Maximum-Entropy Multi-Agent Dynamic Games: Forward and Inverse Solutions

Negar Mehr, *Member, IEEE*, Mingyu Wang, Maulik Bhatt, and Mac Schwager, *Member, IEEE*

***Abstract*—In this article, we study the problem of multiple stochastic agents interacting in a dynamic game scenario with continuous state and action spaces. We define a new notion of stochastic Nash equilibrium for boundedly rational agents, which we call the entropic cost equilibrium (ECE). We show that ECE is a natural extension to multiple agents of maximum entropy optimality for a single agent. We solve both the "forward" and "inverse" problems for the multi-agent ECE game. For the forward problem, we provide a Riccati algorithm to compute closed-form ECE feedback policies for the agents, which are exact in the linear-quadratic-gaussian case. We give an iterative variant to find locally ECE feedback policies for the nonlinear case. For the inverse problem, we present an algorithm to infer the cost functions of the multiple interacting agents given noisy, boundedly rational input and state trajectory examples from agents acting in an ECE. The effectiveness of our algorithms is demonstrated in a simulated multi-agent collision avoidance scenario, and with data from the INTERACTION traffic dataset. In both cases, we show that, by taking into account the agents' game theoretic interactions using our algorithm, a more accurate model of agents' costs can be learned, compared with standard inverse optimal control methods.**

***Index Terms*—Game-theoretic interactions, inverse reinforcement learning (IRL), learning from demonstration, multi-agent systems.**

I. Introduction

IN THIS article, we seek to learn the cost functions of a group of interacting dynamic agents from a set of trajectory demonstrations of those interactions. We call this problem an inverse dynamic game (IDG), analogously to inverse reinforcement learning (IRL) and Inverse optimal control (IOC) in the single-agent setting. To solve the inverse dynamic game, we first formulate a new notion of stochastic Nash equilibrium to describe the boundedly rational equilibrium condition found in natural human demonstrations. We call this equilibrium an entropic cost equilibrium (ECE). We then present an algorithm to find feedback policies for agents engaged in an ECE game, i.e., we solve the "forward" problem. We then use this forward solution within an algorithm to infer the cost functions of the multiple agents given trajectory demonstrations from those agents, i.e., we solve the "inverse" problem.

For many robotic applications, it is not trivial to design a cost function that mimics an expert's behavior, such as a human driver's actions. In such applications, a miss-specified cost function may lead to undesired behaviors and designing the correct cost function is notoriously challenging. A common practice is to infer the cost function from experts' demonstrations through the framework of IRL, also sometimes called IOC. An IRL algorithm infers the cost function by observing an agent's behavior assuming that the agent behaves approximately optimally. However, real-world robotic applications, such as autonomous driving, usually involve multiple interactive agents whose behaviors are coupled through the feedback interactions between the agents. Consequently, when learning from interactive agents' behaviors, we cannot treat them as acting in isolation. Instead, we need to take into account the game theoretic coupling between agent's behaviors. In this article, we develop inverse methods for such interactive multi-agent settings, where we learn each agent's cost function while considering their feedback interactions. We call this an inverse dynamic game (IDG).

One of the main challenges in solving IDGs is that agents no longer optimize their own cost functions in isolation. Rather, they reach a notion of game theoretic equilibrium. Hence, agents' cost functions must be learned such that the learned cost functions rationalize the set of demonstrations as game theoretic equilibrium strategies, rather than optimal strategies. Moreover, when learning from experts such as humans, we need to account for the humans' noisy behavior and bounded rationality. Reaching exact equilibria requires perfect rationality of agents; however, in decision-making settings, humans' rationality is normally bounded due to the limited information they have, their cognitive limitations and the finite amount of time available for decision making. Thus, we need to account for the noise in humans' decision making in a multi-agent setting.

Manuscript received 13 August 2022; accepted 22 November 2022. This work was supported in part by ONR Grant N00014-18-1-2830, Toyota Research Institute, and National Science Foundation, under Grant ECCS-2145134 CAREER Award, Grant CNS-2218759, and Grant CCF-2211542. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. This article was recommended for publication by Associate Editor P. Tokekar and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. *(Corresponding author: Negar Mehr.)*

Negar Mehr is with the Aerospace Engineering Department, University of Illinois Urbana-Champaign, Urbana, IL 61801 USA (e-mail: negar@illinois.edu).

Mingyu Wang is with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: wangmingyu1@gmail.com).

Maulik Bhatt is with the Department of Aerospace Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801 USA (e-mail: mcbhatt2@illinois.edu).

Mac Schwager is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA (e-mail: schwager@stanford.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2022.3232300.

Digital Object Identifier 10.1109/TRO.2022.3232300

In this article, we address these challenges by defining a notion of "noisy" equilibrium for capturing the outcome of interactions between multiple boundedly rational agents. We call this equilibrium an ECE and show its connections to the maximum entropy framework common in IRL [1]. We prove that the ECE concept is indeed an extension of maximum-entropy optimality to multi-agent settings. Once we formalize the notion of ECE, to assess the quality of a set of learned agents' cost functions, one must find the ECE policies under the learned costs and compare them to the set of agents' demonstrations. To enable such comparisons, we develop an algorithm to find ECE policies for a given set of agents' cost functions. We prove how ECE policies can be obtained in closed-form for a special class of games, namely linear-quadratic-Gaussian games, using a Riccati solution reminiscent of the well-known linear-quadratic regulator (LQR) and linear-quadratic game (LQGame) solutions. Leveraging this result, we provide an iterative algorithm for approximating ECE policies in general multi-agent games with general nonlinear dynamics and costs.

Knowing how to approximate ECE policies for a set of given costs, we propose a multi-agent inverse dynamic game algorithm for learning agents' costs. Similar to the common practice in IRL [1], [2], [3], [4], we assume that each agent's cost function is parameterized as a linear combination of a set of known features. We then propose an iterative algorithm for learning the weights of the features in each agent's cost function such that the feature expectations under the learned costs match the empirical feature average from the demonstrations. We verify our algorithm using both synthetic datasets and real-world traffic data. First, we consider a goal-reaching and collision-avoidance scenario involving two and three agents. We show that by taking into account the agents' feedback interactions, a more accurate model of agents' costs can be learned. We then validate the performance of our algorithm on the INTERACION dataset [5] which involves highly interactive multi-agent driving scenarios collected from different countries. We again demonstrate that by taking into account the agents' interactions through our inverse dynamic game framework, more accurate predictions of agents' behavior can be made. We show that the prediction accuracy of our framework is very close to the intelligent driver's model (IDM) [6], which is a highly accurate human-designed driving model used for modeling human drivers' leader-following behavior.

We summarize our contributions as follows.

1) We define and formalize the notion of ECE for capturing the interaction of noisy agents.
2) We develop an algorithm for computing approximate ECE policies for general nonlinear multi-agent systems (the forward problem).
3) We propose an iterative algorithm for learning the agents' costs from a set of interactive demonstrations (the inverse problem).
4) We validate our proposed IDG algorithm in both synthetic and real-world driving scenarios.

The rest of this article is organized as follows. Section II provides an overview of the related work. In Section III, we introduce our notation and discuss the preliminaries. In Section IV, we define the notion of ECE and discuss its connection to the maximum-entropy framework. We discuss finding ECE policies for general nonlinear multi-agent systems in Section V. Section VI provides the description of our inverse dynamic game algorithm. Simulations and experiments with the analysis of the resulting performance are incorporated in Section VII. Finally, Section VIII concludes this article with a discussion on future directions.

II. Related Work

A. Single-Agent IRL

Inferring and learning cost functions from system trajectories has been widely studied for single-agent systems. The problem of inferring an agent's cost function was first studied by Kalman in the context of inverse optimal control for linear quadratic systems with a linear control law [7]. Learning an agent's cost was later studied in [8] and [9] where the assumption was that the demonstrations satisfy optimality conditions. This assumption was relaxed to take into account the bounded rationality and human's noisy demonstrations in the framework of maximum entropy IRL in [1].

Despite the success of these IRL methods, they were mostly developed for discrete state and action spaces. In [3], maximum-entropy cost inference was studied for systems with continuous state and action spaces. The common assumption in these works is that the agent's cost function is parameterized as a weighted sum of a set of known features. This assumption was further relaxed in [10], where the underlying cost function was learned as a neural network in a maximum-entropy framework. In [11], a framework was proposed for directly extracting a policy from an agent's demonstrations, as if it were obtained by reinforcement learning following maximum entropy IRL. Inferring an agent's cost function was also studied as a bilevel optimization where, in the outer loop, the cost parameters were found such that the estimation error of system trajectories or the likelihood of demonstrations was optimized [12], [13]. Recent works have proposed learning cost parameters that minimize the residual of Karush–Kuhn–Tucker (KKT) conditions at the demonstrations [14], [15], [16]. Our work draws inspiration from the maximum entropy IRL framework where the noisy behavior of the agent is captured.

B. Multi-Agent IRL

Extension of single-agent IRL algorithms to the multi-agent settings have been studied in the past. Multi-agent IRL was studied for discrete state and action spaces in [17]. The inverse equilibrium problem was further considered in [18] where the maximum entropy principle and the notion of regret were employed for solving the inverse equilibrium problem. However, no systems dynamics were considered in [18], and the inverse equilibrium problem was considered in matrix games. IRL was considered in [19] for two-person zero-sum stochastic games with discrete state and action spaces where the cost learning problem was formulated as an optimization problem.

This framework was later extended to general-sum games with discrete state and action spaces in [20]. Modeling agents with different hierarchical levels of rationality was considered [21]. In [22], a new framework for multi-agent IRL utilizing adversarial machine learning was proposed for high-dimensional state and action spaces with unknown dynamics. In [23], the generative adversarial imitation learning of [11] in the single agent case was extended to the multi-agent setting. The current work is distinct in that it focuses on multi-agent IRL in general-sum games with *known* system dynamics and continuous state and action spaces. In [24], entropy regularized games were considered for two interacting agents. Entropy regularization was further considered for zero-sum games in [25]. However, both [24] and [25] were developed for finite state and action spaces and are not directly applicable to systems with continuous state and actions spaces.

Multi-agent IRL has been studied as an estimation problem too. In [26], a particle filtering algorithm was utilized for online estimation of human behavior parameters where the critical role of accurate human motion prediction was demonstrated. In [27], a filtering technique based on an unscented Kalman filter was developed for online estimation of cost parameters in multi-agent settings. In [28], IRL was considered for the class of linear quadratic games where the equilibrium strategies of all but one agent were known. This assumption simplified the problem and reduced it to effectively an instance of a single-agent cost inference problem. This framework was extended in [29] by proposing to minimize the residuals of the first-order necessary conditions for open-loop Nash equilibria. In [30], residual errors of optimality conditions for open-loop Nash equilibria were minimized in a maximum-entropy framework. In [31], state and input constraints were also considered in a maximum-entropy residual-minimization framework. In [32], agents' cost functions were estimated under partial observability.

In this work, we focus on cost learning for general *nonlinear* games, and find cost parameters which rationalize interactions under *feedback* information structure. We further capture the *noisy* behavior of humans with our ECE concept. In [30], maximum-entropy multi-agent inverse reinforcement learning (MA-IRL) was considered as a maximum likelihood problem. However, when computing the probability distribution over agents' trajectories, the coupling between the agents was not considered. Moreover, in this work, it was assumed that agents' feedback policies were known a-priori which is a restrictive assumption. In this article, we show how the maximum-entropy framework can be formalized in a multi-agent game theoretic setting to account for agents' feedback interactions. We further provide an algorithm which does not require prior knowledge of the agents' policies and verify it on a real-world dataset. We assume that we know the system dynamics, and we can model the agents' cost functions as a linear combination of a set of known features. These assumptions result in better data efficiency compared to methods that utilize general function approximators, such as neural networks, but require us to have access to a set of hand-tuned features that make sense for the domain in hand.

