Stackelberg Meta-Learning for Strategic Guidance in Multi-Robot Trajectory Planning

Yuhan Zhao and Quanyan Zhu

arXiv:2211.13336v2 [cs.RO] 29 Jul 2023

Abstract—Trajectory guidance requires a leader robotic agent to assist a follower robotic agent to cooperatively reach the target destination. However, planning cooperation becomes difficult when the leader serves a family of different followers and has incomplete information about the followers. There is a need for learning and fast adaptation of different cooperation plans. We develop a Stackelberg meta-learning approach to address this challenge. We first formulate the guided trajectory planning problem as a dynamic Stackelberg game to capture the leader-follower interactions. Then, we leverage meta-learning to develop cooperative strategies for different followers. The leader learns a meta-best-response model from a prescribed set of followers. When a specific follower initiates a guidance query, the leader quickly adapts to the follower-specific model with a small amount of learning data and uses it to perform trajectory guidance. We use simulations to elaborate that our method provides a better generalization and adaptation performance on learning followers' behavior than other learning approaches. The value and the effectiveness of guidance are also demonstrated by the comparison with zero guidance scenarios¹.

I. INTRODUCTION

Guided cooperation is gaining increasing attention in many robotic applications with the advances in robotic research and technology. For example, the path guidance and tracking in multi-robot systems [1], [2], human-robot collaboration in home-assistive services [3] and manufacturing [4], and multirobot collective transportation [5], [6]. Guided cooperation can utilize heterogeneous robot capabilities to achieve task objectives. A more resourceful robotic agent (leader) can guide or assist a less sophisticated robotic agent (follower) to complete the task by utilizing both agents' advantages. A typical and important application of guided cooperation in robotics is strategic guidance for trajectory planning. As an example, we consider an unmanned aerial vehicle (UAV) guiding an unmanned ground vehicle (UGV) for transporting mission-critical objects. The UGV has limited sensing and computational resources and can only do its local planning, resulting in difficulty reaching the destination independently. In contrast, the resourceful UAV can sense the global environment and find a collision-free trajectory to guide the UGV.

Extensive works in trajectory planning, including pathbased planning [7]–[9] and control-based planning [10],



Fig. 1: Illustration of Stackelberg meta-learning approach in trajectory guidance. Different follower UGVs rely on the leader UAV's trajectory guidance to reach their destinations. The leader UAV uses meta-learning to learn a meta-best-response model by interacting with different followers (1-2). When guiding a specific follower, the leader uses the follower-specific data (3) to adapt the meta-model to that follower (4) and performs guided trajectory planning (5).

[11], have investigated single-agent cases. However, simply extending the methodology to multi-agent settings is insufficient since it fails to capture the cooperative interactions between multiple agents. Game theory emerges as a promising tool in multi-robot trajectory planning [12]–[16]. In particular, Stackelberg game [17] provides a quantitative framework that captures the mentor-apprentice or leaderfollower type of interactions in guided cooperation and has shown effectiveness in various cooperative tasks [18]–[21]. In the UAV-UGV example, interactions between UAV and UGV in trajectory guidance can be formulated as a dynamic Stackelberg game. The associated Stackelberg equilibrium solution provides an agent-wise cooperative plan.

However, the leader must know the follower's behavior model to solve the Stackelberg equilibrium, which is challenging in practical applications where the leader can only observe the follower's action instead of his model. Besides, a leader generally needs to work with different followers on various tasks. In the trajectory guidance example, a UAV is expected to assist UGVs of different configurations to accomplish the transport mission. Designing different cooperation plans for various followers can be time-consuming and eventually becomes intractable as the number of followers increases. We need an approach to quickly generate a guidance plan when the leader assists a specific follower.

Meta-learning provides a suitable learning mechanism for Stackelberg game-theoretic cooperation (see Fig. 1). It enables learning a customizable plan from a prescribed set of known tasks and fast adaptation to a specific task using a small amount of learning data [22]. It has also

Y. Zhao and Q. Zhu are with the Department of Electrical and Computer Engineering, New York University, Brooklyn, NY, 11201, USA. Email:{yhzhao, qz494}@nyu.edu.

This work is accepted by 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

¹The simulation codes are available at https://github.com/ yuhan16/Stackelberg-Meta-Learning.

been used in robotics to search for adaptive collaboration plans in multi-robot systems [23], human-robot interaction [24], [25], and trajectory planning and tracking [26], [27]. Specifically, the leader prepares a meta-best-response for different followers as an averaged behavior model based on past interactions. When a specific follower initiates a guidance query, the leader adapts the meta-model to the follower and then performs trajectory guidance by finding an approximate Stackelberg equilibrium.

In this work, we develop a Stackelberg meta-learning approach for leader-follower cooperation in guided trajectory planning problems. We formulate the guided interactions between the leader and the follower as a dynamic Stackelberg game, where each follower makes *myopic* decisions to interact with the leader. Once a specific follower initiates a guidance query, the leader quickly generates a followerspecific behavior model from the meta-best-response and uses it to design effective trajectory guidance strategies with receding horizon planning. We use simulations to show that our approach provides a better generalization and adaptation performance compared with the other two typical learning approaches. We also compare the results with zero guidance scenarios to demonstrate the value of guidance in trajectory planning.

