General-Sum Multi-Agent Continuous Inverse Optimal Control

Neumeyer, Christian; Oliehoek, Frans A.; Gavrila, Dariu M.

DOI
10.1109/LRA.2021.3060411

Publication date
2021

Document Version
Final published version

Published in
IEEE Robotics and Automation Letters

General-Sum Multi-Agent Continuous Inverse Optimal Control

Christian Neumeyer, Frans A. Oliehoek, and Dariu M. Gavrila, *Member, IEEE*

***Abstract*—Modeling possible future outcomes of robot-human interactions is of importance in the intelligent vehicle and mobile robotics domains. Knowing the reward function that explains the observed behavior of a human agent is advantageous for modeling the behavior with Markov Decision Processes (MDPs). However, learning the rewards that determine the observed actions from data is complicated by interactions. We present a novel inverse reinforcement learning (IRL) algorithm that can infer the reward function in multi-agent interactive scenarios. In particular, the agents may act boundedly rational (i.e., sub-optimal), a characteristic that is typical for human decision making. Additionally, every agent optimizes its own reward function which makes it possible to address non-cooperative setups. In contrast to other methods, the algorithm does not rely on reinforcement learning during inference of the parameters of the reward function. We demonstrate that our proposed method accurately infers the ground truth reward function in two-agent interactive experiments.[1]**

***Index Terms*—Inverse reinforcement learning, learning from demonstration, reinforcement learning.**

I. Introduction

PREDICTING the future behaviour of agents (e.g., humans, robots) is essential when deploying autonomous robots in environments shared with humans (indoors, outdoors, traffic). The interactions between the agents make this problem particularly challenging.

One significant area of research uses planning based methods based on MDPs to describe (human) agents that interact with their environment and other agents [1]–[7] (see [8] for an overview with a focus on human motion trajectory prediction). In particular, the agents maximize a reward (avoid collision, be considerate) that may transfer to many different settings. Given a reward function, a predictor (policy) can be trained inside a simulator. We can create many environmental configurations that the policy learns to handle before being deployed in the real world. However, humans do not act optimally when steering a car or walking across a street, so we focus on the maximum-entropy framework [9]. A vital issue is inferring the reward function from observation data. These algorithms are often referred to as inverse reinforcement learning (IRL) [9] or inverse optimal control (IOC) [10].

Previous work on multi-agent maximum-entropy inverse reinforcement learning (MaxEntIRL) relies on reinforcement learning (RL) or MCMC sampling [1], [2], [11] as an additional component of the overall algorithm. RL/MCMC can be hard to optimize for multi-agent setups. Therefore, we will focus on approaches that estimate the reward function without RL/MCMC sampling.

[10], [12] established an efficient single-agent MaxEntIRL algorithm that infers a single agent's reward function, assuming that the agent acts according to the MaxEnt framework.

We present a multi-agent formulation of [10], [12] that can infer the reward function of a diverse set of agents without assuming a cooperative reward or instant communication. The algorithm retains important properties of [10], [12] in that it can deal with sub-optimal demonstrations (agents maximize reward and entropy of policy) and does not rely on a complex RL/MCMC algorithm during reward inference. The main idea is to approximate the reward function with a second-order Taylor expansion and to linearize the dynamics at the observed demonstration data. This results in a formulation similar to that of linear-quadratic games (see [13]) where we obtain an analytical solution for the Nash equilibrium that the agents prefer to play. This approximation simplifies the computations drastically compared to other MA-IRL approaches as we do not search for globally optimal Nash equilibria far away from the observed data during the reward inference.

II. Related Work

The literature on inverse reinforcement learning is extensive. As such, we will focus on work that is most relevant to ours.

Multiple algorithms have been proposed for inverse reinforcement learning in multi-agent settings [1], [2], [11], [14]–[19]. Both [14] and [15] extend the single-agent IRL algorithm of [20] to the multi-agent setting. [14] assumes that the problem can be described in terms of a centralized controller and a weighted

Manuscript received October 15, 2020; accepted January 21, 2021. Date of publication February 18, 2021; date of current version March 23, 2021. This work was supported in part by the German Federal Ministry of Economics, and Energy under the @City project under Grant 19 A 17015 A, in part by the European Research Council (ERC) under the European Union's Horizon 2020 Research, and Innovation Programme under Grant Agreement under Grant 758824 —INFLUENCE. This letter was recommended for publication by Associate Editor S. Chernova and Editor D. Kulic upon evaluation of the reviewers' comments. *(Corresponding author: Christian Neumeyer.)*

Christian Neumeyer is with the Intelligent Vehicles Group, TU Delft, 2628 CD Delft, The Netherlands, and also with the Environment Perception Group, Mercedes-Benz AG, 70569 Stuttgart, Germany (e-mail: c.muench@tudelft.nl).

Frans A. Oliehoek is with the Interactive Intelligence Group, TU Delft, 2628 CD Delft, The Netherlands (e-mail: F.A.Oliehoek@tudelft.nl).

Dariu M. Gavrila A. Oliehoek is with the Intelligent Vehicles Group, TU Delft, 2628 CD Delft, The Netherlands (e-mail: D.M.Gavrila@tudelft.nl).