III. Preliminaries

Consider $N \geq 1$ agents interacting in an environment. We use $[N] = \{1, \ldots, N\}$ to refer to the set of all agents' indices. Let $s_t \in \mathcal{S}$ denote the vector of joint states of all agents at each time t, where the set $\mathcal{S} \subseteq \mathbb{R}^n$ is the joint state space observed by all agents, and n is the dimension of the state space. Each agent $i \in [N]$ decides on its action $a_t^i \in \mathcal{A}^i$ at time step t, where $\mathcal{A}^i \subseteq \mathbb{R}^{m^i}$ is the action space of agent i, and m^i is the dimension of the action space of agent i. Throughout this article, we use bold letters to refer to the concatenation of variables for all agents. For a given time step t, we use $\boldsymbol{a}_t = (a_t^1, \ldots, a_t^N)$ to denote the vector of all agents' actions at time t. Following the conventional notation used in the game theory literature, we utilize the superscript $-i$ to indicate all agents expect agent i. For example, $\boldsymbol{a}_t^{-i}$ represents the vector of all agents' actions excluding the action of agent i at time t. We define $\boldsymbol{\mathcal{A}} = \{\mathcal{A}^i\}_{i=1}^N$ to denote the collection of the action spaces of all the agents.

We assume that agents choose their actions through a stochastic Markovian policy. For each agent i, we use $\pi_t^i(.|s_t)$ to denote such a policy for agent i where $\pi_t^i(a_t^i|s_t)$ encodes the probability of agent i taking the action a_t^i at time t given that the system is in state s_t. Note that we are assuming that each agent selects its action independently, i.e., $\pi_t^i(a_t^i|s_t, \boldsymbol{a}_t^{-i}) = \pi_t^i(a_t^i|s_t)$. For a finite horizon T, we use $\pi^i = \{\pi_t^i\}_{t=1}^T$ to refer to the agent i's policy for the entire horizon T. Moreover, we use $\boldsymbol{\pi} = \{\pi^i\}_{i=1}^N$ to refer to the set of all agents' time-dependent policies. The discrete-time dynamics of the system are represented by state updates of the following form:

$$\begin{aligned} s_{t+1} &= f(t, s_t, \boldsymbol{a}_t) + g(s_t) w_t \\ s_1 &\sim p(s_1), \quad w_t \sim p_w \end{aligned} \tag{1}$$

where p_1 and p_w are the distributions of the system initial state and system noise, respectively, f captures the impact of the current state and action on the future state, and g is the state-dependent multiplier of the process noise. We assume that each agent i has a bounded per-stage cost function $c^i : \mathcal{S} \times \mathcal{A}^1 \cdots \times \mathcal{A}^N \to \mathbb{R}$. We further let $\boldsymbol{c} = \{c^i\}_{i=1}^N$ represent the vector of all agents' per-stage costs. We assume that each agent i is seeking to minimize its own expected cumulative cost $\mathbb{E}_{\boldsymbol{\pi}} \sum_{t=1}^T c^i(s_t, \boldsymbol{a}_t)$.

We model the agents' interaction as a dynamic game. We use the notation $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$ to refer to a game with a time horizon of length T among N agents whose action spaces and costs are defined via $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{c}$ on the state space $\mathcal{S}$ with the prespecified dynamics (1).

It is well known that the outcome of interaction between *perfectly* rational agents is best represented via Nash equilibria of the underlying game. Given a game $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$, a set of (mixed strategy) Nash equilibrium policies are defined through the following.

Definition 1: Given a game $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$, a set of agents' policies $\boldsymbol{\pi}^*$ is a (mixed-strategy) Nash equilibrium if and only if for each agent $i \in [N]$, we have

$$\mathbb{E}_{\pi^{i*},\boldsymbol{\pi}^{-i*}} \sum_{k=1}^{T} c^i(s_k, a_k^i, \boldsymbol{a}_k^{-i}) \le \mathbb{E}_{\pi^{i},\boldsymbol{\pi}^{-i*}} \sum_{k=1}^{T} c^i(s_k, a_k^i, \boldsymbol{a}_k^{-i}), \ \forall \pi^i. \tag{2}$$

Definition 1 implies that, at a Nash equilibrium, no agent will reduce its accumulated cost by unilaterally changing its policy from the equilibrium policy π^{i*} to another policy π^i.

Although Nash equilibrium is a powerful concept for modeling the interaction of agents, achieving Nash equilibria requires perfect rationality of agents. Moreover, it is well known that computing Nash equilibria is in general intractable even in normal-form games [33]. Thus, when learning from a set of demonstrations that are collected from experts such as humans, assuming that humans have achieved a Nash equilibrium may be unreasonable. Not only are humans computationally bounded, but they also may act under noisy information, or produce actions that are different than what they intend, making them appear to act irrationally to some degree. This concept is known in game theory as *bounded rationality*. Instead of making the "best" choices, humans often make choices that are "satisfactory on average" [34], [35]. In the next section, we generalize the notion of Nash equilibrium to capture the bounded rationality of experts such as humans during their interactions.

IV. Entropic Cost Equilibrium

In the seminal work of [36] and [37], it was shown that to take into account the bounded rationality of humans, their interaction can be modeled via the notion of quantal response equilibrium in normal form and extended games. It was demonstrated that the quantal response equilibrium can successfully model humans' choices in a set of lab experiments, while the predictions made by the Nash equilibrium deviated largely from the lab experiments. At quantal response equilibrium, every agent maintains a probability distribution over its actions. At quantal response equilibrium, the noisy behavior of humans is captured, where the probability of an action taken by a human is related to the cost associated to that action for the human. In [38], a continuous version of the notion of quantal response equilibrium, called logit equilibrium, was developed for repeated continuous games from the perspective of evolutionary game theory. In this setup, at logit equilibrium, every agent computes its expected cost with respect to the probability distribution over actions of all agents. Each agent takes actions that are exponentially proportional to the negative of this expectation. In the following, we extend this notion of logit equilibrium to dynamic games with continuous state and action spaces. We use the multi-agent extension of Q functions and prove certain properties of this notion of equilibrium.

Given a game $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$, at every time step $t < T$, for each agent $i \in [N]$, we define the quality of a state s_t and a vector of agents' actions $\boldsymbol{a}_t$ under a given set of agents' policies $\boldsymbol{\pi}$ via the following:

$$Q^i_{t,\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t) = c^i(s_t, \boldsymbol{a_t}) + \mathbb{E}_{s_{t+1:T},\boldsymbol{\pi}} \sum_{k=t+1}^{T} \left[c^i(s_k, \boldsymbol{a}_k) | s_t, \boldsymbol{a}_t\right]. \tag{3}$$

For the final time step T, the cost associated with a state s_T and a vector of actions $\boldsymbol{a}_T$ for an agent i is

$$Q^i_{T,\boldsymbol{\pi}}(s_T, \boldsymbol{a}_T) = c^i(s_T, \boldsymbol{a}_T). \tag{4}$$

Equations (3) and (4) are the extensions of the definition of the Q function to multi-agent game theoretic settings. Following [38], for each agent $i \in [N]$ and every time step $t \le T$, we define

$$\bar{Q}^i_{t,\boldsymbol{\pi}}(s_t, a_t^i) = \mathbb{E}_{\boldsymbol{a}^{-i} \sim \boldsymbol{\pi}^{-i}} Q^i_{t,\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t). \tag{5}$$

In (5), the expectation of the Q function is computed with respect to the actions of all the other agents $\boldsymbol{a}^{-i}$.

Note that $\bar{Q}^i_{\boldsymbol{\pi}}(s_t, a_t^i)$ depends only on the action of agent i and the state, not the actions of the other agents. In fact, $\bar{Q}^i_{\boldsymbol{\pi}}(s_t, a_t^i)$ determines the quality of a pair of the system state and an agent's action given the set of other agents' policies $\boldsymbol{\pi}^{-\boldsymbol{i}}$. Ideally, if agents were perfectly rational, given knowledge of the other agents' policies, at every time step, each agent would have taken actions that minimize $\bar{Q}^i_{\boldsymbol{\pi}}(s_t, a_t^i)$ at equilibrium. However, when agents are boundedly rational, we propose that agents' noisy behavior can be modeled through the following notion of equilibrium.

Definition 2: For a given game $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$, a set of agents' policies $\boldsymbol{\pi}^* = \{\pi^{i*}\}_{i=1}^N$ is an ECE if and only if for every agent $i \in [N]$ and every action a_t^i, the following holds at every time step $t \le T$:

$$\pi_t^{i*}(a_t^i | s_t) = \frac{e^{-\bar{Q}^i_{t,\boldsymbol{\pi}^*}(s_t, a_t^i)}}{\int e^{-\bar{Q}^i_{t,\boldsymbol{\pi}^*}(s_t, \tilde{a}_t^i)} d\tilde{a}_t^i}. \tag{6}$$

We require the above conditions to hold starting from any time step $1 \le t \le T$ using the Q function that captures the optimal cost to go from time t to T, i.e., this notion of equilibrium is a subgame perfect equilibrium. Note that (6) must hold for all agents. An equilibrium policy $\boldsymbol{\pi}^*$ is indeed the fixed point of (6), and $\bar{Q}^i_{\boldsymbol{\pi}^*}(s_t, a_t^i)$ depends on the policy of all agents. It is important to note that in ECE, at every time step t, the agents' actions a_t^i are independent. No agent needs to know the action choice of any other agent to choose its own action. However, the probability of taking an action is implicitly dependent on the *policies* of other agents $\boldsymbol{\pi}^{-i^*}$ through the expectation with respect to the agents' policies in $\bar{Q}^i_{\boldsymbol{\pi}^*}(s_t, a_t^i)$. In other words, at every time step, agents' *actions*, i.e., the particular value of the control input that they choose are independent, but the agent's *policies* are not independent and they should reach an equilibrium in the sense of Definition 2. This implies that the probability of taking an action by one agent is implicitly dependent on the policies of other agents. In fact, what our notion of equilibrium captures is that the strategies (policies) of agents are dependent upon one another but the probability of taking an action is independent of the particular action that another agent has chosen in a given state. Therefore, although the agents' instantaneous actions are independent, their policies, i.e., the probability distribution over their actions, are related in ECE through (6), and the ECE policies $\boldsymbol{\pi}^*$ are indeed the fixed points of (6) for all agents. This is in contrast to the definition of equilibrium in [22] where a stochastic version of correlated equilibrium was developed, and the actions were assumed to be correlated.

We would like to highlight the connection between (6) and the maximum entropy framework. If there exists only one single agent in the environment, (6) reverts to the well-known maximum entropy formulation, which is widely used in the development of both reinforcement learning and IRL algorithms [3], [39], [40], [41], [42]. In fact our notion of entropic cost equilibrium can be viewed as an extension of energy-based policies to multi-agent settings. If there is only one agent, the probability of taking an action a_t given a state s_t is proportional to the exponential of the negative accumulated cost from that state (s_t, a_t). Now that we have defined ECE as our equilibrium notion for capturing humans' noisy interactions, we will prove a property of ECE which demonstrates its applicability and relevance to modeling the interaction of multiple agents with bounded rationality.

Theorem 1: For a game $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$, a set of agents' policies $\boldsymbol{\pi}^* = \{\pi^{i*}\}_{i=1}^N$ is an ECE if and only if $\boldsymbol{\pi}^*$ is a (mixed-strategy) feedback Nash equilibrium for the maximum entropy game $\tilde{G} = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \tilde{\boldsymbol{c}}, T)$ where $\tilde{\boldsymbol{c}} = \{\tilde{c}^i\}_{i=1}^N$ is defined as

$$\tilde{c}^i(s_t, \boldsymbol{a}_t) = c^i(s_t, \boldsymbol{a}_t) - \mathcal{H}(\pi^i(\cdot|s_t)) \tag{7}$$

where $\mathcal{H}(\pi_t^i(\cdot|s_t))$ is the entropy of policy $\pi^i(\cdot|s_t)$.