Notations: We use superscripts L and F to denote the leader and the follower-related quantities. We use bold variables to denote aggregated trajectory, e.g., $\mathbf{u} := \{u_t\}_{t=0}^{T-1}$. We use the two-norm $\|x\|_2 = \sqrt{x^{\mathsf{T}}x}$ and the matrix norm $\|x\|_Q = \sqrt{x^{\mathsf{T}}Qx}$.

II. PROBLEM FORMULATION

A. Trajectory Guidance as Stackelberg Games

We consider a leader robotic agent L (she) assisting the follower robotic agent F (he) to reach the destination while circumventing obstacles in the working environment $\mathcal{X} \subset \mathbb{R}^2$. Due to limited sensing and computational capabilities, the follower requires guidance from the leader to find a collision-free trajectory. The leader is expected to assist different followers in trajectory planning. The followers with different configurations are characterized by their type $\theta \in$ Θ , and $p(\theta)$ is the type distribution known to the leader. The leader assists one follower at a time when a specific follower drawn from $p(\theta)$ requests a guidance query.

Let $x_t^L \in \mathbb{R}^{n^L}$, $x_t^F \in \mathbb{R}^{n^F}$ denote the leader and follower's states, including their positions at time t; $x_t := [x_t^L, x_t^F]$ denotes the joint state. Let $u_t^L \in \mathcal{U}^L \subset \mathbb{R}^{m^L}$, $u_t^F \in \mathcal{U}^F \subset \mathbb{R}^{m^F}$ be their controls and admissible control sets. In the guidance of a follower with type θ , the leader announces an action u_t^L based on x_t . The follower then myopically responds to the leader with the optimal action $u_{\theta}^{F*}(x_t, u_t^L)$ based on his cost function J_{θ}^F . Using the follower's response, the leader seeks a collision-free trajectory $\mathbf{x}^* := \{x_t^*\}_{t=0}^T$ for both agents and strategically guides the follower to the destination. We formulate the trajectory guidance problem as

a dynamic Stackelberg game \mathcal{G}_{θ} as follows:

$$\min_{u^L \in \mathcal{U}^L} J^L_{\theta}(\mathbf{u}^L) := \sum_{t=0}^{T-1} g^L_{\theta}(x_t, u^L_t, u^{F*}_{\theta}(x_t, u^L_t)) + q^L_{\theta}(x_T)$$
(1)

s.t.
$$u_{\theta}^{F*}(x_t, u_t^L) = \arg\min_{u^F \in \mathcal{U}^F} J_{\theta}^F(x_t, u_t^L, u_t^F),$$

 $t = 0, \dots, T-1, \quad (2)$

$$x_{t+1}^L = f^L(x_t^L, u_t^L), \quad t = 0, \dots, T-1,$$
 (3)

$$\begin{aligned} x_{t+1}^F &= f_{\theta}^F(x_t, u_t^L, u_t^F), \quad t = 0, \dots, T-1, \quad (4) \\ \operatorname{dist}(x_t^i, p_j^O) &\geq d_j, \quad i \in \{L, F\}, \ j = 1, \dots, M, \end{aligned}$$

$$t = 0, \dots, T. \tag{5}$$

Here, g_{θ}^{L} and q_{θ}^{L} denote the leader's stage and terminal costs. The follower's problem is given by (2) where J_{θ}^{F} is his cost function. The leader and follower's dynamic models are captured by (3) and (4), respectively. In the safety constraints (5), $p_{j}^{O} \in \mathcal{X}$ is the position of the *j*-th obstacle in \mathcal{X} and d_{j} is the safety distance. The function dist measures the distance between the leader/follower and an obstacle.

We use the open-loop Stackelberg equilibrium $\langle \mathbf{x}^*, \mathbf{u}^{L*}, \mathbf{u}^{F*}_{\theta} \rangle$ of the game \mathcal{G}_{θ} as the cooperative plan for trajectory guidance. The leader and the follower take the actions in \mathbf{u}^{L*} and \mathbf{u}^{F*}_{θ} respectively and generates a collision-free trajectory starting from given $x_0 := [x_0^L, x_0^F]$.

1) Guidance in Trajectory Planning: Guidance is an interactive process instead of unilateral instructions from the leader. Followers have the freedom to (myopically) decide where to go after observing the state x_t and the leader's action u_t^L . The leader's trajectory serves as a reference to assist followers to perform effective local planning. When there is no guidance, followers can only perform one-step planning using limited sensing and planning capabilities. It is worth noting that the leader's recommended action coincides with the follower's optimal action u_{θ}^{F*} only if the leader knows the follower's exact decision-making model. In this case, followers follow the recommendation and do not need local planning. However, the anticipated action may not be taken if the leader only has an approximate follower model. Then all followers need to make decisions by themselves in the guidance task.

B. Meta-Best-Response and Meta-Learning Problem

When the leader knows the follower's decision-making model $(J_{\theta}^{F} \text{ or } u_{\theta}^{F*})$, the Stackelberg equilibrium can be computed using model-based approaches, such as mixed integer linear programming [21], [28]. However, the leader can only observe the follower's actions in many practical cases. We need learning-based methods to find the Stackelberg equilibrium of \mathcal{G}_{θ} to design effective guidance plans.