Digital Object Identifier 10.1109/LRA.2021.3060411

[1][Online]. Available: github.com/neumeyerc/GSCIOC

cooperative reward function. In particular, there is no interaction between the agents. In contrast to that, [15] considers the non-cooperative setting but assumes that other agents' policies are known in advance. [19] considers MA-IRL for different equilibria types that the agents agree on beforehand (e.g., Nash equilibrium). They derive convex optimization algorithms to find these equilibira but assume that all agents' policies are known beforehand. Additionally, the algorithms do not scale to continuous states and actions. [1], [2] consider the interaction of a mobile robot with people walking around in an indoor environment. The problem formulation assumes a cooperative setup with a centralized controller. [16] tackle the problem of a robot learning to help a human achieve a task, i.e., a cooperative setup. The problem formulation assumes that the human knows the underlying reward function while the robot does not. This results in a game where the human may perform educational demonstrations instead of expert demonstrations to increase the information content about the reward function. The problem formulation is therefore different to that of standard MA-IRL. [18] applies the deep RL algorithm introduced by [21] to modelling the trajectories of people indoor. While the approach is decentralized, it relies on discrete states and actions and (approximate) value iteration. [11] extends the adversarial inverse reinforcement learning (AIRL) [22] algorithm to the multi-agent setting. The algorithm alternates between training a policy (generator) through RL and updating the reward (discriminator) through binary logistic regression.

An essential aspect of any IRL method is the ambiguity of the reward function. A set of demonstrations can correspond to an infinite number of reward functions. MaxEntIRL [9] tackles this by maximizing the entropy of the policy of each agent. This also allows us to incorporate sub-optimal demonstrations naturally. The most daunting task in MaxEntIRL for high-dimensional continuous action and state spaces (i.e., no dynamic programming) is the derivation of the partition function $Z = \int p(\tau) \exp(r(\tau))$. [1], [2] approximate the partition function using MCMC sampling. [11], [22] alternate between RL and updating the reward function where the partition function is approximated with policy rollouts. In general, this corresponds to solving the full reinforcement learning problem in an inner loop of the inverse reinforcement learning algorithm [23].

[10], [12], [24] avoid the expensive and often-times hard to optimize RL/ MCMC sampling step. [24] approximate the distribution over trajectories with weighted sums of delta-functions representing the observed data points, optimizing the data likelihood by gradient ascent. A more advanced algorithm — which we will use in this paper — is the use of the Laplace approximation [25] (second-order Taylor expansion) around the observed data points that models the curvature of the reward function [10], [12]. The Laplace approximation has been used in multi-agent settings before [4], [7]. Although, [4], [7] consider multi-agent interactions in so-called Stackelberg games for their prediction algorithms, they infer the rewards from real-world data using the single-agent CIOC algorithm [10]. Other agents are reduced to dynamic obstacles simplifying the reward inference (i.e., non-reacting).

III. Background

We are considering a N-agent stochastic game with shared states x_t, agent specific actions u_{kt} ($k \in (1, \ldots, N)$ - agent index), agent specific rewards $r_k(x_t, u_t)$ and stochastic transitions $p(x_t|u_t, x_{t-1})$ to the state x_t given the actions $u_t = [u_{1t}, \ldots, u_{kt}]$ and state x_{t-1}. In general, both the transitions and the reward function depend on the actions of all agents.[2]

Also, the rewards of the agents are discounted with a discount factor γ. In the following, we give an overview of the most important formulas for the single-agent case. These will translate to the N-agent setting naturally.

Given a state x_{t-1} at time-step $t-1$ and an action u_t, an agent will transition to the next state according to the stochastic environment transitions $p(x_t|x_{t-1}, u_t)$.[3] The agent will also receive a reward $r(x_t, u_t)$ depending on the action u_t and the state x_t that the environment (including the agent) transitions to. In this work, we assume that an agent does not act fully rationally and may choose sub-optimal actions. A natural description of this type of bounded rationality is the maximum-entropy (MaxEnt) framework [9] which can be used to describe the sub-optimal decisions of humans (e.g., [26]). In the MaxEnt framework, a trajectory is sampled from a probability distribution given by

$$p(\tau) \sim \Pi_t p(x_t|x_{t-1}, u_t) \exp(r(x_t, u_t)) \tag{1}$$

with the trajectory τ corresponding to the sequence of actions u_t and states x_t over multiple time-steps t. One can show that a policy that leads to (1) can be simplified as follows [27].

$$\pi_t(u_t|x_{t-1}) = \frac{\exp(Q_t(x_{t-1}, u_t))}{\int \exp(Q_t(x_{t-1}, u'_t) du'_t)} \tag{2}$$

The policy $\pi_t(u_t|x_{t-1})$ of an agent, i.e., the conditional probability density function that describes the most likely actions u_t that an agent takes given its current state x_{t-1} depends on the Q-function $Q_t(x_{t-1}, u_t)$. The Q-function may be derived by performing dynamic programming, iterating over the soft-Bellman equation until convergence.

$$Q_t(x_{t-1}, u_t) = \int p(x_t|u_t, x_{t-1}) \left(r(x_t, u_t) + \gamma V_{t+1}(x_t)\right) dx_t \tag{3}$$

The second term inside the first integral is the value function.

$$V_{t+1}(x_t) = \log \int \exp(Q_{t+1}(x_t, u_{t+1})) du_{t+1} \tag{4}$$

The reason we refer to (3) as the soft-Bellman equation is the soft-maximization operator $\log \int \exp$. In contrast, the standard Bellman equation employs the "hard" maximization operator $\max$. A connection can be established by scaling the reward function and considering the limit of $\lim_{\alpha \to 0}(\frac{1}{\alpha} r)$ which will recover the "hard" maximization in the soft-Bellman equation and a policy that satisfies the standard Bellman equation given the unscaled reward function. Please refer to the tutorial on

[2] A fully cooperative reward is an example where one agent may receive a reward for an action that another agent executes.