Proof: First, we show that if $\boldsymbol{\pi}^*$ is a feedback Nash equilibrium for the game $\tilde{G}$, then, $\boldsymbol{\pi}^*$ is an ECE for the game G. Let $\boldsymbol{\pi}^*$ be a feedback Nash equilibrium of game $\tilde{G}$. Fix an agent $i \in [N]$ and the policy of all the other agents $\boldsymbol{\pi}^{-i*}$. Then, at Nash equilibrium of game $\tilde{G}$, the agent's policy π^{i*} optimizes the following:

$$\min_{\pi^i} \mathbb{E} \sum_{k=1}^{T} \left(c^i(s_k, a_k^i, \boldsymbol{a}_k^{-i*}) - \mathcal{H}(\pi_k^i(\cdot|s_k)) \right) \tag{8}$$

where the expectation is with respect to $\boldsymbol{a}_k^{-i*} \sim \boldsymbol{\pi}_k^{-i*}$, and $a_k^i \sim \pi_k^i$. We can solve the above for finding Nash equilibrium policies π^{i*} using dynamic programming. Starting from the final time step T, for each state s_T, we have

$$\min_{\pi_T^i} \mathbb{E} \left(c^i(s_T, a_T^i, \boldsymbol{a}_T^{-i*}) - \mathcal{H}(\pi_T^i(\cdot|s_T)) \right) \tag{9}$$

where the expectation is with respect to $\boldsymbol{a}_T^{-i*} \sim \boldsymbol{\pi}_T^{-i*}$, and $a_T^i \sim \pi_T^i$. Equation (9) can be rewritten as

$$\min_{\pi_T^i} \mathbb{E} \left(\mathbb{E} \left[c^i(s_T, a_T^i, \boldsymbol{a}_T^{-i*}) \right] - \mathcal{H}(\pi_T^i(\cdot|s_T)) \right) \tag{10}$$

where the inner expectation is with respect to $\boldsymbol{a}_T^{-i*} \sim \boldsymbol{\pi}_T^{-i*}$, and the outer expectation is with respect to $a_T^i \sim \pi_T^i$. Define the following:

$$\bar{c}^i(s_T, a_T^i) = \mathbb{E}_{\boldsymbol{a}^{-i*}{}_T \sim \boldsymbol{\pi}_T^{-i*}} \left(c^i(s_T, a_T^i, \boldsymbol{a}_T^{-i*}) \right). \tag{11}$$

Then, (10) can be rewritten as minimizing the following:

$$\begin{aligned} &\mathbb{E}_{\pi_T^i} \left[\bar{c}^i(s_T, a_T^i) - \mathcal{H}(\pi_T^i(\cdot|s_T)) \right] \\ &\quad = \mathbb{E}_{\pi_T^i} \left[\bar{c}^i(s_T, a_T^i) + \log(\pi_T^i(\cdot|s_T)).\right] \end{aligned} \tag{12}$$

Now, following the results in [43], (12) can be rewritten as

$$\begin{aligned} D_{\mathrm{KL}} &\left(\pi_T^i(.|s_T) \| \frac{1}{\exp\left(V_T^i(s_T)\right)} \exp\left(-\bar{c}^i(s_T, a_T^i)\right) \right) \\ &\qquad + V_T^i(s_T) \end{aligned} \tag{13}$$

where D_{KL} denotes the Kullback-Leibler (KL) divergence, and $V_T^i(s_T)$ is defined as

$$V_T^i(s_T) = \log \int e^{-\bar{c}^i(s_T, \tilde{a}_T^i)} d\tilde{a}_T^i. \tag{14}$$

Now, to minimize (13), note that $\exp(V_T^i(s_T))$ is a constant for a given state s_T. Since the KL divergence is minimum when the two arguments are the same, the policy which minimizes (13) is

$$\pi_T^{i*}(a_T^i|s_T) = \frac{e^{-\bar{c}^i(s_T, a_T^i)}}{e^{V_T^i(s_T)}}. \tag{15}$$

For every state s_T, $V^i(s_T)$ is indeed the soft cost-to-go (or value function) of agent i at time T. We can use dynamic programming to propagate this cost-to-go backwards in time and find the equilibrium policy for agent i at time $T-1$. At time $T-1$, we need to solve for

$$\begin{aligned} \min_{\pi_{T-1}^i} \mathbb{E} &\left[c^i(s_{T-1}, a_{T-1}^i, \boldsymbol{a}_{T-1}^{-i*}) \right. \\ &\left. - \mathcal{H}(\pi_{T-1}^i(\cdot|s_{T-1})) + \mathbb{E}_{s_T}(V_T^i(s_T)) \right] \end{aligned} \tag{16}$$

where the first expectation is with respect to $a_{T-1}^i \sim \pi_{T-1}^i$ and $\boldsymbol{a}_{T-1}^{-i*} \sim \boldsymbol{\pi}_{T-1}^{-i*}$. Again, we can separate the expectations in (16) as

$$\begin{aligned} \min_{\pi_{T-1}^i} \mathbb{E}_{\pi_{T-1}^i} &\left(\mathbb{E}_{\boldsymbol{\pi}_{T-1}^{-i*}} \left[c^i(s_{T-1}, a_{T-1}^i, \boldsymbol{a}_{T-1}^{-i*}) + \mathbb{E}_{s_T} V_T^i(s_T) \right] \right. \\ &\left. - \mathcal{H}(\pi_{T-1}^i(\cdot|s_{T-1})) \right). \end{aligned} \tag{17}$$

Then, using (5) we see that (17) can be rewritten as

$$\min_{\pi_{T-1}^i} \mathbb{E}_{\pi_{T-1}^i} \left[\bar{Q}_{T-1}^i(s_{T-1}, a_{T-1}^i) - \mathcal{H}(\pi_{T-1}^i(\cdot|s_{T-1})) \right]. \tag{18}$$

Thus, similar to minimizing (12), to minimize (18), the equilibrium policy for agent i at time step $T-1$ is

$$\pi_{T-1}^{i*}(a_{T-1}^i|s_{T-1}) = \frac{e^{-\bar{Q}_{T-1}^i(s_{T-1}, a_{T-1}^i)}}{e^{V_{T-1}^i(s_{T-1})}} \tag{19}$$

where the cost-to-go $V_{T-1}^i(s_{T-1})$ is defined as

$$V_{T-1}^i(s_{T-1}) = \log \int e^{-\bar{Q}_{T-1}^i(s_{T-1}, \tilde{a}_{T-1}^i)} d\tilde{a}_{T-1}^i.$$

Repeating this procedure, we can solve for the mixed-strategy Nash equilibrium policies backwards in time. Thus, we show via induction that mixed-strategy Nash equilibrium policies of the game $\tilde{G} = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \tilde{\boldsymbol{c}}, T)$ are in fact the ECE policies of the original game $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$. To prove the reverse, i.e., every ECE of the game G is a mixed strategy Nash equilibrium for $\tilde{G} = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \tilde{\boldsymbol{c}}, T)$, we follow the same reasoning. Starting from the final time step, one can show that ECE policies optimize the cost-to-go in the game $\tilde{G}$, and use induction to prove that ECE policies of G are indeed Nash equilibria of $\tilde{G}$. ■

Theorem 1 connects ECE to the Nash equilibria of a game among agents who aim to maximize the entropy of their policy while minimizing their accumulated cost. The entropic game can be viewed as an auxiliary game to compute the ECE equilibria. Intuitively, it models the equilibrium that will be attained if exploratory agents (agents who want to have a policy that has a nonzero entropy) interact with each other. This in fact has a connection to the notion of bounded rationality. If agents were perfectly rational, they should have always played the Nash equilibrium strategy of game G with probability 1, i.e., the entropy of their policy must have been zero. But in the game $\tilde{G}$ that agents are exploratory, they play strategies which have a nonzero entropy, i.e., they exhibit a noisy behavior around the exact Nash equilibria of the game G.

ECE is in fact an extension of the maximum entropy-framework in the single-agent setting to the setting of multiple interactive agents. It is well known that when it comes to learning from demonstrations in the single-agent scenarios, to capture the bounded rationality and noisiness of the demonstrator, the demonstrator is best modeled via the maximum-entropy framework [1], [2]. Theorem 1 extends this notion to multi-agent games. We will use Theorem 1 in the remainder of this article for computing the ECE policies in general dynamic games.

Remark 1: The notion of ECE can incorporate a temperature parameter γ^i for each agent such that $p(a^i|s_t) \propto \exp\left(\frac{-\bar{Q}^i_{\boldsymbol{\pi}}(s_t,a^i_t)}{\gamma^i}\right)$. This in turn will lead to an additional weight γ^i on the entropy of each agent's policy in (7). The temperature weight γ^i captures each agent's rationality. The higher γ^i is, the noisier the agent acts. If for each agent $i \in [N]$, $\gamma^i \to \infty$, then in the limit, all agents act completely randomly with a uniform probability distribution over actions. On the other hand, when $\gamma^i \to 0$ for all agents, the Nash equilibrium policies are recovered. Therefore, the temperature coefficient γ^i reflects the rationality of each agent. For simplicity, in this article, we assume that $\gamma^i = 1$ for all agents.

V. Finding ECE Policies

So far, we have defined ECE for capturing the interaction of boundedly rational and noisy agents. In this section, we give an algorithm for computing ECE policies for a given game $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$ using Theorem 1. To solve the maximum entropy dynamic game, first, we prove that ECE policies can be computed in closed form for the class of linear quadratic Gaussian games using a Riccati solution similar to the classic LQR [7] and LQGames [44] solutions. Then, we extend this algorithm to general nonlinear dynamic games through iterative linear-quadratic approximations, similar to differential dynamic programming [45], iterative linear quadratic regulator (iLQR) [46], and iterative linear quadratic games (iLQGames) [47].

A. Quadratic Gaussian Games

Consider a class of games $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$ where the system dynamics are linear, and the system noise is normally distributed, i.e., the state update (1) is of the following form:

$$s_{t+1} = As_t + \sum_{j\in[N]} B^j a^j_t + w_t$$
$$s_1 \sim \mathcal{N}(\hat{s}_1, \Sigma_1), \quad w_t \sim \mathcal{N}(0, I) \tag{20}$$

where $A \in \mathbb{R}^{n\times n}$ and $B^j \in \mathbb{R}^{n\times m^j}$ are known time-invariant matrices of appropriate dimensions, the initial state s_1 is normally distributed with known mean and covariance $\hat{s}_1$ and Σ_1, and w_t is a zero-mean normally distributed random variable with covariance matrix being identity. Note that in general, the dynamics matrices can be time-variant, and the process noise can be any normal distribution. Here, for the ease of description, we assume that the system dynamics are linear time-invariant subject to process noise with covariance matrix being identity. Moreover, consider the class of cost functions where at every time step $t \leq T$, every agent $i \in [N]$ minimizes a convex quadratic cost function

$$c^i(s_t, a^1_t, \ldots, a^N_t) = \frac{1}{2}\Bigg(s_t^\top Q^i s_t + {l^i}^\top s_t$$
$$+ \sum_{j\in[N]} {a^j_t}^\top R^{ij} a^j_t\Bigg) \tag{21}$$

where $Q^i \in \mathbb{R}^{n\times n}$ is a positive semidefinite matrix, and l^i is a vector capturing the affine penalty of agent i for the system state. Moreover, for every agent $j \in [N], j \neq i$, $R^{ij} \in \mathbb{R}^{m^j\times m^j}$ is a positive semidefinite matrix while R^{ii} is a positive definite matrix. Equations (20) and (21) define the class of linear quadratic games where the dynamics are linear and the stage costs are quadratic in the states and actions.