We approximate the follower's decision problem (2) using an input-output function (i.e., the best-response model) $b(x, u^L; \mathbf{w})$ parameterized by \mathbf{w} . The leader uses b to predict the follower's response and \mathcal{G}_{θ} becomes a parameterized single-agent trajectory optimization problem, denoted by $\mathcal{G}_{\theta}(\mathbf{w})$ for a given parameter \mathbf{w} . The optimal solution $\{\mathbf{x}^*(\mathbf{w}), \mathbf{u}^{L*}(\mathbf{w})\}\$ of $\mathcal{G}_{\theta}(\mathbf{w})$ together with the estimated best response $\{b(x_t^*(\mathbf{w}), u_t^{L*}(\mathbf{w}))\}_{t=1}^{T-1}$ approximate the Stackelberg equilibrium of \mathcal{G}_{θ} and is used for trajectory guidance.

When the leader assists different followers, it is timeconsuming for the leader to learn separate best-response models from scratch. We aim to develop an approach to quickly generate a guidance plan when a follower initiates a query. To this end, we formulate a meta-learning problem such that the leader learns a meta-best-response model from a set of sampled or encountered followers through offline learning. When a new query occurs, the leader uses a small amount of learning data (from history) to adapt the meta-best-response model to the follower-specific one and performs trajectory guidance.

With a little abuse of notation, we denote $b(x, u^L; \mathbf{w})$ as the meta-best-response model and define task \mathcal{T}_{θ} as learning the best-response of the type θ follower with the task cost

$$L_{\theta}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \left\| b(\hat{x}_i, \hat{u}_i^L; \mathbf{w}) - \hat{u}_{\theta, i}^{F*} \right\|_2^2, \quad (6)$$

where $\mathcal{D}_{\theta} = \{(\hat{x}_i, \hat{u}_i^L, \hat{u}_{\theta,i}^{F*})\}_{i=1}^N$ is the sampled best-response data. $L_{\theta}(\mathbf{w})$ measures the data fitting cost over \mathcal{D}_{θ} . The meta-learning can be formulated as the following optimization problem:

$$\min_{\mathbf{w}} \quad \mathbb{E}_{\theta \sim p} \left[\mathbb{E}_{\mathcal{D}_{\theta}} [L_{\theta}(\mathbf{w} - \alpha \nabla_{\mathbf{w}} L_{\theta}(\mathbf{w}))] \right].$$
(7)

where $\alpha > 0$ is the inner gradient update step size.

III. STACKELBERG META-LEARNING

A. Meta-Best-Response Training

We use the empirical task distribution to approximate the expectation in (7) and obtain

$$\min_{\mathbf{w}} \quad \sum_{\theta \sim p} \sum_{\mathcal{D}_{\theta}} L_{\theta} \left(\mathbf{w} - \alpha \nabla_{\mathbf{w}} L_{\theta}(\mathbf{w}; \mathcal{D}_{\theta}^{\text{train}}); \mathcal{D}_{\theta}^{\text{test}} \right).$$
(8)

Here, we split the dataset $\mathcal{D}_{\theta} = \mathcal{D}_{\theta}^{\text{train}} \cup \mathcal{D}_{\theta}^{\text{test}}$; $\theta \sim p$ is the empirical task distribution of sampled batch tasks $\mathcal{T}_{\text{batch}}$ from p. We use first-order methods to solve (8). The intermediate parameter \mathbf{w}_{θ}' for task \mathcal{T}_{θ} is updated by one gradient step:

$$\mathbf{w}_{\theta}' \leftarrow \mathbf{w}_k - \alpha \nabla_{\mathbf{w}} L_{\theta}(\mathbf{w}_k; \mathcal{D}_{\theta}^{\text{train}}).$$
(9)

Next, we perform stochastic gradient descent (SGD) on the meta-optimization across T_{batch} with the step size $\beta > 0$:

$$\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \frac{\beta}{|\mathcal{T}_{\text{batch}}|} \sum_{\theta \in \mathcal{T}_{\text{batch}}} \nabla_{\mathbf{w}} L_{\theta}(\mathbf{w}_{\theta}'; \mathcal{D}_{\theta}^{\text{test}}), \quad (10)$$

Alg. 1 summarizes the Stackelberg meta-learning algorithm and outputs a meta-best-response model as the averaged behavior model for all types of followers.

1) Importance Sampling: The follower's action near an obstacle differs from that in the flat region. The best-response model should capture the follower's response in both situations as precisely as possible. Using the idea of importance sampling, we randomly sample K_1 data in \mathcal{X} and sample K_2 data near the obstacles and use $\kappa := K_1/K_2$ to control the sampling ratio.

Algorithm 1: Stackelberg meta-learning algorithm.

- 1 **Input:** Follower's type distribution $p(\theta)$, hyperparameters α, β , MAX_ITER;
- 2 Initialize: Initial model parameter \mathbf{w}_0 , ;

$$\mathbf{3} \ k \leftarrow 0$$
;