[3] We follow the notation by [10] in which the action is indexed with the stage *to which* it takes us.

maximum-entropy reinforcement learning and its connection to probabilistic inference [27] for a thorough derivation of the soft-Bellman equation and its connection to (1).

Solving the Bellman equation for high-dimensional continuous state and action spaces is in general intractable. Though for linear dynamics and quadratic reward functions the value function, Q-function and policy can be derived analytically. Following this insight, [10], [12] developed an algorithm that uses the so-called Laplace approximation [25] that deals with the difficulty of calculating the integrals in (3) and (4) by approximating the reward function with a second-order Taylor expansion and linearizing the dynamics.

$$x_t \approx A_t x_{t-1} + B_t u_t$$

$$r(x_t, u_t) \approx r_t + \frac{1}{2} u_t^T \tilde{H}_t u_t + u_t^T \tilde{g}_t + \frac{1}{2} x_t^T \hat{H}_t x_t + x_t^T \hat{g}_t$$

The Bellman equations are solved in a recursive manner starting from the last time-step T and going back to the first time-step. The resulting policy is a unimodal Gaussian distribution.

$$\pi_t(u_t|x_{t-1}) \sim \exp\left((\mu_t - u_t)^T \Sigma_t^{-1} (\mu_t - u_t)\right)$$

Given demonstration data, $(x_t, u_t) \in \tau$ (τ - trajectory) we are interested in inferring the reward parameters θ of a reward function $R_\theta(x_t, u_t)$. We can do so by maximizing the likelihood of the data.

$$\theta^* = \arg\max_\theta \sum_t \ln \pi_{\theta,t}(u_t|x_{t-1})$$

The algorithm is referred to as continuous inverse optimal control (CIOC). We will discuss the mathematics of the multi-agent version in length in the following sections. The single-agent version is explained in detail in [10].

IV. Contributions

We will show how to extend CIOC to multi-agent settings. We will refer to this algorithm as general-sum multi-agent continuous inverse optimal control (GS-CIOC). Our method

- learns the reward functions of a diverse set of agents. It neither considers the other agents as dynamic obstacles, nor does it assume instant communication between agents.
- allows us to choose different reward functions for each agent.
- accounts for variation in the decisions of an agent because it belongs to the family of maximum-entropy algorithms.

In particular,

- we extend the continuous inverse optimal control (CIOC) [10] algorithm to the general-sum N-agent setting.
- we verify the algorithm on simulations and show its usefulness.

V. General-Sum Multi-Agent Continuous Inverse Optimal Control

We extend CIOC to the N-agent setting where each agent may receive a different reward. A major difference to the derivation of CIOC is that the environment transitions are not deterministic anymore. We assume that the other agents are part of the environment and act according to a stochastic policy. A major advantage of CIOC and its extension is the relative ease of inferring the reward parameters from demonstrations. We can backpropagate the gradients directly through the policy, eliminating the need to run a multi-agent reinforcement learning algorithm every time we update the reward parameters.

We will present two algorithms that are interconnected. The first is GS-CIOC (algorithm 1) that returns policies for quadratic rewards and linear environment transitions. The other is the reward inference algorithm 2 that uses GS-CIOC for obtaining local policy approximations around observed real-world data for any reward functions and transitions. Given the policy approximations, the algorithm infers the parameters θ of a reward function R_θ. While the approximations may lead to biased results, we would like to point out that non-linear reward functions and dynamics can be used with this algorithm.

N-Agent Soft-Bellman Equation

We illustrated how an agent might choose its actions in the maximum-entropy framework. Next, we want to investigate how we can deal with the presence of other agents. We assume that the other agents are part of the environment, similar to the multi-agent setting in interacting partially observable Markov decision processes (POMDPs) as described by [28]. We show how we may describe the N-agent problem from the perspective of one agent. Let $u_t = [u_{1t}, \ldots, u_{Nt}]$ be the actions of agent 1 through N. One issue is that the reward function of agent 1 may depend on the actions of another agent $r_1(x_t, u_t)$. Therefore, we introduce a state variable $\tilde{x}_t = [x_t, u_{-1t}]$ that incorporates the actions of other agents except agent 1 with $u_{-1t} = [u_{2t}, \ldots, u_{Nt}]$. The corresponding stochastic environment transitions are $p(\tilde{x}_t|\tilde{x}_{t-1}, u_{1t})$.

We assume the soft-Bellman equation for agent 1 (other agents analogous) to be as follows

$$\tilde{Q}_{1t}(\tilde{x}_{t-1}, u_{1t}) = \int p(\tilde{x}_t|\tilde{x}_{t-1}, u_{1t}) \left(\tilde{r}_1(\tilde{x}_t, u_{1t}) + \gamma \tilde{V}_1(\tilde{x}_t)\right) d\tilde{x}_t \quad (5)$$

$$\tilde{V}_{1t+1}(\tilde{x}_t) = \log \int \exp(\tilde{Q}_{1t+1}(\tilde{x}_t, u_{1t+1})) du_{1t+1} \quad (6)$$