Definition 3: A given game $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$ is a linear quadratic Gaussian game if its dynamics are of the form (20), and further, for each agent $i \in [N]$, the cost function c^i is of the form (21).

Note that for a linear quadratic Gaussian game G, in the resulting maximum entropy game $\tilde{G}$, every agent $i \in [N]$ minimizes the following accumulated cost when we fix the policy of all the other agents to be $\boldsymbol{\pi}^{-i}$:

$$\min_{\pi^i} \mathbb{E}_{\pi^i}\mathbb{E}_{\boldsymbol{\pi}^{-i}} \sum_{k=1}^{T} \frac{1}{2}\Bigg(s_k^\top Q^i s_k + {l^i}^\top s_k$$
$$+ \sum_{j\in[N]} {a^j_k}^\top R^{ij} a^j_k\Bigg) - \sum_{k=1}^{T} \mathcal{H}(\pi^i(.|s_k)). \tag{22}$$

Now that we have defined the class of linear-quadratic games, using (22) and Theorem 1, we will prove how ECE policies can be obtained in closed form for this class of games.

Theorem 2: Consider a linear quadratic Gaussian game $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$. For every agent $i \in [N]$, at every time step $t < T$, the ECE policy π^{i*}_t is a normal distribution $\pi^{i*}_t \sim$

$\mathcal{N}(\mu_t^i, \Sigma_t^i)$ where the mean μ_t^i and the covariance Σ_t^i are computed by

$$\mu_t^i = -P_t^i s_t - \alpha_t^i \tag{23}$$

$$\Sigma_t^i = (R^{ii} + {B^i}^\mathsf{T} Z_{t+1}^i B^i)^{-1}. \tag{24}$$

The matrices P_t^i and the vectors α_t^i satisfy the following sets of linear equations:

$$\left[R^{ii} + {B^i}^\mathsf{T} Z_{t+1}^i B^i\right] P_t^i + {B^i}^\mathsf{T} Z_{t+1}^i \sum_{j \in [N], j \neq i} B^j P_t^j = {B^i}^\mathsf{T} Z_{t+1}^i A \tag{25}$$

$$\left[R^{ii} + {B^i}^\mathsf{T} Z_{t+1}^i B^i\right] \alpha_t^i + {B^i}^\mathsf{T} Z_{t+1}^i \sum_{j \in [N], j \neq i} B^j \alpha_t^j = {B^i}^\mathsf{T} \xi_{t+1}^i \tag{26}$$

where Z_t^i and ξ_t^i are recursively computed from the following:

$$Z_t^i = F_t^\mathsf{T} Z_{t+1}^i F_t + \sum_{j \in [N]} {P_t^j}^\mathsf{T} R^{ij} P_t^j + Q^i \tag{27}$$

$$\xi_t^i = F_t^\mathsf{T}(\xi_{t+1}^i + Z_{t+1}^i \beta_t) + \sum_{j \in [N]} {P_t^j}^\mathsf{T} R^{ij} \alpha_t^j \tag{28}$$

where

$$F_t = A_t - \sum_{j \in [N]} B^j P_t^j \tag{29}$$

$$\beta_t = -\sum_{j \in [N]} B^j \alpha_t^j. \tag{30}$$

The terminal conditions for (27) and (28) are

$$\xi_T^i = l^i, \quad Z_T = Q^i. \tag{31}$$

Proof: The proof can be found in Appendix (A). ■

Note that for the linear quadratic Gaussian case, the closed-form policies are subgame perfect equilibria as the policies are indeed found by dynamic programming backwards in time. Theorem 2 is in fact an extension of maximum entropy LQR derived in [2] to the multi-agent setting. Equations (25), (26), (27), and (28) are the extensions of the Riccati backward recursion to the multi-agent setup. Indeed, these equations are similar to the backward recursions for deterministic LQGames derived in [44]. Theorem 2 suggests that for maximum entropy LQGames, the optimal policy is obtained by a normal distribution whose mean is found by solving LQgames. Then, the variance of the normal policies at every time step is found via (24).

Remark 2: In Equation (22), if we have a weight γ_i on the entropy of agent i's policy, then only the variance of the closed-loop policies will change. More precisely, for every agent $i \in [N]$, at every time step $t < T$, the ECE policy $\pi_t^{i^*}$ is a normal distribution $\pi_t^{i*} \sim \mathcal{N}(\mu_t^i, \Sigma_t^i)$ where the mean μ_t^i and the covariance Σ_t^i are computed by

$$\mu_t^i = -P_t^i s_t - \alpha_t^i \tag{32}$$

$$\Sigma_t^i = \gamma_i (R^{ii} + {B^i}^\mathsf{T} Z_{t+1}^i B^i)^{-1} \tag{33}$$

where matrices P_t^i and the vectors α_t^i are found similar to Theorem 2. Note than when $\gamma_i = 0$, the variance of the distribution shrinks to zero and the Nash equilibria of the underlying LQGame are recovered. In the limit, when $\gamma \to \infty$, the variance of the distribution goes to infinity. See Fig. 1 for examples of trajectories under various weights on the entropy term.

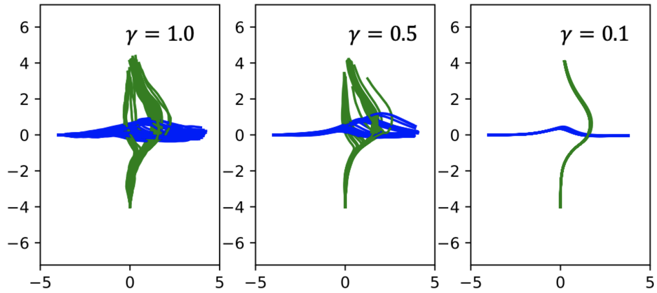


Fig. 1. Demonstrations of ECE equilibria for two-agent setup as we decrease the value of γ^i for the two agents. The variance of trajectories decreases as we decrease γ^i.

B. General Nonlinear Games

Theorem 2 provides a closed-loop expression for computing ECE policies of linear quadratic Gaussian games over a finite horizon of time T. Inspired by [47] and [48], we will leverage this result for approximating the ECE trajectories in general games with nonlinear dynamics and nonquadratic cost functions. Note that in the general nonlinear setting, we operate at a trajectory level, i.e., we start with an initial trajectory guess and refine that iteratively. We will run this method in a receding horizon fashion to hopefully mimic the feedback nature of a subgame perfect ECE. We assume that each agent i can minimize a possibly nonlinear cost function of the form

$$c^i(s_t, \boldsymbol{a}_t) = v_{s,t}^i(s_t) + \sum_{j \in [N]} {a_t^j}^\mathsf{T} R^{ij} a_t^j \tag{34}$$

where $v_{s,t}^i(s_t)$ is a nonlinear state cost. Note that in general, we could have nonquadratic nonlinear costs on the actions as well, and our algorithm will still be applicable. For simplicity of description, and since (34) captures a wide range of applications, we consider cost functions of the form (34) in this section.

To approximate ECE policies, we propose an iterative algorithm where we start by a nominal trajectory. Then, we linearize the dynamics, find a quadratic approximation of the cost function for every agent, and solve the resulting maximum entropy linear quadratic Gaussian game. We then update our reference trajectory and repeat this process until convergence (see Algorithm 1). Consequently, our algorithm shares a similar structure with differential dynamic programming (DDP) [45], [46] and iLQR methods [49]. More precisely, we start with a nominal sequence of the gain matrices and offset vectors $\{P_t^i, \alpha_t^i\}$ for every time step $t \in T$, and every agent $i \in [N]$. If such a nominal sequence of control matrices is not available, a trivial initialization is initializing all matrices to be zeros. We choose the mean value found by (23) as our reference controller. Then, at every iteration, a sequence of states and actions $\eta = \{\bar{s}, \bar{\boldsymbol{a}}^1 = (\mu_1^1, \ldots, \mu_T^1), \ldots, \bar{\boldsymbol{a}}^N = (\mu_1^N, \ldots, \mu_T^N)\}$ is found by forward simulating the system dynamics using the nominal control inputs (which are chosen to be μ_t^i for each agent i at time t). We then linearize the system dynamics (1) around the nominal trajectory η to obtain

$$\delta s_{t+1} \approx A_t \delta s_t + \sum_{j \in [N]} B_t^j \delta a_t^j \tag{35}$$

where $\delta s_t = s_t - \bar{s}_t$, $\delta a_t^i = a_t^i - \bar{a}_t^i$, and $A_t = D_s f(\cdot)$ and $B_t^j = D_{a_t^j} f(\cdot)$ are the Jacobians of the system dynamics (1) with respect to the state s_t, and actions a_t^j, respectively. We further acquire a quadratic approximation of the cost function for each agent. For each agent $i \in [N]$, we let

$$v_{s,t}^i(\bar{s}_t + \delta s_t) \approx v_{s,t}^i(\bar{s}_t) + \frac{1}{2} \delta s_t^\top H_t^i \delta s_t + l_t^{i\top} \delta s_t \tag{36}$$

denote such an approximation where $H_t^i = D_{s_t s_t} v_{s,t}^i(\cdot)$ and $l_t^i = D_{s_t} v_{s,t}^i(\cdot)$ are the Hessian and the gradient of the cost function $v_{s,t}^i(\cdot)$ with respect to s_t. Note that our formulation only considers nonlinear costs on state variables, and the dependence on agents' actions is quadratic. For a more general case, where the cost function is nonlinear in control actions too, a similar approximation could be used to derive the quadratic terms and linear terms in a_t^i. Note that all the approximations A_t, B_t^j, H_t^j, l_t^j are evaluated at η.

For the linearized system dynamics (35) and quadratized cost (36), we reach a new approximated linear quadratic Gaussian game with new variable sequences $\delta s, \delta \boldsymbol{a}^1, \ldots, \delta \boldsymbol{a}^N$. These approximations result in a new game that can be solved using Theorem 2. Once the approximated game is solved, we obtain a new sequence of mean control actions

$$\{\bar{a}_t^i + \delta a_t^{i*}, t = 0, \ldots, T-1\} \tag{37}$$

where δa_t^{i*} is the mean value of the action distribution found by solving for the ECE policies of the approximated linear quadratic Gaussian game. A new $\bar{s}_t$ is attained from the forward simulation of the original system dynamics (1) using the newly obtained nominal control actions. We repeat the above process until convergence, i.e., the deviation of the new state trajectory from the state trajectory in the previous iteration lies within a desired tolerance. Note that in our iterative process, we always forward simulate the mean value of the distribution of agents' action. Once the mean trajectories converge, we sample ECE policies by sampling control actions from (23) and (24) computed around the converged mean trajectory.

Remark 3: Applying a_t^i directly from (37) may lead to nonconvergence since the resulting trajectory could deviate too much from the original nonlinear system which we approximated around η. As in other iterative linear quadratic methods [47], [50], [51], we only take a small step in the proposed direction. At each iteration, rather than (37), we apply the following control input:

$$\bar{a}_t^i - P_t^i \delta s_t - \epsilon \alpha_t^i \tag{38}$$

Algorithm 1: Approximating ECE Trajectories.

```
Inputs
system dynamics (1), agents' cost functions (34)
Initialization
initialize the control policy using $P_t^i = 0$, and $\alpha_t^i = 0$, $\forall i \in [N]$
forward simulate and obtain $(\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_T)$, $\bar{\boldsymbol{a}}^i$, $\cdots$, $\bar{\boldsymbol{a}}^N$
while not converged do
  linear approximation of (1)
  quadratic approximation of (34)
  solve the backward recursion with (23-31)
  forward simulation using $\mu^j$'s and obtain the new trajectories
end while
return $P_t^i$, $\alpha_t^i$, and $Z_{t+1}^i$
```

where ϵ is the step size for improving our control strategy. Initially, we set $\epsilon = 1$. Drawing inspiration from line search method in optimization problems, we decrease ϵ by half until the new trajectory's deviation from the nominal trajectory is within a threshold.