- 4 while $k < MAX_{ITER}$ do
- Sample a batch of tasks $\mathcal{T}_{batch} := \{\mathcal{T}_{\theta}\}, \theta \sim p$; 5 // Inner level gradient evaluation for every task $\mathcal{T}_{\theta} \in \mathcal{T}_{batch}$ do 6 Sample K_1 best response data given random 7 feasible state-action pair (x, u^L) ; Sample K_2 best response data near the 8 obstacle $j = 1, \ldots, M$; $\mathcal{D}_{\theta}^{\text{train}} \leftarrow \text{all samples with } K = K_1 + K_2$; Compute \mathbf{w}_{θ}' by (9) and $\mathcal{D}_{\theta}^{\text{train}}$; 10 $\mathcal{D}_{\theta}^{\text{test}} \leftarrow \text{sample } K \text{ best response data using}$ 11 \mathbf{w}_{θ}' , following same sample rule as $\mathcal{D}_{\theta}^{\text{train}}$; // Meta-optimization evaluation Update \mathbf{w}_{k+1} by (10) and $\{\mathcal{D}_{\theta}^{\text{test}}\}_{\theta \sim p}$; 12 $k \leftarrow k+1$; 13 14 $\mathbf{w}_{\text{meta}} \leftarrow \mathbf{w}_k$;
- 15 **Output:** meta-parameter \mathbf{w}_{meta} and meta-best-response model $b(x, u^L; \mathbf{w}_{meta})$;

B. Best-Response Adaption

After receiving $b(x, u^L; \mathbf{w}_{meta})$ from Alg. 1, the leader can fast adapt to the new task (the new follower) using only a small amount of data samples when a guidance query occurs. Specifically, the leader adapts $b(x, u^L; \mathbf{w}_{meta})$ to a followerspecific best-response model $b(x, u^L; \mathbf{w}_{\theta})$ using the adaption algorithm Alg. 2 and uses it to perform trajectory guidance.

Algorithm 2: Adaption of Stackelberg meta-learning.

- Input: w_{meta} and new follower with type θ;
 Initialize: Step size parameter α, adaption step C;
 Sample D_θ with K' samples using w_{meta};
 k ← 0, w₀ ← w_{meta};
 while k < C do
 w_{k+1} ← w_k α∇_wL_θ(w_k) with D_θ;
 k ← k + 1;
 Adapted parameter w_θ ← w_k;
 Output: adapted parameter w_θ and adapted best
 - response model $b(x, u^L; \mathbf{w}_{\theta})$;

C. Receding Horizon Planning For Trajectory Guidance

The leader uses the adapted best-response model and solves the problem $\mathcal{G}_{\theta}(\mathbf{w})$ to perform trajectory guidance. The safety constraints (5) bring computational challenges. We penalize the violation of safety constraints using barrier functions for $j = 1, \ldots, M$:

$$c_j(x_t) = -\sum_{i \in \{L,F\}} \nu \log \left(\operatorname{dist}(x_t^i, p_j^O) - d_j \right),$$

where $\nu > 0$ is the penalty parameter. Then, the modified cost function of $\mathcal{G}_{\theta}(\mathbf{w})$ is given by

$$\widetilde{J}_{\theta}^{L}(\mathbf{u}^{L};\mathbf{w}) :=$$

$$\sum_{t=0}^{T-1} \left[\widetilde{g}_{\theta}^{L}(x_{t}, u_{t}^{L}; \mathbf{w}) + \sum_{j=1}^{M} c_{j}(x_{t}) \right] + q_{\theta}^{L}(x_{T}) + \sum_{j=1}^{M} c_{j}(x_{T}^{i}),$$

where $\widetilde{g}_{\theta}^{L}(x_t, u_t^{L}; \mathbf{w}) := g_{\theta}^{L}(x_t, u_t^{L}, b(x_t, u_t^{L}; \mathbf{w}))$. Instead of solving $\mathcal{G}_{\theta}(\mathbf{w})$. We solve the following modified problem

$$\mathcal{G}_{\theta}(\mathbf{w}): \min_{\mathbf{u}^{L} \in \mathcal{U}^{L}} J_{\theta}^{L}(\mathbf{u}^{L}; \mathbf{w})$$

s.t. $x_{t+1}^{L} = f^{L}(x_{t}^{L}, u_{t}^{L}), \ t = 0 \dots, T-1,$
 $x_{t+1}^{F} = f_{\theta}^{F}(x_{t}, u_{t}^{L}, b(x_{t}, u_{t}^{L}; \mathbf{w})), \ t = 0, \dots, T-1,$

and uses its solution $\{\widetilde{\mathbf{x}}^*(\mathbf{w}), \widetilde{\mathbf{u}}^{L*}(\mathbf{w})\}\$ to approximate the one of $\mathcal{G}_{\theta}(\mathbf{w})$. We can leverage existing optimization solvers to solve $\widetilde{\mathcal{G}}_{\theta}(\mathbf{w})$. However, we note that the follower's dynamics f_{θ}^F contain the parameterized best-response model $b(x, u^L; \mathbf{w})$. Parameterized models in general have complex function surfaces, such as neural networks [29]. The direct use of the parameterized model in the equality constraint may result in infeasible solutions. To address this issue, we penalize the follower's dynamics difference by putting it into the objective function. We define

$$d_t(x_{t+1}, x_t, u_t^L; \mathbf{w}) = \mu \left\| x_{t+1}^F - x_t^F - f_\theta^F(x_t, u_t^L; \mathbf{w}) \right\|_2^2$$

for t = 0, ..., T - 1, where $\mu > 0$ is the penalty parameter. Hence we solve the following problem

$$\widetilde{\mathcal{G}}_{\theta}^{\text{opt}}(\mathbf{w}): \min_{\mathbf{u}^{L} \in \mathcal{A}} \quad \widetilde{J}_{\theta}^{L}(\mathbf{u}^{L}; \mathbf{w}) + \sum_{t=0}^{T-1} d_{t}(x_{t}, x_{t}, u_{t}^{L}; \mathbf{w}),$$