With $\tilde{Q}_{1t}(\tilde{x}_{t-1}, u_{1t}) := Q_{1t}(x_{t-1}, u_{-1t-1}, u_{1t})$ and $\tilde{r}(\tilde{x}_t, u_{1t}) := r(x_t, u_{-1t}, u_{1t})$. Reformulating the environment transitions in terms of the policies of the other agents we get

$$p(\tilde{x}_t|\tilde{x}_{t-1}, u_{1t}) \quad (7)$$

$$:= p(x_t, u_{-1t}|x_{t-1}, u_{-1t-1}, u_{1t}) \quad (8)$$

$$= p(x_t, u_{-1t}|x_{t-1}, u_{1t}) \quad (9)$$

$$= p(x_t|x_{t-1}, u_{1t}, u_{-1t}) p(u_{-1t}|x_{t-1}, u_{1t}) \quad (10)$$

$$= p(x_t|x_{t-1}, u_t) \Pi_{k=2}^N p(u_{kt}|x_{t-1}, u_{-kt}) \quad (11)$$

$$= p(x_t|x_{t-1}, u_t) \Pi_{k=2}^N p(u_{kt}|x_{t-1}) \quad (12)$$

$$= p(x_t|x_{t-1}, u_t) \Pi_{k=2}^N \pi_{k,t}(u_{kt}|x_{t-1}) \quad (13)$$

Algorithm 1: GS-CIOC

Input: Reference trajectory $\tau^* = (x_0^*, u_{k1}^*, \ldots, x_T^*)$, $k \in \{1, \ldots, N\}$ and reward functions r_1,..., r_N
Taylor expansion (17) of r_1,..., r_N along τ^*
Linearization of dynamics (23) along τ^*
Initialize value function matrices $\hat{V}_{(kl)T}, \hat{v}_{kT} \leftarrow 0$
for $t \leftarrow T$ **to** 1 **do**
  {Update Gaussian policy:}
  for $k = 1, \ldots, N$ **do**
    $\mu_{kt} \leftarrow$ Determine mean action (28)
    $\tilde{M}_{(kk)t} \leftarrow$ Determine precision matrix (29)
  end for
  {Recompute value function, given updated policy:}
  if $t > 1$ **then**
    $\hat{V}_{(kl)t-1}, \hat{v}_{kt-1} \leftarrow$ Value recursion (31)
  end if
end for
Return: Policies π_1, ..., π_N (30)

In particular, the u_{-1t-1} dependency disappears. The soft-Bellman equation can now be reformulated as

$$Q_{1t}(x_{t-1}, u_{-1t-1}, u_{1t}) = \int p(x_t|x_{t-1}, u_t)\Pi_{k=2}^N \pi_{k,t}(u_{kt}|x_{t-1}) \left(r_1(x_t, u_t) + \gamma \tilde{V}_{1t+1}(x_t, u_{-1t})\right) du_{-1t} dx_t \quad (14)$$

As we can see, the Q-function does not depend on u_{-1t-1}. Therefore, we can drop the dependency in both the Q and the value function.

$$Q_{1t}(x_{t-1}, u_{1t}) = \int p(x_t|x_{t-1}, u_t)\Pi_{k=2}^N \pi_{k,t}(u_{kt}|x_{t-1}) \times \left(r_1(x_t, u_t) + \gamma V_{1t+1}(x_t)\right) du_{-1t} dx_t \quad (15)$$

$$V_{1t+1}(x_t) = \log \int \exp(Q_{1t+1}(x_t, u_{1t+1})) du_{1t+1} \quad (16)$$

From here on we will assume the environment transitions $p(x_t|x_{t-1}, u_t)$ to be deterministic. In other words, if we know the current state and are given the actions of all agents, there is no uncertainty left which state we transition to next. Though, we never assume that the actions of the other agents are given. Their policies are stochastic, and the actions are not revealed before they are executed (see 15). Additionally, we will only consider finite horizon problems, i.e., each agent will collect rewards for a limited amount of time. Furthermore, we set the discount factor to $\gamma = 1$.

Value Recursion Algorithm

The procedure that we obtain in this section is illustrated in algorithm 1. We take a reference trajectory τ^* - a sequence of states x_t^* and actions u_{kt}^* - of each agent $k \in (1, \ldots, N)$ and approximate the reward function r_k close to the reference trajectory τ. This will allow us to derive a local policy approximation π_k — an approximation that works best if the agent stays close to the reference trajectory — by working our way from the end of the reference trajectory to the beginning calculating the value function $V_k(x_t)$ for each time-step. In other words, the formulas are recursive.

We sketch the derivation of algorithm 1 starting at the final time-step (the horizon) of the reference trajectory τ. All formulas will be presented from the perspective of agent 1. Though, we can easily obtain the formulas for any agent k by swapping the indices $(1 \longleftrightarrow k)$.

First, we introduce the approximations that are fundamentally important to solve the soft-Bellman equation analytically. The reward function is assumed to be a second-order polynomial in the states and actions. If this is not the case, we approximate the reward function with a Taylor expansion.[4]

$$\bar{r}_1(\bar{x}_T, \bar{u}_T) \approx \bar{r}_{1T} + \bar{x}_T^T \left(\frac{1}{2}\hat{H}_T\right) \bar{x}_T + \bar{u}_T^T \left(\frac{1}{2}\tilde{H}_T\right) \bar{u}_T + \bar{u}_T^T \tilde{g}_T + \bar{x}_T^T \hat{g}_T \quad (17)$$

$$\bar{x}_T = x_T - x_T^*, \qquad \bar{u}_T = u_T - u_T^* \quad (18)$$

$$\bar{r}_1(\bar{x}_T, \bar{u}_T) := r_1(\bar{x}_T + x_T^*, \bar{u}_T + u_T^*) \quad (19)$$

where x^* and u^* refer to the fixed reference trajectory. The state $\bar{x}_T = [\bar{x}_{1T}, \ldots, \bar{x}_{NT}]$ is split into the agent-specific sub-states which are directly controlled by each agent. H and g refer to the Hessians and gradients w.r.t the states and actions of all agents.

