversarial game, we also took snapshots of one game as in Fig. 9. 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.

C. Cooperative task

Finally, we evaluate the robot in a cooperative setting where the robot tries to be helpful for human teams. The aim of the

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

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

robot is to lead its teammates in a way that everyone can reach a target goal that gives the team the greatest joint utility $g^* \in G$. g^* is not immediately observable to all teammates. We assume a setting where only the robot knows where g^* is.

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 greatest joint utility g^* is captured and failed if any other suboptimal goal is captured or the game time exceeds the 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 tested 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 IV. In this robot leading human team scenario, going directly to the desired goal is a very strong baseline since it already conveys the message to other players that the robot is going for a specific goal. This baseline strategy is especially effective when the game is not complex, i.e., the number of goals is small. However, our model based on the LFG still demonstrates competitive performance compared to it. Especially 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 initial conditions. We took snapshots of one game as in Fig. 10. In this game, the robot approaches other agents and the desired goal in the collaborative pattern, trying to help catch the goal g_1.

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

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

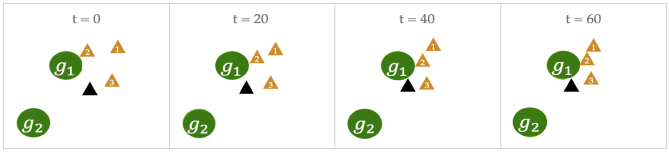


Fig. 10: adversarial game snapshots. (orange is human agent and black is the robot agent)

VI. Discussion

Summary. We propose an approach for modeling leading and following behavior in multi-agent human teams. We use a combination of data-driven and graph-theoretic techniques to learn a Leader-Follower Graph. This graph representation encoding human team hidden structure is scalable with the team size (number of agents), since we base the model on *local*, pairwise relationship prediction and combine them to create a *global* model. We demonstrate the effectiveness of the leader-follower graph 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.

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

For future work, we plan to test our model on large scale human-robot experiments in both simulation and using navigation platforms of robot swarms to further improve our model's generalization capacity. We also plan on experimenting with combining the LFG with more advanced policy learning methods such as reinforcement learning. We think the

LFG could contribute to multi-agent reinforcement learning in various ways such as reward design and state representation.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[26] Stefanos Nikolaidis, Anton Kuznetsov, David Hsu, and Siddharta Srinivasa. Formalizing human-robot mutual adaptation: A bounded memory model. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, pages 75–82. IEEE Press, 2016.

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

[28] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi,

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

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

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

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

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

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

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

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

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

VII. APPENDIX

A. Potential Field for Simulated Human Planning

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

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

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

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