s.t. $x_{t+1}^{L} = f^{L}(x_{t}^{L}, u_{t}^{L}), \quad t = 0 \dots, T-1,$

to obtain an approximate solution. We further use the necessary optimality conditions of $\tilde{\mathcal{G}}_{\theta}(\mathbf{w})$ derived from Pontryagin's Minimum Principle (PMP) to refine the approximate solution of $\tilde{\mathcal{G}}_{\theta}^{\text{opt}}(\mathbf{w})$:

$$\begin{aligned} x_{t+1} &= f(x_t, u_t^L; \mathbf{w}), \quad t = 0, \dots, T - 1, \\ \lambda_t &= \nabla_{x_t} H_t(x_t, u_t^L, \lambda_{t+1}), \quad t = 1, \dots, T - 1, \\ \lambda_T &= \nabla_{x_T} q_{\theta}^L(x_T) + \nabla_{x_T} \sum_{j=1}^M c_j(x_T), \\ u_t^L &= \arg\min_{u \in \mathcal{U}^L} H_t(x_t, u, \lambda_{t+1}), \ t = 0, \dots, T - 1, \end{aligned}$$
(11)

where $f(x_t, u_t^L; \mathbf{w}) := [f^L(x_t, u_t^L), f^F_{\theta}(x_t, u_t^L, b(x_t, u_t^L; \mathbf{w}))]$ is the aggregated dynamics, $\lambda_t \in \mathbb{R}^{n^A + n^B}$ is the costate at time t, and the Hamiltonian H_t is given by

$$H_t(x_t, u_t^L, \lambda_{t+1}) :=$$

$$\widetilde{g}_{\theta}^L(x_t, u_t^L; \mathbf{w}) + \sum_{j=1}^M c_j(x_t) + \lambda_{t+1}^{\mathsf{T}} f(x_t, u_t^L; \mathbf{w})$$

for t = 0, ..., T - 1. Gradient methods can be applied to find the minimizer of H_t in (11).

Based on $\widetilde{\mathcal{G}}_{\theta}^{\text{opt}}$ and (11), we use receding horizon planning to generate effective and robust trajectory guidance strategies, which is summarized in Alg. 3.

Algorithm 3: Receding horizon planning.

1 Input: Query type $\theta \in \Theta$, initial state x_{init} , destination p^{d} , MAX_TIME ; 2 Run Alg. 2 to obtain the adapted model $b(x, a; \mathbf{w}_{\theta})$; 3 $x^{d} \leftarrow \text{form target state}$; 4 $t \leftarrow 0, x_t \leftarrow x_{\text{init}};$ 5 while True do Leader sets x_t as the initial state in $\widetilde{\mathcal{G}}_{\theta}^{\text{opt}}(\mathbf{w})$; 6 $\bar{\mathbf{u}}^L \leftarrow \text{Leader solves } \widetilde{\mathcal{G}}^{\text{opt}}_{\theta}(\mathbf{w}) ;$ 7 $\widetilde{\mathbf{u}}^{L*}(\mathbf{w}) \leftarrow \text{Leader refines } \widetilde{\mathbf{u}}^{L} \text{ by solving (11) ;}$ 8 Leader announces $\widetilde{u}_t^{L*}(\mathbf{w})$ to the follower ; 9 $u_t^{F*} \leftarrow$ follower observes x_t and \widetilde{u}_t^{L*} ; 10 $x_{t+1} \leftarrow$ real system dynamics (3)-(4); 11 if Reach destination or $t > MAX_TIME$ then 12 break 13 14 $x_t \leftarrow x_{t+1};$ $t \leftarrow t + 1$; 15

IV. SIMULATIONS AND EVALUATIONS

A. Simulation Settings

We choose a $[0, 10] \times [0, 10]$ working space \mathcal{X} with four obstacles shown in Fig. 3. We use a single integrator $\dot{p}^L = v^L$ as the leader's dynamic model, where $p^L, v^L \in \mathbb{R}^2$ are the leader's position and velocity. We use a unicycle model to all followers: $\dot{\phi}^F = \omega^F, \dot{p}^F_x = v^F \cos(\phi^F), \dot{p}^F_y = v^F \sin(\phi^F),$ where $\phi^F \in (-\pi, \pi], p^F := [p^F_x, p^F_y] \in \mathbb{R}^2$ denotes the rotation angle and x, y-positions. $\omega^F, v^F \in [-1, 1]$, are the input angular velocity and linear velocity. We assume that the leader and all followers know the dynamic models. We denote the joint state $x := [p^L, p^F, \phi^F], u^L := v^L$, and $u^F := [v^F, \omega^F]$. The interaction (planning) time is set as T = 2s, and the discretization time step is dt = 0.2s. The discrete-time model is discussed in Appendix A. We assume that all followers have the same destination $p^{d} = [9, 9]$. For simplicity, we elaborate on the definitions of the leader and followers' cost functions in Appendix B. The penalty parameters $\nu = 0.5$ and $\mu = 50$.