$$\hat{g}_{kT} = \frac{\partial \bar{r}_1(0,0)}{\partial \bar{x}_{kT}}, \qquad \tilde{g}_{kT} = \frac{\partial \bar{r}_1(0,0)}{\partial \bar{u}_{kT}} \quad (20)$$

$$\hat{H}_{(kl)T} = \frac{\partial^2 \bar{r}_1(0,0)}{\partial \bar{x}_{kT} \partial \bar{x}_{lT}}, \qquad \tilde{H}_{(kl)T} = \frac{\partial^2 \bar{r}_1(0,0)}{\partial \bar{u}_{kT} \partial \bar{u}_{lT}} \quad (21)$$

The derivatives are evaluated at the reference trajectory throughout the paper. An additional approximation that is necessary is the linearization of the dynamics.

$$\bar{x}_{kT}(\bar{x}_{kT-1}, \bar{u}_{kT}) \approx A_{kT}\bar{x}_{kT-1} + B_{kT}\bar{u}_{kT} \quad (22)$$

$$A_{kT} = \frac{\partial \bar{x}_{kT}(0,0)}{\partial \bar{x}_{kT-1}}, \qquad B_{kT} = \frac{\partial \bar{x}_{kT}(0,0)}{\partial \bar{u}_{kT}} \quad (23)$$

Given the quadratic reward function and the linear dynamics, the Q-function can be calculated as follows

$$\bar{Q}_{1T}(\bar{x}_{T-1}, \bar{u}_{1T}) = \int \bar{p}(\bar{x}_T|\bar{x}_{T-1}, \bar{u}_T) \times \Pi_{k=2}^N \pi_{k,t}(\bar{u}_{kT}|\bar{x}_{T-1}) \bar{r}_1(\bar{x}_T, \bar{u}_T) d\bar{u}_{-1T} d\bar{x}_T \quad (24)$$

with

$$\bar{p}(\bar{x}_T|\bar{x}_{T-1}, \bar{u}_T) := p(\bar{x}_T + x_T^*|\bar{x}_{T-1} + x_{T-1}^*, \bar{u}_T + u_T^*) \quad (25)$$

$$\bar{Q}_{1T}(\bar{x}_{T-1}, \bar{u}_{1T}) := Q_{1T}(\bar{x}_{T-1} + x_{T-1}^*, \bar{u}_{1T} + u_{1T}^*) \quad (26)$$

[4]Here, we stay close to the single-agent LQR derivation in [10] where actions and states separate in the reward function $r(x_t, u_t) = g(x_t) + f(u_t)$. This is also the structure that we assume in the experimental section.

The definitions for the policy $\bar{\pi}$ and the value function $\bar{V}$ further down will follow the same logic. Thanks to the approximations introduced above, the integration is tractable except for the policies of the other agents $\pi_2, \ldots, \pi_N$. We assume the policies to be unimodal Gaussian distributions. We show that this assumption is consistent further down. Given this assumption, the Q-function is a second-order polynomial in the states and actions

$$\bar{Q}_{1T}(\bar{x}_{T-1}, \bar{u}_{1T}) = -\frac{1}{2}(\bar{u}_{1T} - \mu_{1T})^T \tilde{M}_{(11)T}(\bar{u}_{1T} - \mu_{1T}) + f(\bar{x}_{T-1}) \quad (27)$$

where f is a second order polynomial in the state at $T-1$. The variables μ_{1T} and $\tilde{M}_{(11)T}$ are defined as follows

$$\tilde{g}_{1T} + B_{1T}^T \hat{g}_{1T} + \sum_{k=1}^{N} B_{1T}^T \hat{H}_{(1k)T} A_{kT} \bar{x}_{kT-1} + \sum_{k=1}^{N} \tilde{M}_{(1k)T} \mu_{kT} = 0 \quad (28)$$

$$\tilde{M}_{(kl)T} = B_{kT}^T \hat{H}_{(kl)T} B_{lT} + \tilde{H}_{(kl)T} \quad (29)$$

In particular, equation (28) is a system of linear equations in the variables μ_{kT} (replace index 1 by $k \in (1, \ldots, N)$ to obtain other equations) which we can solve (solution omitted for brevity). The solution corresponds to the mean action that each agent chooses in a linear-quadratic game. A similar derivation exists for the classical (non-maximum entropy) Bellman-equation, given quadratic rewards and linear dynamics [13].

Given that $\pi \sim \exp(Q)$ the resulting policy is

$$\bar{\pi}_{1,T}(\bar{u}_{1T}|x_{T-1}) \sim \exp\left(-\frac{1}{2}(\bar{u}_{1T} - \mu_{1T})^T \tilde{M}_{(11)T}(\bar{u}_{1T} - \mu_{1T})\right) \quad (30)$$

with μ_{1T} being the mean and $\tilde{M}_{(11)T}$ the precision matrix of a multivariate normal distribution.

We can conclude that the policy of agent 1 is a Gaussian policy. The same is true for the other agents since we apply the same approximations, i.e., $\pi_2, \ldots, \pi_N$ is a Gaussian policy validating our assumption that this is the case further up. In particular, we would like to emphasize that this follows from the second-order Taylor expansion of the rewards and the linearization of the dynamics and does not constitute an additional approximation.