VI. Learning Agents' Costs

So far, we have discussed the concept of noisy equilibria that we adopt in this article and how we can solve for approximate equilibrium policies under this model. Now, we are ready to discuss how to learn agents' cost functions c^i from a set of interaction demonstrations from humans or other experts. The main intuition behind our algorithm is similar to the single-agent maximum entropy IRL [1]. The common assumption in single-agent IRL is that the agent's cost function is parameterized as a linear combination of a set of features, and the weight of features is learned such that the expectation of the features under the learned cost function matches the empirical feature means under the demonstrations. This is achieved through an iterative algorithm where at every iteration, the difference between the feature expectations under the demonstrations and the feature expectations under the learned cost functions is utilized for updating the cost parameters.

We extend such an iterative cost learning framework to the multi-agent game setting. We assume that agents' dynamics are known, and each agent's cost function $c^i(\cdot)$ is parameterized as a weighted sum of features, i.e., for each agent $i \in [N]$, we have

$$c^i(s_t, \boldsymbol{a}_t) = w^{i^\top} \phi^i(s_t, \boldsymbol{a}_t) \tag{39}$$

where w^i is the vector of weight parameters for agent i and $\phi^i(s_t, \boldsymbol{a}_t)$ is the vector of features for agent i. Note that for each agent i, the vector of features $\phi^i(s_t, \boldsymbol{a}_t)$ depends on the entire state vector and the actions of all agents. We define $\boldsymbol{w} = (w^1, \ldots, w^N)$ to be the collection of cost weights for all agents.

Note that although the cost is parameterized as a linear combination of features, the features themselves can be nonlinear functions of states and control inputs. For example, the classical

LQR cost function is a linearly parameterized function in terms of the features that are quadratic functions of state and inputs. Hence, similar to the classical notion of function approximation which is built on linearly parameterized function spaces, in theory, this is not a limiting factor. But in practice, feature-based methods require hand-tuned features that make sense for the domain.

Our goal is to find the weight parameters $\boldsymbol{w}$ such that the feature expectation under the learned cost weights matches the empirical feature mean under the demonstrations. Let $\mathcal{D}_{\boldsymbol{w}}$ be the probability distribution over equilibrium trajectories induced by the cost parameters $\boldsymbol{w}$. We further let $\bar{\mathcal{D}}$ be the empirical distribution of equilibrium trajectories in the interaction demonstrations. Moreover, we define a trajectory $(s, \boldsymbol{a}) = (s_t, \boldsymbol{a}_t)_{t=1}^{T}$ to be the vector of agents' states and actions over a trajectory of length T. We assume that agents maintain an entropic cost equilibrium in their demonstrations.

With a slight abuse of notation, for each agent i, we denote the empirical mean of the features under the interaction demonstrations by $\mathbb{E}_{(s,\boldsymbol{a})\sim\bar{\mathcal{D}}}\, \phi^i(s, \boldsymbol{a})$. Likewise, we let $\mathbb{E}_{(s,\boldsymbol{a})\sim\mathcal{D}_{\boldsymbol{w}}}\, \phi^i(s, \boldsymbol{a})$ be the expected value of the features under the equilibrium trajectories induced by the weight vector $\boldsymbol{w}$.

We want to find weight parameters $\boldsymbol{w}$ to induce a probability distribution $\mathcal{D}_{\boldsymbol{w}}$ over equilibrium trajectories such that feature expectation under the learned cost weights matches the empirical feature mean under demonstrations, i.e., $\mathbb{E}_{(s,\boldsymbol{a})\sim\bar{\mathcal{D}}}\, \phi^i(s, \boldsymbol{a}) = \mathbb{E}_{(s,\boldsymbol{a})\sim\mathcal{D}_{\boldsymbol{w}}}\, \phi^i(s, \boldsymbol{a})$ for all agents $i \in [N]$. To this end, we propose an iterative algorithm. We initialize the cost weights $\boldsymbol{w}$ for all agents. At each iteration of the algorithm, we iterate over all the agents. For agent i, we compute the difference between the feature expectations under the demonstrations and the current model $\mathbb{E}_{(s,\boldsymbol{a})\sim\bar{\mathcal{D}}}\, \phi^i(s, \boldsymbol{a}) - \mathbb{E}_{(s,\boldsymbol{a})\sim\mathcal{D}_{\boldsymbol{w}}}\, \phi^i(s, \boldsymbol{a})$ and use this difference for updating the weight parameters

$$w^i \leftarrow w^i - \beta \left(\mathbb{E}_{(s,\boldsymbol{a})\sim\bar{\mathcal{D}}}\, \phi^i(s, \boldsymbol{a}) - \mathbb{E}_{(s,\boldsymbol{a})\sim\mathcal{D}_{\boldsymbol{w}}}\, \phi^i(s, \boldsymbol{a})\right) \quad (40)$$

where β is the learning rate. Note that once w^i is updated, the entire vector of cost weights $\boldsymbol{w}$ is in fact updated. In the single-agent setting where an agent is noisily optimizing a cost function, the update rule (40) is the gradient of the likelihood of trajectories under weight parameters $\boldsymbol{w}$. In other words, in the single-agent setting, (40) solves a maximum likelihood problem for finding weight parameters $\boldsymbol{w}$ that maximize the likelihood of demonstrations. If we fix all agents except agent i, similar to the single-agent setting, we can interpret (40) as a gradient-descent update rule. Once the cost parameters of agent i are updated, we compute the feature expectations under the new set of cost parameters and update the cost parameters for the next agent. We iterate over all agents i and repeat this process until convergence. This can be interpreted as a block coordinate descent method for solving the coupled parameter estimation problem for the multi-agent game.

Algorithm 2 summarizes our method. Note that in a multi-agent setting, once the cost parameters of one agent are updated, we need to recompute the feature expectation for updating the next agent's cost since agents' cost parameters and feature expectations are interdependent, i.e., a change in one agent's cost parameters affects the expectation of features for other agents. This is due to the coupling among agents at entropic cost equilibrium.

Algorithm 2: Multi-Agent Inverse Reinforcement Leaning.

```
 1:  Inputs
 2:  system dynamics (1), agents' cost features $\phi^i(\cdot)$, and
     the set of interaction demonstrations.
 3:  Initialization
 4:  for each agent $i \in [N]$, compute the empirical feature
     mean under demonstrations $\mathbb{E}_{(s,\boldsymbol{a})\sim\bar{\mathcal{D}}}\, \phi^i(s, \boldsymbol{a})$
 5:  initialize the cost weights $\boldsymbol{w}$
 6:  while not converged do
 7:    for each agent $i \in [N]$ do
 8:      Compute $\mathbb{E}_{(s,\boldsymbol{a})\sim\mathcal{D}_{\boldsymbol{w}}}\, \phi^i(s, \boldsymbol{a})$
 9:      Compute the difference
           $\mathbb{E}_{(s,\boldsymbol{a})\sim\bar{\mathcal{D}}}\, \phi^i(s, \boldsymbol{a}) - \mathbb{E}_{(s,\boldsymbol{a})\sim\mathcal{D}_{\boldsymbol{w}}}\, \phi^i(s, \boldsymbol{a})$
10:      Update cost weight for agent $i$:
           $w^i \leftarrow w^i - \beta\big(\mathbb{E}_{(s,\boldsymbol{a})\sim\bar{\mathcal{D}}}\, \phi^i(s, \boldsymbol{a}) - \mathbb{E}_{(s,\boldsymbol{a})\sim\mathcal{D}_{\boldsymbol{w}}}\, \phi^i(s, \boldsymbol{a})\big)$
11:    end for
12:  end while
14:  returncost parameters $w^i$ for all agents $i \in [N]$
```

Algorithm 3: Approximating Feature Expectations.

```
Inputs
system dynamics (1), agents' cost features $\phi^i(\cdot)$,
agents' cost wights $w^i$, number of samples $p$
for $1 \leq j \leq p$ do
  Sample an ECE trajectory $(s_j, \boldsymbol{a}_j)$
end for
return $\frac{1}{p}\sum_j \phi^i(s_j, \boldsymbol{a}_j)$
```

Each iteration of Algorithm 2 at line 8 requires computing agents' feature expectations $\mathbb{E}_{(s,\boldsymbol{a})\sim\mathcal{D}_{\boldsymbol{w}}}\, \phi^i(s, \boldsymbol{a})$ under ECE policies. However, computing $\mathbb{E}_{(s,\boldsymbol{a})\sim\mathcal{D}_{\boldsymbol{w}}}\, \phi^i(s, \boldsymbol{a})$ for general nonlinear games is not tractable. We propose to approximate this by sampling ECE policies through Algorithm 1 under the current cost parameter estimates. In each iteration, we sample p ECE equilibrium trajectories $(s_1, \boldsymbol{a}_1), \ldots, (s_p, \boldsymbol{a}_p)$ using Algorithm 1, and approximate these through the following:

$$\mathbb{E}_{(s,\boldsymbol{a})\sim\mathcal{D}_{\boldsymbol{w}}}\, \phi^i(s, \boldsymbol{a}) \approx \frac{1}{p}\sum_{j=1}^{p} \phi^i(s_j, \boldsymbol{a}_j). \quad (41)$$

Thus, with known system dynamics, Algorithm 1 lets us compare equilibrium trajectories under the learned cost parameters with the demonstration trajectories and update the cost parameters if needed. Algorithm 3 summarizes the computation of feature expectations.

VII. Experiments

In this section, we evaluate the performance of our MA-IRL algorithm in two different scenarios. In the first scenario, we

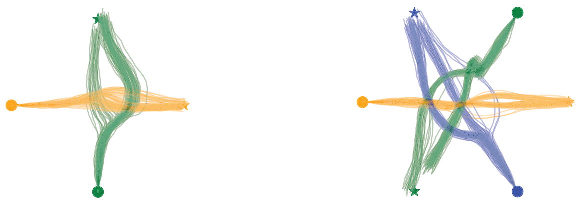
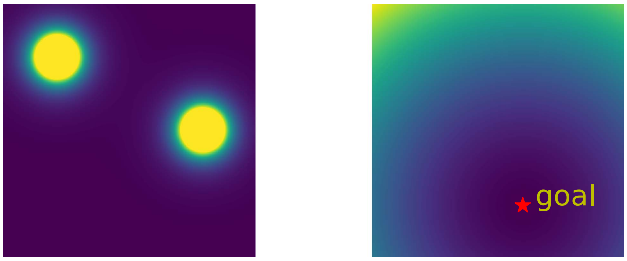

(a) (b)

Fig. 2. Demonstrations for two-player and three-player collision avoidance motion planning. Trajectories of different players are shown in green, yellow, and blue, respectively. Players have different cost functions. Due to the stochasticity of the equilibrium policies, we observe different modes of interactions. (a) Two-player case. (b) Three-player case.



(a) (b)

Fig. 3. Cost features for the motion planning with collision avoidance task. Costs are higher close to other agents and away from the goal location. (a) Feature associated with distance to other agents. (b) Feature associated with distance to the goal location.

consider a motion planning task with multiple agents that navigate to goal locations while avoiding collisions with each other. This is a synthetic dataset which enables us to compare the learned cost functions with ground truth cost parameters. In the second scenario, we use the INTERACTION dataset [5] which contains trajectories in interactive traffic scenes, such as intersection, roundabouts, and highway merging, collected from different locations in different countries.

We compare our algorithm with a baseline method, continuous inverse optimal control (CIOC) [3], [52], which is an IOC algorithm for large continuous domains and does not assume any feedback interaction among the observed agents. We further compare the performance of our method with [32] which is a recent work on inferring objectives in continuous dynamic games (IOCD). In addition, for the INTERACTION dataset, we also compare with the IDM [6], [53] as our reference model. IDM is a widely used expert-designed model to simulate traffic flow that is known to yield accurate predictions of drivers' trajectories.