We consider five types of followers $(|\Theta| = 5)$ with a type distribution p = [0.2, 0.3, 0.1, 0.3, 0.1]. Using the cost function definition (14), the follower in each type has a different set of parameters $\mathbf{c} := [c_1, \ldots, c_4]$. Type 1: $\mathbf{c} = [1, 8, 1, 0.8]$. Type 2: $\mathbf{c} = [1, 10, 2, 0.7]$. Type 3: $\mathbf{c} = [1, 10, 2, 0.6]$. Type 4: $\mathbf{c} = [1, 5, 0.5, 1]$. Type 5: $\mathbf{c} = [1, 5, 0.3, 1.2]$. Intuitively, we can label type 2-3 as "careful" agents and type 4-5 as "aggressive" agents. This is because type 2-3 followers are more sensitive to the guidance cost and have a wider sensing range. They tend to follow the leader more closely than type 4-5 followers.

We note from (2) that collecting the follower's best response data does not need the leader to take optimal actions. The leader can record the follower's response by providing a feasible state x and action u^L and prepare the dataset.

B. Meta-Learning Results

We use a neural network with two hidden layers of 50 ReLU nonlinearities to parameterize the follower's best response. We perform 5×10^5 iterations for meta-optimization. In each meta iteration, we sample 5 tasks ($|\mathcal{T}_{\text{batch}}| = 5$) and K = 100 for $\mathcal{D}_{\theta}^{\text{train}}$ and $\mathcal{D}_{\theta}^{\text{test}}$. The sample ratio $\kappa = 2$. In adaptation, we sample K' = 1000 data with the same κ .

Since meta-training aims to find an averaged initial model for fast adaptation, we implement another two modelaveraging approaches for comparison. We first train a model to average the output space (Output-Ave), i.e., training a bestresponse model with the same structure as meta-learning using the shuffled data of types of followers (average followers' behaviors). We also train a model to average the parameter space (Param-Ave), i.e., training $|\Theta|$ individual models for each type of follower and then averaging the model parameters using $p(\theta)$. Each individual model is trained by supervised learning and the type-specific dataset. These two comparative approaches are typical ways to generate averaged models used for adaptation. They have trained over 10^4 epochs with SGD with fine-tuned hyperparameters. Each epoch contains 150 iterations. All learning algorithms are performed on AMD Ryzen 3990X CPU. Tab. I summarizes the data usage and training time of different approaches. We manually implement the training of Output-Ave and Param-Ave by PyTorch but implement meta-learning algorithms, which explains the training time difference.

	Meta-learning	Output-Ave	Param-Ave	Adaptation	pro
Time	80.6 min	24.2 min	27.4 min (of 5)	5.3 s	4-5
Data	7.5×10^4 (of 5)	1.5×10^4	7.5×10^4 (of 5)	1000	

TABLE I: Summary of learning statistics. *Output-Ave* means averaging the output space; *Param-Ave* means averaging the parameter space. Training time for Param-Ave is averaged on separate models. The data for meta-learning and Param-Ave are summed over types.

We perform C = 50 steps of gradient adaptation for three approaches using the same sampled dataset with $\alpha = 10^{-4}$. For a detailed view, we plot the adaptation loss curve for type $\theta = 2$ follower in Fig. 2a and use the bar plot to show the adapted result (MSE loss) for all type θ in Fig. 2b.

The smaller adaptation error and the faster convergence rate indicate a better-generalized performance. As we observe in Fig. 2a, using the meta-learned model can fast reduce the adaptation error in the first few gradient steps compared with the Output-Ave approach, meaning that the meta-model can be adapted to the specific follower using fewer amounts of data and generating better performances. The meta-learned model performs best after C adaptation rounds, as shown in Fig. 2b. Conversely, the significant adaptation error produced by Param-Ave shows that averaging the model is not a practical option for predicting a new follower's behavior in real-world applications.

C. Receding Horizon Planning

After getting the adapted best-response model, the leader runs Alg. 3 to perform trajectory guidance. We simulate



(a) Adaptation loss curve for type 2 follower. A zoom-in box shows more details of meta adaptation and Output-Ave results.

(b) MSE errors after C = 50 gradient steps adaptation for all followers. Metalearning provides the best adaptation result.

Fig. 2: Adaptation results for three learning approaches. Meta-learning provides the best generalization adaptation performance. Param-Ave approach yields a significant adaptation error and poor generalization performance. We divide its loss by 2 in both plots for better visualization.

guided trajectories for each follower starting from different initial positions, shown in Fig. 3.

The leader successfully guides all types of followers to the destinations using corresponding adapted models, showing the effectiveness of the trajectory guidance. We can visualize different guidance plans that the leader uses for different followers. For example, we take the trajectory starting from [6, 0]. As mentioned, type 2-3 followers tend to follow the leader more closely than type 4-5 followers. Therefore, the leader makes more aggressive trajectories in Fig. 3b-3c to attract the follower and help adjust their initial heading directions. While in Fig. 3d-3e, the leader simply needs to provide a reference trajectory to the follower because type [4-5 followers rely less on the guidance.

We also observe that the leader's trajectory is zigzagged near obstacles and smooth in flat regions. It shows that the leader is aware of the follower's behavior near the obstacle and adjusts her action to better guide the follower. Some trajectories can be complicated, such as in Fig. 3a and Fig. 3e. This is mainly due to the learning accuracy of the model. However, the leader still manages to guide the follower passing the obstacle, showing that the model is also effectively learned and robust.