In the next step, we move the procedure above backwards in time along the reference trajectory, calculating the Q-function (15) the mean actions, the precision matrix for time-step $T-1$. There is only one ingredient missing to evaluate equation (15), namely, the value function at T, which we can derive via (16).

$$\bar{V}_{1T}(\bar{x}_{T-1}) = \bar{x}_{T-1}^T \left(\frac{1}{2}\hat{V}_{T-1}\right) \bar{x}_{T-1} + \bar{x}_{T-1}^T \hat{v}_{T-1} \quad (31)$$

The value function is a second-order polynomial in the states. Thus, the Q-function given by (15) can be derived analytically for time-step $T-1$. Due to space constraints we do not provide the exact make-up of the resulting value function matrices $\hat{V}, \hat{v}$. Though, we discussed all steps that are necessary for their derivation.

The procedure repeats itself until the beginning of the reference trajectory for $T, T-1, \ldots, 1$.

Algorithm 2: GS-CIOC Reward Inference

Input: τ from data, initial θ
repeat
  $\pi_{1,\theta}$,..., $\pi_{N,\theta} \leftarrow$ **GS-CIOC**(τ, R_θ)
  Gradient ascent step on objective (32)
until max iterations or convergence of θ
Return: Reward parameters θ

VI. Recovering Reward Parameters

Until now, we have discussed how to construct a local policy given a set of reference trajectories τ and a reward function (see algorithm 1). Next, we will infer the parameters θ of a reward function R_θ given expert demonstrations. Specifically, given observation data we can infer the reward parameters by maximizing the log-likelihood of the observed data

$$\theta^* = \arg\max_\theta \frac{1}{|\tau|} \sum_{\tau,(u,x)\in\tau} \ln p_\theta(x_{0:T-1}, u_{1:T}) = \arg\max_\theta \frac{1}{|\tau|} \sum_{\tau,(u,x)\in\tau} \sum_{k,t} \ln \pi_{k,t,\theta}(u_{kt}|x_{t-1}) \quad (32)$$

θ refers to the reward parameters and $|\tau|$ to the number of trajectories τ in the data set. $\pi_{1,\theta}$,..., $\pi_{N,\theta} \leftarrow$ GS-CIOC(τ, R_θ) correspond to the policies of agent 1 through N and are calculated with GS-CIOC. Indeed, it is possible to perform backpropagation through the entire GS-CIOC algorithm. The procedure is illustrated in algorithm 2. We optimize the objective (32) using gradient ascent. Overall, the approach is similar to the single-agent reward inference of CIOC.

Depending on the reward function, reward parameter initialization and the training data the algorithm may get numerically unstable if the precision matrix in (30) is not positive definite temporarily. It is possible to prevent this using the augmented Lagrangian method as outlined in [10].

Relationship to Linear Quadratic Game

We can scale the reward function with a "temperature" parameter $r \to \frac{1}{\alpha} r$. For $\alpha \to 0$ the entropy of the policies will collapse, and the soft-Bellman equations will converge to the standard Bellman equations [27]. In particular, the N-agent game solution in the previous section will resemble that of the deterministic LQ-game as described by [13]. Therefore, it is reasonable to ask whether GS-CIOC can recover the ground-truth reward of an LQ-game demonstration.

Scaled ground-truth reward maximizes log-likelihood First, we observe that (28) is independent of α (g, H, M scale linearly with $\frac{1}{\alpha}$) at time-step T. The same is true for all time-steps. Hence, μ_{kt} is independent of α and will correspond to the

LQ-game Nash equilibrium solution ($\alpha \to 0$) given the ground-truth reward. Therefore, transforming the ground-truth reward $r \to \frac{1}{\alpha} r + c; \alpha \in R^+, c \in R$ will maximize the log-likelihood in (32) as the GS-CIOC mean actions are identical to the LQ-game demonstrations.[5]

VII. Experiments

We show in this section that GS-CIOC infers useful reward parameters. Additionally, we show that GS-CIOC has a clear advantage over its single-agent counterpart CIOC. Indeed, previous work [4], [7] uses of CIOC to infer reward parameters to describe the interaction of vehicles (other agent assumed as dynamic obstacle during reward inference). We demonstrate why this can be a sensible approach and point out the limitations, i.e., where GS-CIOC is at a significant advantage. We used JAX [30] for the implementation of GS-CIOC.

Evaluation Setup

The purpose of GS-CIOC is to infer reward parameters that explain observed multi-agent behaviour. Ideally, we know the reward parameters beforehand to judge the performance of the algorithm better. Therefore, we will define several setups where we know the underlying reward function. A ground-truth reward function is not enough, since GS-CIOC needs actual data to reason about likely reward parameters. We create the data with a multi-agent RL algorithm as described in algorithm 3. The algorithm derives the policies of the agents, and the data corresponds to roll-outs from these policies. GS-CIOC succeeds if it infers reward parameters given the data that are close to the ground-truth.

A difficulty arises when the number of reward parameters increases. Alternative parameter values may explain the data equally well. It is often not obvious why an alternative configuration may be reasonable even though it seems to deviate significantly from the ground-truth. Therefore, we also look at the policies that result from the inferred parameters. Again, algorithm 3 is applied to the reward that GS-CIOC deems most likely. We can evaluate the log-likelihood on some test data (from ground-truth policy roll-outs) to see if GS-CIOC inferred a reasonable parameter configuration.

$$\mathcal{L}_k = \frac{1}{T} \sum_t \log \pi_{k,t}(u_{kt} | x_{t-1}) \tag{33}$$

Crossing Scenario

It is not uncommon to use single-agent IRL algorithms in multi-agent scenarios in the intelligent vehicles domain [4], [7]. This approach can be feasible as we can see in the following experiment. We assume that two agents move on a one-dimensional line. The overall configuration imitates the intersection of two roads. One agent moves from the left to the right towards its goal position and the other agent moves from the bottom to the top towards its goal position. Both need to pass the intersection point where they may collide. We use the following ordinary differential equation to describe their movement

$$\frac{d}{dt}[s, v] = [v, a] \tag{34}$$

Where s is the position, v is the velocity and a is the acceleration. The acceleration corresponds to the action of the agent u and the position and velocity correspond to the state x. Each agent receives rewards that determine the preferred movement speed, acceleration/ deceleration, goal position and interaction.