A. Motion Planning With Collision Avoidance

We first consider a synthetic environment where two or three agents move to their goal locations and try to avoid collisions with each other. Demonstrations for all scenarios are generated by solving an approximate entropic cost equilibrium as described in Section V. Fig. 2 illustrates the demonstration trajectories for the scenarios with two and three agents. Each agent wants to minimize a cost function which is linear in the set of features: Distance to the goal position, distance to other agents, and control effort. Plots of different features are shown in Fig. 3. The reference trajectory is a straight line between initial location and goal location. Each agent has different weightings on the features, which leads to different interactive behaviors as shown in Fig. 2.

The demonstration dataset contains 200 trajectories. We use MA-IRL to learn the cost function coefficients for all agents jointly. To apply CIOC method, we learn the cost coefficients of each agent individually and treat all other agents as obstacles. In our implementation of the CIOC method, we use a generic optimization solver. The computation time of CIOC scales cubically with the planning horizon. Hence, for computational tractability, we partition the demonstration trajectories into sections of shorter trajectories. For the implementation of the IOCD method, we input the demonstration trajectories as noisy observations to the full observation solver which has access to all the states of all the agents in the demonstration trajectories.

We evaluate the performance of all algorithms in test scenarios where the initial configurations are generated randomly and are different from the demonstration dataset. To evaluate the performance of the algorithms, we solve the ECE solutions with the learned cost functions using our MA-IRL approach, the CIOC approach, the IOCD method, as well as with the true cost functions in the test scenarios. We simulate the stochastic equilibrium policies for 200 trials in each case and compute the feature expectations. If the feature distribution from the learned cost functions is similar to the feature distribution under the true cost parameters, it means that our learned costs are close to the true costs. The simulation results are shown in Tables I and II.

We compute the Kullback–Leibler divergence, which captures the difference between two distributions. Lower values of KL divergence between the two distributions means that the distribution of trajectories under learned cost parameters is closer to the ground truth distribution. As demonstrated by Tables I and II, in the two different test tasks with different and randomized initial configurations, MA-IRL algorithm achieves closer similarity to the true feature distribution, which means that the MA-IRL learned cost parameters generalize to different tasks better than CIOC and IOCD.

In addition, we follow prior work and also compute task-relevant statistics to evaluate how much the recovered costs and the corresponding stochastic policies resemble the demonstrations. We compute the distance to goal location at the end of the simulation horizon (6 s) and demonstrate the statistic of this feature for the two-player setup in Table III. We can see that MA-IRL better resembles the true cost functions.

In our simulations, the forward planning problem takes on average $9.51\,\mathrm{ms}$ to solve at each iteration for the two-agent scenario and $63.28\,\mathrm{ms}$ for the three-agent scenario. For the inverse problem, it takes around 2 h for the two-agent case and 26 h for three-agent case on a laptop with Intel 2.7Ghz CPU. The CIOC algorithm takes about 24 h to converge with shorter trajectory horizon as the full length trajectory becomes computationally infeasible. Note that when running our method

TABLE I
THREE-AGENT COLLISION AVOIDANCE SIMULATIONS

KL DIV		**agent 1**			**agent 2**			**agent 3**		
		Tracking	Control	Obstacle	Tracking	Control	Obstacle	Tracking	Control	Obstacle
task 1	MA-IRL (ours)	**0.031**	0.063	**0.167**	**0.074**	0.144	**0.052**	**0.050**	0.066	**0.127**
	CIOC	2.583	**0.054**	0.939	1.083	**0.039**	0.489	0.082	**0.022**	0.629
	IOCD	1.060	36.336	26.684	0.386	9.912	63.118	0.384	13.814	47.135
task 2	MA-IRL (ours)	0.096	**0.019**	**0.116**	**0.054**	**0.019**	**0.148**	**0.030**	**0.108**	**0.080**
	CIOC	1.104	0.073	0.701	1.685	0.032	0.780	2.198	0.171	0.487
	IOCD	**0.014**	34.408	0.262	4.729	38.548	0.429	0.762	27.753	0.262

KL-divergence of feature distributions between demonstrations and simulations from learned costs. Smaller values are better. The two different test tasks refer to two different and randomized initial configurations.
The bold entities represent the lowest KL divergence value for the corresponding features.

TABLE II
TWO-AGENT COLLISION AVOIDANCE SIMULATIONS

KL DIV		**agent 1**			**agent 2**		
		Tracking	Control	Obstacle	Tracking	Control	Obstacle
task 1	MA-IRL (ours)	**0.054**	**0.024**	**0.044**	**0.134**	**0.059**	**0.044**
	CIOC	1.701	0.165	1.766	0.836	0.065	1.766
	IOCD	53.597	25.273	8.240	2.008	13.136	8.340
task 2	MA-IRL (ours)	**0.025**	**0.080**	**0.061**	**0.034**	**0.053**	**0.061**
	CIOC	1.904	0.146	1.086	0.152	0.144	1.086
	IOCD	0.103	28.279	12.593	3.381	20.111	12.593

KL divergence of feature distributions between demonstrations and simulations from learned costs. Smaller values are better. The two different test tasks refer to two different and randomized initial configurations.
The bold entities represent the lowest KL divergence value for the corresponding features.

TABLE III
TASK-RELEVANT STATISTICS FOR THE TWO-PLAYER COLLISION AVOIDANCE CASE

dist.		true	MA-IRL	CIOC	IOCD
T1	A1	0.21±0.10	0.23±0.12	0.14±0.08	**0.20±0.25**
	A2	0.32±0.15	**0.32±0.16**	0.25±0.10	0.30±0.28
T2	A1	0.72±0.70	**0.86±0.74**	0.38±0.43	0.07±0.09
	A2	2.08±1.88	**1.80±1.95**	1.35±1.62	0.08±0.12

Distances to the goal location at the end of simulations are shown. Overall, the statistics of the MA-IRL approach are closer to the true cost samples compared to other methods. A1 stands for Agent 1, and A2 stands for Agent 2 while T1 and T2 refer to Tasks 1 and 2.
The bold entities represent the average distance to goal location that is the closest to the demonstrations.

using Algorithm 2, changing the order of agents does not affect the empirical convergence as shown in Fig. 4.

B. Interaction Dataset

We further apply our MA-IRL algorithm on a real world traffic dataset, the INTERACTION dataset [5], which has highly interactive driving scenarios collected from different countries. Scenarios such as roundabouts, highway merging, and intersections are included in the data.

1) Data Processing: We pick a highway merging scenario as it involves vehicles negotiating their merging actions, leading to challenging game-theoretic planning for all vehicles. Among all the merging trajectories, each data point consists of the following three vehicles: 1) a merging vehicle whose trajectory starts from the onramp and ends on a main highway lane, a leader vehicle that is ahead of the merging vehicle, and a follower vehicle that is behind the merging vehicle in the highway. Leader vehicles are treated as obstacles which do not interact with the other two vehicles, and we assume that the merging vehicle and the follower vehicle follow an ECE solution. We discard the trajectories that are too long or too short (longer than 9 s or shorter than 5 s) and end up with 87 trajectories in total. All 87 data points are recorded in the same highway onramp in China from different recordings. The average time to finish merging is 6.3 s in the dataset. Fig. 5 shows an example snapshot, where car 8 is merging onto highway after car 13.

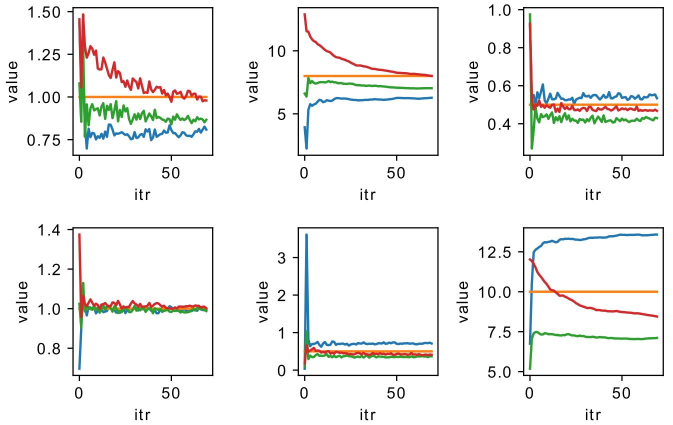


Fig. 4. We randomize the order of agents during inverse learning problem and the cost parameters converge consistently regardless of the order of iterations and the initial conditions. Each plot corresponds to one learnt parameter and different colors represent a different initial condition or the order of iteration. *X*-axis is the number of iterations and *Y*-axis is the parameter value.

2) Features: For this challenging and complex driving task, we propose features that capture efficiency (driving forward and achieving the lane merge for the merging vehicle), comfort and energy efficiency (penalizing large accelerations), and safety (penalizing relative speed and a Gaussian-cost on close proximity with other cars). In total, we learn the cost coefficients for nine features for both merging and follower vehicles.

To evaluate the learning results on the INTERACTION dataset, since we do not know the true cost function, we cannot compute the corresponding cost distribution. Instead, we compute the root-mean -squared error (rmse) on trajectories that are

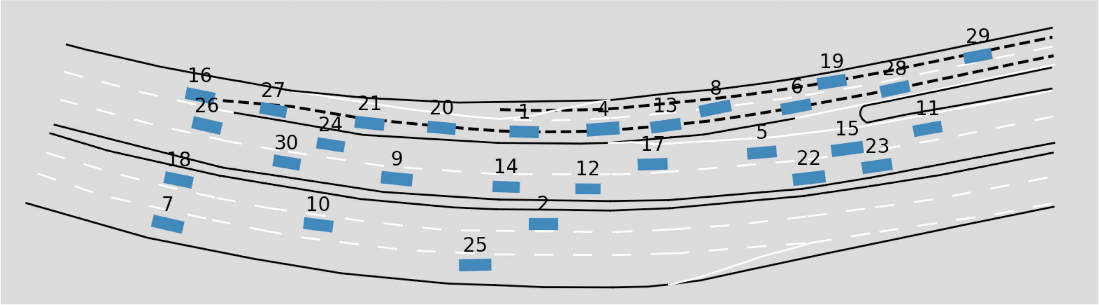


Fig. 5. Snapshot of highway merging scenario in the INTERACTION dataset. Vehicle 8 is a merging vehicle between cars 13 and 6.

TABLE IV
Task Statistics for the Generated Simulations Using Learned Cost Function/Policy Compared to Human Demonstrations From the Same Initial Configuration

task statistics	true	MA-IRL	IDM	CIOC
M avg. speed [m/s]	3.272	3.372	2.876	2.949
F avg. speed [m/s]	2.893	2.592	2.574	2.410
ML avg. dist [m]	7.770	7.438	7.951	8.041
MF avg. dist [m]	7.447	8.716	6.844	7.009

The notation M refers to the merging vehicle, and F denotes the following vehicle.

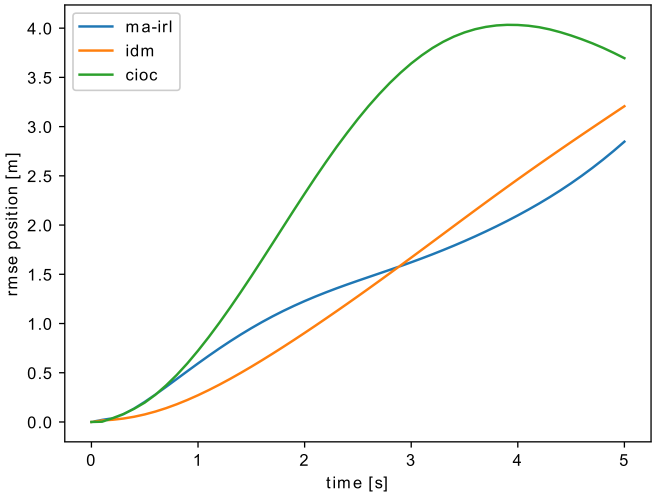


Fig. 6. rmse for the predicted trajectories using the INTERACTION dataset.