D. Comparison With Zero Guidance

To demonstrate the necessity of the leader's guidance, we simulate a zero guidance scenario where the follower seeks a myopic trajectory by himself, i.e., disregarding the guidance cost in J_{θ}^{F} . For simplicity, we plot the myopic trajectories for type 3 and type 5 followers starting from different positions, shown in Fig. 4. The myopic trajectories of type 1-3 followers are similar, while the ones of type 4-5 followers are similar. We also use color maps to plot the sensing cost for better visualization.

As we observe, two followers have trouble reaching their destination due to myopic planning. In fact, all followers fail to reach the destination from initial positions in the zero guidance scenario. They either get stuck on obstacles or run out of working space \mathcal{X} , similar to type 3 and type 5 followers in Fig. 4. The reason for sticking at obstacles is twofold. First, followers use unicycle models, which do not



Fig. 3: Guidance trajectories for different followers. The blue and the orange represent the leader and follower trajectories, respectively. Followers start from [0, 8] and [6, 0] to reach the goal region centered around [9, 9]. The leader successfully guides all followers to the destination using adapted best-response models and receding horizon planning algorithms.



Fig. 4: Myopic trajectories for two types of followers starting from [0, 8], [0, 4], and [6, 0]. None of them reach their destination. The color map represents the follower's sensing cost. We can see that type 3 follower has a wider sensing region than type 5 follower.

have the flexibility to move around as freely as the leader (a single integrator). The second reason is related to rectangle obstacles. The follower senses the homogeneous cost along the obstacle surface. Thus, any action will lead to the same obstacle cost unless the follower can cross the obstacle in one step. Besides, taking action can also incur a control cost. So the best option is to stay still. We observe that both type 3 and type 5 follower starting from position [0, 4] and the type 5 follower starting from [0, 8] face the same issue. The guidance from the leader can effectively steer the follower away from obstacles and reach the destination (see Fig. 3), showing the value of trajectory guidance.

V. CONCLUSION

In this work, we have proposed a Stackelberg metalearning approach to address guided trajectory planning problems with multiple follower robots with unknown decisionmaking models. Our approach not only provides a gametheoretic characterization of the leader-follower type of interactions in trajectory guidance tasks but also develops an effective learning mechanism to learn and fast adapt to different guidance tasks to assist followers in reaching their destination. Simulations have validated our approach in providing successful trajectory guidance and show better generalization and adaptation performance in learning followers' behavior models than other non-meta-learningbased approaches. Comparisons with zero guidance scenarios have demonstrated the value of guidance in assisting the follower robot to the destination. For future work, we would investigate the impact of the response model properties on the planning results, such as smoothness, in more general Stackelberg frameworks. We would also generalize our approach to other application domains, such as human-robot interactions.

APPENDIX

A. Discrete Dynamic Models

We discretize the continuous dynamical system using dt = 0.2. We write $x^L = [p_x^L, p_y^L]$ and $u^L = [v_x^L, v_y^L]$ as leader's state and control. The leader's discrete model is given by

$$x_{t+1}^{L} = x_{t}^{L} + u_{t}^{L} \cdot \mathrm{d}t.$$
 (12)

Likewise, we write $x^F = [p_x^F, p_x^F, \phi^F]$ and $u^F = [v^F, \omega^F]$ as follower's state and control. We use mixed discretization for the follower, which yields

$$x_{t+1}^{F} = x_{t}^{F} + \begin{bmatrix} v_{t}^{F} \cos(\phi_{t+1}^{F}) \\ v_{t}^{F} \sin(\phi_{t+1}^{F}) \\ \omega_{t}^{F} \end{bmatrix} dt.$$
(13)

The follower first adjusts the heading angle and then applies the linear velocity in the discrete dynamic model. Here we assume that all followers have independent dynamics and are unaffected by the leader, a special case of the general form in (4).

B. Leader and Follower Cost Functions

The leader's cost is the same for all $\theta \in \Theta$, which is given by

$$g_{\theta}(x, u^{L}, u^{F}) = \left\| x - x^{d} \right\|_{Q_{1}}^{2} + \left\| p^{L} - p^{F} \right\|_{Q_{2}}^{2} + \left\| u^{L} \right\|_{R}^{2},$$
$$q_{\theta}(x) = \left\| x - x^{d} \right\|_{Q_{f_{1}}}^{2} + \left\| p^{L} - p^{F} \right\|_{Q_{f_{2}}}^{2},$$

where $x^{d} = [p^{d}, p^{d}, 0] \in \mathbb{R}^{5}$ is the aggregated target state; $Q_{1}, Q_{2}, R, Q_{f1}, Q_{f2} \succeq 0$ are weighting parameters of appropriate dimensions. The term $||p^{L} - p^{F}||$ is interpreted as the guidance cost, producing a high value if the leader is far from the follower. In the simulation, we set $Q_{1} =$ $2 \times I_{5}, Q_{2} = 5 \times I_{5}, R = I_{2}$, the terminal cost parameters are set as $Q_{f1} = 5 \times Q_{1}, Q_{f2} = 5 \times Q_{2}$. To simulate the interactive data, we construct the follower's *ground truth* cost function using the following four base functions:

$$J_{\theta}^{F}(u^{F}; x, u^{L}) = c_{1} \left\| p_{+}^{F} - p^{d} \right\|_{2}^{2} + c_{2} \left\| p_{+}^{L} - p_{+}^{F} \right\|_{2}^{2} + c_{3} \left\| u^{F} \right\|_{2}^{2} + \sum_{j=1}^{M} h(c_{4}(\left\| \Lambda(p_{+}^{F} - p_{j}^{O}) \right\|_{l} - d_{j})),$$
(14)