- Quadratic acceleration reward that punishes acceleration
- Quadratic reward that punishes velocities other than zero
- Quadratic reward that rewards being close to a goal position
- Gaussian reward that punishes the agents for being close to each other (at intersection point) - i_k

The reward function for each agent is the linear combination of the rewards described above.

We create a training and testing dataset (11 time-steps for each roll-out) and infer the parameter i_k that the interaction reward (Gaussian) is scaled with. The remaining parameters are fixed to the ground-truth. We are only interested in the ability of GS-CIOC and CIOC to reason about interactions. In particular, CIOC infers hard to interpret reward parameter configurations if no parameter is fixed (see the results of the next experiment in table II).

We find that the interaction reward of our setup is critical, i.e., it cannot be ignored to model the observed behaviours. The

Algorithm 3: Alternating Soft-Value Iteration

Initialize policy matrices π_1, π_2 as (discrete) uniform distributions
Initialize value matrices V_1, V_2 as all zeros arrays
ϵ is chosen so that policy updates below converge
repeat
  for k $\leftarrow \{1, 2\}$ **do**
    for M **do**
      Execute soft-Bellman equation (15) to update value matrices
    end for
    Averaging of new and previous policy
    $\pi_{k,new} = \exp(Q_k) / \sum_u \exp(Q_k)$
    $\pi_k \leftarrow \epsilon \pi_{k,new} + (1 - \epsilon)\pi_k, \quad \epsilon \in (0, 1]$
  end for
until convergence of π_1, π_2
Return: Policy matrices π_1 and π_2

TABLE I
Crossing Scenario

		GT	GS-CIOC	CIOC	GT int=0
Reward-Parameters	i_1	-8.	**-7.1**	-6.1	0.
	i_2	-8.	**-7.3**	-6.3	0.
Log-Likelihood	$\mathcal{L}_1$	-1.581	**-1.586**	-1.616	-1.926
	$\mathcal{L}_2$	-1.595	**-1.595**	-1.611	-1.934

[5]This does not mean that GS-CIOC will recover the ground-truth reward from a single demonstration. The number of reward parameters is likely to lead to an ill-defined problem (a general property of IRL algorithms). Additionally, there may be transformations of the reward function that will lead to the same policy (reward shaping, as discussed in [29]).

TABLE II
ATTRACTION SCENARIO

Quadratic	a_1	v_1	a_2	v_2	i_1	i_2	$\mathcal{L}_1 + \mathcal{L}_2$
GT	*10.*	*10.*	*10.*	*10.*	*-10.*	*25.*	*-3.15*
GS-CIOC	**10.3**	**9.7**	**10.2**	**10.3**	**-9.8**	**23.9**	**-3.15**
CIOC	9.3	3.4	7.6	18.0	-2.5	0.	-3.65
Gaussian							
GT	*10.*	*10.*	*10.*	*10.*	*-10.*	*25.*	*-3.36*
GS-CIOC	**10.0**	**11.4**	**8.4**	**13.0**	**-8.0**	**34.3**	**-3.37**
CIOC	7.9	5.4	8.3	15.0	-3.7	0.	-3.68

table below lists the results of CIOC and GS-CIOC for the case where we fix all reward parameters except the interaction reward parameter during reward inference. CIOC is surprisingly close to the performance of GS-CIOC. In particular, both methods provide a clear benefit to the baseline where no interaction reward is present (all other parameters correspond to the ground truth).

i_k are the reward parameters that GS-CIOC and CIOC try to infer from demonstration data. $\mathcal{L}_k$ is the log-likelihood as defined in (33). GT corresponds to the ground-truth, i.e., algorithms close to values listed as GT perform best. GT int $= 0$ is an additional baseline where the interaction is set to zero. This baseline demonstrates the significance of the interaction reward in the experiments.

The reader may wonder why GS-CIOC did not derive the exact ground-truth interaction reward parameter. For one, the value iteration algorithm discretizes states and actions, whereas GS-CIOC reasons in terms of continuous states and actions. This can result in a biased estimate. Additionally, GS-CIOC relies on a second-order approximation of the reward function, which introduces a systematic bias for non-quadratic rewards (same for CIOC). We discuss this further in the following experiment.

Attraction Scenario

While it is nice to show that GS-CIOC has an edge on CIOC the improvement appears minor. The following experiments provide an intuition when the benefits are certain to show up. We consider two agents that live on a one-dimensional line. In contrast to the previous experiment, both agents move on the same line (no intersection of two different lines). The dynamics model is the same as in (34). The rewards are as follows.

Non-interaction rewards

- Quadratic acceleration reward that punishes acceleration - a_k (the reward parameter this reward is scaled with)
- Quadratic reward punishing velocities $\neq$ zero - v_k

Interaction rewards

- Quadratic (or Gaussian) reward that rewards agent 2 for being close to agent 1 - i_1
- Quadratic reward that rewards agent 1 if agent 2 is close to a certain goal position. Agent 2 does not receive the reward. - i_2

Agent 2 is attracted to agent 1 and moves to the position of agent 1. Agent 1 on the other hand gets rewarded if agent 2 is at a certain position. Therefore, agent 1 will move to that position, and agent 2 follows along. In other words, agent 1 guides agent 2.