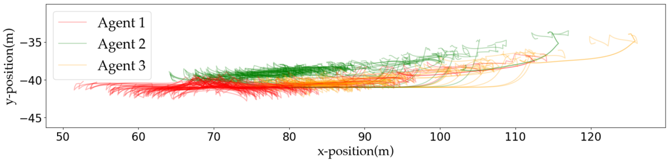


Fig. 7. Trajectories that are predicted under IOCD in the merging scenario using the INTERACTION dataset are nonsmooth trajectories which are not realistic.

generated from the learned cost functions. For each trajectory tuple in the dataset, we run ten trials using the ECE policy with the learned cost function from both MA-IRL and CIOC. Then, we compute the average deviation from the true trajectories over 5 s. For position errors, MA-IRL achieves significantly better prediction accuracy than the CIOC model due to the interactive nature of this scenario. IDM's performance is roughly on par with MA-IRL. We note that IDM is a popular expert designed and empirically validated driving model for traffic simulation and is only applicable for car following scenarios. Hence, IDM acts as a proxy for the unknown ground truth policy in this scenario. Moreover, we learn the parameters of IDM to specifically fit this dataset, thereby directly approximating the control policy of the vehicles in the dataset.

For MA-IRL, the forward problem takes 8.62 ms and the inverse problem consistently converges after 100 iterations. IDM fits the parameters using least squares method. The total computation time is 3 h with an initial parameter sweep to avoid local optima. CIOC takes around 24 h.

In addition, we follow previous work and evaluate the learned policy by computing task-related statistics from demonstrations and generated samples. We measure average speed of both merging and following cars, the average Euclidean distance between the leader and merging cars, and average Euclidean distance between the merging and following cars. The results are summarized in Table IV. From Table IV, we can see that MA-IRL achieves good approximation results in average speed of the merging vehicle and the follower vehicle, which is on par with IDM's performance and better than CIOC. For average distance between merging vehicle and other vehicles, the performance of all three algorithms are similar and have larger deviation than speed statistics. We think the reason is that average distance is subject to more uncertainty. In Fig. 6, we plotted the rmse of the predicted trajectories for all methods. As Fig. 6 demonstrates, the position rmse of MA-IRL is similar to that of IDM and better than CIOC.

We would like to point out that on the INTERACTION dataset, IOCD resulted in nonsmooth trajectories that were far from realistic. We found that IOCD overfitted to the training data and could not generalize well to the test data which resulted in nonsmooth trajectories as shown in Fig. 7. Consequently, the underlying generated trajectories are not smooth and realistic; and, hence we did not discuss its performance in detail.

VIII. Conclusion

In this article, we presented an algorithm for maximum entropy IRL in multi-agent settings. To enable learning from boundedly rational agents such as humans, we defined a notion of noisy equilibrium called ECE. We proved that this noisy equilibrium is indeed an extension of the maximum entropy principle to the multi-agent setting. We then proved that ECE policies can be obtained in closed form for the special class of linear quadratic Gaussian games. We further presented an algorithm for approximating ECE policies for general nonlinear games. Finally, knowing how to find maximum-entropy multi-agent policies, we provided an iterative algorithm for learning agents' costs from a set of demonstrations when cost functions are represented as a linear combination of a set of features. We verified and validated our algorithm using both synthetic and real-world data demonstrating that our MA-IRL algorithm can successfully capture the interactions between agents and learn accurate models of agents' costs.

Limitations and Future Work: While our simulations and experiments show the effectiveness of our MA-IRL algorithm, a key future direction is to enable learning general nonlinear costs. In this article, we assumed that agents' costs can be represented as a weighted sum of a set of features. An important future direction is to relax this assumption. Moreover, to avoid handpicking the set of relevant features a-priori, we would like to study the extensions of this work to enable learning cost functions that are represented in the form of function approximators such as neural networks. Finally, the current work can be extended to the case of unknown system dynamics as well. We assumed that the system dynamics were known and fixed. We believe that our method can be further extended to systems with unknown dynamics.

Appendix A
Proof of Theorem 2

We provide the proof for Theorem 2. We know from Theorem 1 that to find the ECE policies of a linear quadratic game $G = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \boldsymbol{c}, T)$, we can equivalently find the Nash equilibrium policies of the auxiliary game $\tilde{G} = (\mathcal{S}, \boldsymbol{\mathcal{A}}, f, g, \tilde{\mathbf{c}}, T)$. The key idea is that at Nash equilibria of the game $\tilde{G}$, the cost-to-go or the value function for each agent at any given state can be represented in closed form via a quadratic function of the system state. Using dynamic programming, the closed-form solution of the cost-to-go function can be propagated backwards in time to find both the Nash equilibrium policy and the cost-to-go of agent i.

More precisely, let $\boldsymbol{\pi}^*$ denote a set of Nash equilibrium policies of the game $\tilde{G}$. Consider an agent $i \in [N]$. Fix the policies of all the other agents $\boldsymbol{\pi}^{-i^*}$. Then, using Definition 1, for the game $\tilde{G}$, the (Nash) equilibrium policy of the agent π^{i*} optimizes

$$\min_{\pi^i} \mathbb{E}_{\pi^i, \boldsymbol{\pi}^{-i*}} \sum_{k=1}^{T} \frac{1}{2}\left(s_k^\top Q^i s_k + l_i^\top s_k + \sum_{j\in[N]} {a_k^j}^\top R^{ij} a_k^j\right) - \sum_{k=1}^{T} \mathcal{H}(\pi_k^i(.|s_t)). \tag{42}$$

First, note that for any policy $\pi^i(.|s_t)$, since the dynamics are linear and the cost is quadratic, the expectation of the terms inside the parenthesis in (42) depends only the first and second order moments of the policy π^i. Therefore, for any fixed first and second order moments of the policy π^i, the optimal policy π^{i*} will be the distribution with maximum entropy subject to fixed first and second order moments. Leveraging the fact that with fixed first and second order moments, the maximum entropy distribution is a normal distribution (see for instance [54, Th. 8.6.5]), we can conclude that the optimal policy for each agent i at any time step π_t^{i*} is normally distributed. Hence, it remains to find the mean and covariance of the optimal policy at any given time step. For every agent i, let the mean and covariance of agent i's policy be represented by μ_t^i and Σ_t^i at every time step t. The optimal solution to (42) can be found using dynamic programming. Let $V_t^{i*}(s_t)$ denote the cost-to-go of agent i at Nash equilibrium at a given time step t

$$V_t^{i*}(s_t) := \min_{\pi^i} \mathbb{E}_{\pi^i, \boldsymbol{\pi}^{-i*}} \sum_{k=t}^{T} \frac{1}{2}\left(s_k^\top Q^i s_k + {l^i}^\top s_k + \sum_{j\in[N]} {a_k^j}^\top R^{ij} a_k^j\right) - \sum_{k=t}^{T} \mathcal{H}(\pi_k^i(.|s_k)). \tag{43}$$

We prove via induction that the cost-to-go for each agent at a given time step t is a quadratic function of the state s_t. We assume for time $t+1$, we have

$$V_{t+1}^{i*}(s_{t+1}) = \frac{1}{2} s_{t+1}^\top Z_{t+1}^i s_{t+1} + {\xi^i}^\top_{t+1} s_{t+1} + n_{t+1}^i \tag{44}$$

where Z_{t+1}^i, ξ_{t+1}^i, and n_{t+1}^i are the coefficients of the quadratic cost-to-go at time $t+1$. We will propagate the cost-to-go structure (44) backwards in time to prove that

$$V_t^{i*}(s_t) = \frac{1}{2} s_t^\top Z_t^i s_t + {\xi_t^i}^\top s_t + n_t^i. \tag{45}$$

We prove this via dynamic programming. For time step t, we have

$$V_t^{i*}(s_t) = \min_{\pi_t^i} \mathbb{E}_{\pi_t^i, \boldsymbol{\pi}_t^{-i*}} \frac{1}{2}\left(s_t^\top Q^i s_t + \sum_{j\in[N]} {a_t^j}^\top R^{ij} a_t^j\right) - \mathcal{H}(\pi_t^i(.|s_t)) + \mathbb{E}_{s_{t+1}} V_{t+1}^{i*}(s_{t+1}). \tag{46}$$

Using dynamics (20), the structure of cost-to-go function for time $t+1$ from (44), and the fact that the policies for every time step are normally distributed, one can verify that

$$\begin{aligned}\mathbb{E}\, V_{t+1}^{i*}(s_{t+1}) = \frac{1}{2}\Bigg[& s_t^\top A^\top Z_{t+1}^i A s_t + \sum_{j\in[N]} {\mu_t^j}^\top {B^j}^\top Z_{t+1}^i A s_t \\ & + s_t^\top A^\top Z_{t+1}^i \sum_{j\in[N]} B^j \mu_t^j \\ & + \sum_{j\in[N]} \left(B^j \mu_t^j\right)^\top Z_{t+1}^i + n_{t+1}^i \\ & + \operatorname{tr}\left(Z_{t+1}^i \sum_{j\in[N]} B^j \Sigma_t^j {B^j}^\top + Z_{t+1}^i \right)\end{aligned}$$

$$+{\xi^i_{t+1}}^{\top}\left(As_t+\sum_{j\in[N]}B^j\mu_t^j\right)\Bigg]. \tag{47}$$

Moreover, we know that for a normal policy π_t^i with mean μ_t^i and covariance Σ_t^i, the entropy of the policy is

$$\mathcal{H}(\pi_t^i(.|s_t))=\frac{m^i}{2}\log(2\pi e)+\frac{1}{2}\log\det(\Sigma_t^i) \tag{48}$$

where m^i is the dimension of the action space of agent i, and e is the Euler's number. Using (47) and (48), (46), which is the cost-to-go at time t, can be rewritten as a minimization with respect to μ_t^i and Σ_t^i when we fix the mean and covariance of other agents' policies

$$\begin{aligned}V_t^{i*}(s_t)=\min_{\mu^i,\Sigma^i}\ &\mathbb{E}_{\pi^i,\boldsymbol{\pi}^{-i*}}\frac{1}{2}\Bigg[s_t^\top Q^i s_t+\sum_{j\in[N]}{\mu_t^j}^\top R^{ij}\mu_t^j\\ &+s_t^\top A^\top Z^i_{t+1}As_t+\sum_{j\in[N]}\mu_t^j {B^j}^\top Z^i_{t+1}As_t\\ &+s_t^\top A^\top Z^i_{t+1}\sum_j B^j\mu^j+\left(\sum_{j\in[N]}B^j\mu_t^j\right)^\top\\ &\times Z^i_{t+1}\left(\sum_{j\in[N]}B^j\mu_t^j\right)\Bigg]\\ &+{\xi^i_{t+1}}^\top\left(As_t+\sum_{j\in[N]}B^j\mu^j\right)+n^i_{t+1}\\ &-\frac{1}{2}\log\det(\Sigma_t^i)-\frac{1}{2}m^i2\pi e\\ &+\frac{1}{2}\sum_{j\in[N]}\mathrm{tr}(R^{ij}\Sigma_t^j+Z^i_{t+1}\left(\sum_{j\in[N]}B^j\Sigma_t^j{B^j}^\top\right)+Z^i_{t+1}).\end{aligned} \tag{49}$$

The objective function of (49) is a convex quadratic function of μ_t^i and a convex function of Σ_t^i. Taking the derivative of the objective function in (49) with respect to μ_t^i and Σ_t^i and setting the derivatives equal to zero, one can verify that μ_t^{i*} and Σ_t^{i*} have the structure of (23) and (24). By replacing μ_t^{i*} and Σ_t^{i*} in (49) and rearranging the terms, one can see that $V_t^{i*}(s_t)$ is also a quadratic function of the state s_t. This proves that at every time step t, $V_t^{i*}(s_t)=\frac{1}{2}s_t^\top Z_t^i s_t+{\xi_t^i}^\top s_t+n_t^i$. By matching the coefficients of this quadratic function, recursions (27), (28), (29), and (30) are obtained. Note that we also need to verify the base case for our induction, i.e, the cost-to-go at the final time step $V_T^{i*}(s_T)$ is also quadratic. At the final time step T, the optimal policy π_T^{i*} is a zero-mean normal distribution with covariance matrix $\Sigma_T^i=\left(R^i\right)^{-1}$. Plugging in this policy, we verify that in the final time step the cost-to-go is also quadratic in the state s_T. This proves the base case for our induction, which completes our proof that the optimal cost-to-go at any time step t is quadratic in the state s_t. Now that we have proved that at every time step, $V_t^{i*}(s_t)$ is of the form (45), we can obtain the recursions (25) and (26) on the matrix P_t^i and the vector α_t^i by replacing (23) and (24) in (46) and matching the coefficients.