where $h(x) = 10 \log(x)$ if $0 < x \le 1$ and 0 otherwise. The interpretation of the terms associated with $c_1 \cdot c_3$ is straightforward. c_4 relates the follower's sensing capabilities. A smaller c_4 indicates that the follower has a wider sensing range and can avoid obstacles earlier. $\Lambda \in \mathbb{R}^2$ is a scaling factor to define obstacles with different shapes. For example, we can choose a proper Λ to define elliptic obstacles if l = 2. Note that the follower takes the state x and the leader's action u^L as the parameter. His myopic decision is based on the one-step prediction p_+^L and p_+^F using the dynamics (12)-(13). We assume that different followers have the same cost function structure but different parameters $c_1 \cdot c_4$. We use (14) as the Oracle to generate the best-response data for learning.

REFERENCES

- D. Panagou, D. M. Stipanović, and P. G. Voulgaris, "Distributed coordination control for multi-robot networks using lyapunov-like barrier functions," *IEEE Transactions on Automatic Control*, vol. 61, no. 3, pp. 617–632, 2015.
- [2] M. Bibuli, M. Caccia, L. Lapierre, and G. Bruzzone, "Guidance of unmanned surface vehicles: Experiments in vehicle following," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 92–102, 2012.
- [3] S. Nikolaidis, S. Nath, A. D. Procaccia, and S. Srinivasa, "Gametheoretic modeling of human adaptation in human-robot collaboration," in *Proceedings of the 2017 ACM/IEEE international conference* on human-robot interaction, 2017, pp. 323–331.
- [4] H. Bo, D. M. Mohan, M. Azhar, K. Sreekanth, and D. Campolo, "Human-robot collaboration for tooling path guidance," in 2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob). IEEE, 2016, pp. 1340–1345.
- [5] Z. Wang and M. Schwager, "Kinematic multi-robot manipulation with no communication using force feedback," in 2016 IEEE international conference on robotics and automation (ICRA). IEEE, 2016, pp. 427–432.
- [6] T. Machado, T. Malheiro, S. Monteiro, W. Erlhagen, and E. Bicho, "Multi-constrained joint transportation tasks by teams of autonomous mobile robots using a dynamical systems approach," in 2016 IEEE international conference on robotics and automation (ICRA). IEEE, 2016, pp. 3111–3117.
- [7] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
- [8] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.
- [9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
- [10] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on control systems technology*, vol. 18, no. 2, pp. 267–278, 2009.
- [11] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.
- [12] P. Hang, C. Lv, Y. Xing, C. Huang, and Z. Hu, "Human-like decision making for autonomous driving: A noncooperative game theoretic approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2076–2087, 2020.

- [13] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Gametheoretic planning for self-driving cars in multivehicle competitive scenarios," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1313– 1325, 2021.
- [14] A. Turnwald and D. Wollherr, "Human-like motion planning based on game theoretic decision making," *International Journal of Social Robotics*, vol. 11, no. 1, pp. 151–170, 2019.
- [15] M. Zhu, M. Otte, P. Chaudhari, and E. Frazzoli, "Game theoretic controller synthesis for multi-robot motion planning part i: Trajectory based algorithms," in 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014, pp. 1646–1651.
- [16] M. Wang, N. Mehr, A. Gaidon, and M. Schwager, "Game-theoretic planning for risk-aware interactive agents," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 6998–7005.
- [17] T. Başar and G. J. Olsder, *Dynamic noncooperative game theory*. SIAM, 1998.
- [18] J. J. Koh, G. Ding, C. Heckman, L. Chen, and A. Roncone, "Cooperative control of mobile robots with stackelberg learning," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 7985–7992.
- [19] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Robotics: Science and systems*, vol. 2. Ann Arbor, MI, USA, 2016, pp. 1–9.
- [20] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," in 2019 International conference on robotics and automation (ICRA). IEEE, 2019, pp. 9590–9596.
- [21] Y. Zhao, B. Huang, J. Yu, and Q. Zhu, "Stackelberg strategic guidance for heterogeneous robots collaboration," in 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 4922–4928.
- [22] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.
- [23] H. Jia, Y. Zhao, Y. Zhai, B. Ding, H. Wang, and Q. Wu, "Crmrl: Collaborative relationship meta reinforcement learning for effectively adapting to type changes in multi-robotic system," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 362–11 369, 2022.
- [24] Y. Gao, E. Sibirtseva, G. Castellano, and D. Kragic, "Fast adaptation with meta-reinforcement learning for trust modelling in human-robot interaction," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 305–312.
- [25] D. Luipers and A. Richert, "Concept of an intuitive human-robotcollaboration via motion tracking and augmented reality," in 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA). IEEE, 2021, pp. 423–427.
- [26] S. Xu and M. Zhu, "Meta value learning for fast policy-centric optimal motion planning," in *Robotics science and systems*, 2022.
- [27] S. M. Richards, N. Azizan, J.-J. Slotine, and M. Pavone, "Controloriented meta-learning," arXiv preprint arXiv:2204.06716, 2022.
- [28] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus, "Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, 2008, pp. 895–902.
- [29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.