References

[1] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. 23rd AAAI Conf. Artif. Intell.*, Chicago, IL, USA, vol. 8, 2008, pp. 1433–1438.

[2] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Ph.D. dissertation, Dept. Mach. Learn., Carnegie Mellon Univ., Pittsburgh, PA, USA, Dec. 2010.

[3] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," in *Proc. 29th Int. Cof. Int. Conf. Mach. Learn.*, 2012, pp. 475–482.

[4] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," *Adv. Neural Inf. Process. Syst.*, vol. 29 2016.

[5] W. Zhan et al., "Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps," Sep. 2019, *arXiv:1910.03088 [cs, eess]*.

[6] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E*, vol. 62, no. 2, 2000, Art. no. 1805.

[7] R. E. Kalman, "When is a linear control system optimal?," *J. Basic Eng.*, vol. 86, no. 1, pp. 51–60, 1964.

[8] A. Y. Ng et al., "Algorithms for inverse reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, vol. 1, 2000, pp. 663–670.

[9] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 1. [Online]. Available: https://dl.acm.org/doi/10.1145/1015330.1015430

[10] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 49–58.

[11] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4572–4580.

[12] S. Albrecht et al., "Imitating human reaching motions using physically inspired optimization principles," in *Proc. 11th IEEE-RAS Int. Conf. Humanoid Robots*, 2011, pp. 602–607.

[13] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion–an inverse optimal control approach," *Auton. Robots*, vol. 28, no. 3, pp. 369–383, 2010.

[14] P. Englert, N. A. Vien, and M. Toussaint, "Inverse KKT: Learning cost functions of manipulation tasks from demonstrations," *Int. J. Robot. Res.*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[15] C. Awasthi, "Forward and inverse methods in optimal control and dynamic game theory," *Master's thesis*, Univ. Minnesota, Minneapolis, MN, USA, 2019.

[16] M. Menner and M. N. Zeilinger, "Maximum likelihood methods for inverse learning of optimal controllers," *IFAC-PapersOnLine*, Jan. 1, 2020, vol. 53, no. 2, pp. 5266–5272.

[17] S. Natarajan, G. Kunapuli, K. Judah, P. Tadepalli, K. Kersting, and J. Shavlik, "Multi-agent inverse reinforcement learning," in *Proc. 9th Int. Conf. Mach. Learn. Appl.*, 2010, pp. 395–400.

[18] K. Waugh, B. D. Ziebart, and J. A. Bagnell, "Computational rationalization: The inverse equilibrium problem," 2013, *arXiv:1308.3506*.

[19] X. Lin, P. A. Beling, and R. Cogill, "Multi-agent inverse reinforcement learning for zero-sum games," 2014, *arXiv:1403.6508*.

[20] X. Lin, S. C. Adams, and P. A. Beling, "Multi-agent inverse reinforcement learning for certain general-sum stochastic games," *J. Artif. Intell. Res.*, vol. 66, pp. 473–502, 2019.

[21] Y. Wen, Y. Yang, R. Luo, and J. Wang, "Modelling bounded rationality in multi-agent interactions by generalized recursive reasoning," 2019, *arXiv:1901.09216*.

[22] L. Yu, J. Song, and S. Ermon, "Multi-agent adversarial inverse reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7194–7201.

[23] J. Song, H. Ren, D. Sadigh, and S. Ermon, "Multi-agent generative adversarial imitation learning," *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[24] J. Grau-Moya, F. Leibfried, and H. Bou-Ammar, "Balancing two-player stochastic games with soft Q-learning," 2018, *arXiv:1802.03216*.

[25] Y. Savas, M. Ahmadi, T. Tanaka, and U. Topcu, “Entropy-regularized stochastic games,” in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 5955–5962.

[26] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, “Social behavior for autonomous vehicles,” *Proc. Nat. Acad. Sci.*, vol. 116, no. 50, pp. 24972–24978, 2019.

[27] S. Lecleac’h, M. Schwager, and Z. Manchester, “Lucidgames: Online unscented inverse dynamic games for adaptive trajectory prediction and planning,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 5485–5492, Jul. 2021.

[28] F. Köpf, J. Inga, S. Rothfuß, M. Flad, and S. Hohmann, “Inverse reinforcement learning for identification in linear-quadratic dynamic games,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14902–14908, 2017.

[29] S. Rothfuß, J. Inga, F. Köpf, M. Flad, and S. Hohmann, “Inverse optimal control for identification in non-cooperative differential games,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14909–14915, 2017.

[30] J. Inga, E. Bischoff, F. Köpf, and S. Hohmann, “Inverse dynamic games based on maximum entropy inverse reinforcement learning,” 2019, *arXiv:1911.07503*.

[31] C. Awasthi and A. Lamperski, “Inverse differential games with mixed inequality constraints,” in *Proc. Amer. Control Conf.*, 2020, pp. 2182–2187.

[32] L. Peters, D. Fridovich-Keil, V. Rubies-Royo, C. J. Tomlin, and C. Stachniss, “Inferring objectives in continuous dynamic games from noise-corrupted partial state observations,” in *Proc. Robot.: Sci. Syst.*, Jul. 2021, doi: 10.15607/RSS.2021.XVII.030.

[33] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, “The complexity of computing a Nash equilibrium,” *SIAM J. Comput.*, vol. 39, no. 1, pp. 195–259, 2009.

[34] H. A. Simon, “A behavioral model of rational choice,” *Quart. J. Econ.*, vol. 69, no. 1, pp. 99–118, 1955.

[35] H. A. Simon, “Rational decision making in business organizations,” *Amer. Econ. Rev.*, vol. 69, no. 4, pp. 493–513, 1979.

[36] R. D. McKelvey and T. R. Palfrey, “Quantal response equilibria for normal form games,” *Games Econ. Behav.*, vol. 10, no. 1, pp. 6–38, 1995.

[37] R. D. McKelvey and T. R. Palfrey, “Quantal response equilibria for extensive form games,” *Exp. Econ.*, vol. 1, no. 1, pp. 9–41, 1998.

[38] S. P. Anderson, J. K. Goeree, and C. A. Holt, “Noisy directional learning and the logit equilibrium,” *Scand. J. Econ.*, vol. 106, no. 3, pp. 581–602, 2004.

[39] S. Levine and V. Koltun, “Guided policy search,” in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1–9.

[40] S. Levine and V. Koltun, “Learning complex neural network policies with trajectory optimization,” in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 829–837.

[41] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, “Reinforcement learning with deep energy-based policies,” in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1352–1361.

[42] J. Schulman, X. Chen, and P. Abbeel, “Equivalence between policy gradients and soft Q-learning,” 2017, *arXiv:1704.06440*.

[43] S. Levine, “Reinforcement learning and control as probabilistic inference: Tutorial and review,” 2018, *arXiv:1805.00909*.

[44] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. Philadelphia, PA, USA: SIAM, 1998.

[45] D. H. Jacobson, “New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach,” *J. Optim. Theory Appl.*, vol. 2, no. 6, pp. 411–440, 1968.

[46] Y. Tassa, N. Mansard, and E. Todorov, “Control-limited differential dynamic programming,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 1168–1175.

[47] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, “Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1475–1481.

[48] M. Wang, N. Mehr, A. Gaidon, and M. Schwager, “Game-theoretic planning for risk aware interactive agents,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Las Vegas, NV, USA(Virtual), 2020, pp. 6998–7005.

[49] W. Li and E. Todorov, “Iterative linear quadratic regulator design for nonlinear biological movement systems,” in *Proc. ICINCO (1)*. Citeseer, 2004, pp. 222–229.

[50] S. Yakowitz, “Algorithms and computational techniques in differential dynamic programming,” *Control Dyn. Syst.: Adv. Theory Appl.*, vol. 31, pp. 75–91, 2012.

[51] J. Van Den Berg, S. Patil, and R. Alterovitz, “Motion planning under uncertainty using iterative local optimization in belief space,” *Int. J. Robot. Res.*, vol. 31, no. 11, pp. 1263–1278, 2012.

[52] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, “Planning for autonomous cars that leverage effects on human actions,” in *Proc. Robot.: Sci. Syst.*, vol. 2. Ann Arbor, MI, USA, 2016, pp. 1–9.

[53] R. P. Bhattacharyya, R. Senanayake, K. Brown, and M. J. Kochenderfer, “Online parameter estimation for human driver behavior prediction,” in *Proc. IEEE Amer. Control Conf.*, 2020, pp. 301–306.

[54] T. Cover and J. Thomas, *Elements of Information Theory*. Germany: Wiley, 2012.

Negar Mehr (Member, IEEE) received the B.S. degree from the Sharif University of Technology and the Ph.D. degree from UC Berkeley, Berkeley, CA, USA, in 2013 and 2019, respectively, both in mechanical engineering.

From 2019 to 2020, she was a Postdoctoral Scholar with Stanford Aeronautics and Astronautics Department. She is an Assistant Professor with the Aerospace Engineering Department University of Illinois Urbana-Champaign. Her research interest includes learning and control algorithms for interactive robots, robots that can safely and intelligently interact with other agents.

Dr. Mehr is the recipient of the NSF CAREER award in 2022. Her Ph.D. dissertation was awarded the 2020 IEEE ITSS Best Ph.D. Dissertation Award.

Mingyu Wang received the B.S. degree in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 2015, and the M.S. and Ph.D degrees from Stanford University, Stanford, CA, USA, in 2019 and 2022, respectively, both in mechanical engineering.

Her research interests include game-theoretic planning, optimization, and control, especially for autonomous cars interacting with other human-driven or autonomous cars.

Maulik Bhatt received the interdisciplinary dual degree (bachelor’s in aerospace engineering + master’s in systems and control engineering) from the Indian Institute of Technology Bombay, Mumbai, India, in 2021. He is working toward the Ph.D. degree in aerospace engineering at the UIUC.

His research interests include leveraging game theory, stochastic control, and machine learning to enable robotic multi-agent interactions and safe motion planning. During his undergraduate, he worked in the area of state estimation using geometric control and variational principles.

Mac Schwager (Member, IEEE) received the B.S. degree, in 2000, from Stanford University, Stanford, CA, USA, and the M.S. and Ph.D. degrees in mechanical engineering from Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2005 and 2009, respectively.

From 2010 to 2012, he was a Postdoctoral Researcher working jointly with the GRASP lab, University of Pennsylvania, and CSAIL, and was an Assistant Professor with Boston University from 2012 to 2015. He is currently an Associate Professor with the Aeronautics and Astronautics Department at Stanford University. His research interests include distributed algorithms for control, perception, and learning in groups of robots, and models of cooperation and competition in groups of engineered and natural agents.

He is the recipient of the NSF CAREER award in 2014, the DARPA YFA in 2018, a Google faculty research award in 2018, and the IROS Toshio Fukuda Young Professional Award in 2019.