We create training and testing data from the ground-truth reward function (4 time-steps for each roll-out) and infer the reward parameters using GS-CIOC and CIOC. The result can be seen in table II. Given these results it is clear that CIOC is not able to reason about the interaction between the agents. It cannot learn all interaction rewards (CIOC does not produce any gradient for the i_2 parameter during optimization). The intuition is rather simple. CIOC works well in scenarios where other agents can be treated as dynamic obstacles (agent 2 does not change its behaviour no matter what agent 1 does). The assumption was sufficient for the first experiment in this section. Here it is simply wrong. GS-CIOC on the other hand can reason about this type of behaviour and infers reward parameters that are close to the ground-truth. In particular, when choosing all quadratic rewards the parameters are remarkably close. This is to be expected as the second-order Taylor approximation of the reward function is exact. Again, differences remain and are probably due to the discretization of states and actions for the value iteration. Though, choosing a Gaussian interaction reward will result in a biased reward parameter estimate, which we have observed in the crossing experiment as well. Nevertheless, the rewards and log-likelihood indicate that the inferred parameters are useful. The following experiment will illustrate this further.

Repulsion Scenario

We explore another scenario that is identical to the attraction scenario above (with Gaussian interaction reward) except that the attractive interaction reward is changed into a repulsive one (change sign). One agent pushes the other agent to a desired goal position.

As we can see in figure 1, CIOC struggles with properly modelling the interaction. Agent 2 is supposed to move down to push agent 1 towards a certain goal position. Instead, agent 2 remains standing still when using the reward parameters inferred by CIOC (figure 1 (c)). GS-CIOC on the other hand infers reward parameters that imply a behaviour close to the ground-truth (figure 1 (a)). There are two lessons to take away from the experiments. Single-agent IRL algorithms can be useful in multi-agent scenarios, and it is a good idea to apply such an algorithm and see if it produces reasonable results. But generally speaking, agents will manipulate each other to achieve their goals, resulting in a breakdown of single-agent algorithms such as CIOC. GS-CIOC can fill the gap.

Runtime

Executing algorithm 1 is the computationally most expensive step. Given 500 data points that we evaluate in parallel: On an Intel i7-7820X using one core it takes around 160 ms and 40 ms on a Nvidia TITAN Xp for the attraction and repulsion experiments. Though, we do not consider our implementation fully optimized. Overall, we expect a similar scaling behaviour for time, state and action dimension as CIOC, which has been applied to real-world use-cases before [4], [7], [10], [12]. Other than CIOC, the complexity will also increase with the faculty of the number of agents, i.e., in step with the number of potential agent-agent interactions. This may be addressed by pruning

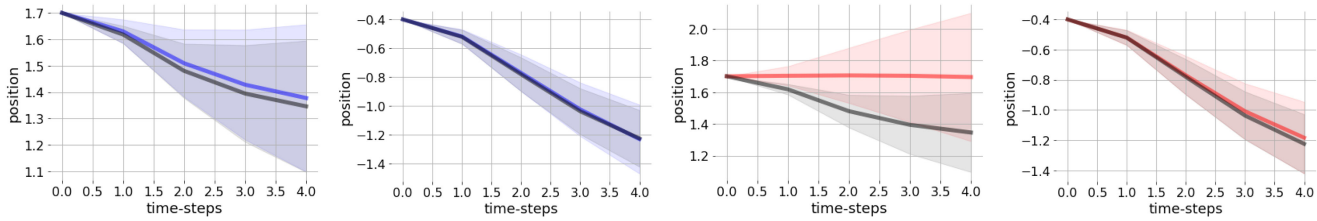

Fig. 1. Repulsion scenario: (a) Agent 2 moves down to push agent 1 towards the position at -1.5 (b). The curves illustrate the mean path that agent 2 (a) and agent 1 (b) take, whereas the shaded areas correspond to one standard deviation around these paths. Grey corresponds to roll-outs from the ground-truth reward parameters and blue to those inferred by GS-CIOC. (c) and (d) represent the same analysis for CIOC (red). While GS-CIOC infers parameters that result in a policy close to the ground-truth, CIOC cannot reason about the behavior of agent 2 (stands still).

the number of interaction partners by e.g., considering nearest neighbours only.

VIII. Conclusion

We presented a novel algorithm for inferring the reward function in stochastic games with boundedly rational agents efficiently. While the single-agent CIOC algorithm can be useful in some interactive scenarios with limited interaction, we have demonstrated the superiority of multi-agent GS-CIOC for multiple experimental setups where the algorithm recovered a reward function close to the ground-truth.

In future work, we want to investigate alternatives to the Laplace approximation to improve the performance for non-quadratic rewards and probe the performance of GS-CIOC on multi-agent interaction scenarios similar to [4], [7].

References

[1] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The Int. J. Robot. Res.*, 2016.

[2] M. Pfeiffer, U. Schwesinger, H. Sommer, E. Galceran, and R. Siegwart, "Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models," in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 2016, pp. 2096–2101.

[3] W. C. Ma, D. A. Huang, N. Lee, and K. M. Kitani, "Forecasting interactive dynamics of pedestrians with fictitious play," in *Proc. 30th IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 4636–4644.

[4] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for cars that coordinate with people: Leveraging effects on human actions for planning and active information gathering over human internal state," *Auton. Robot.*, vol. 42, no. 7, pp. 1405–1426, 2018.

[5] A. Rudenko, L. Palmieri, and K. O. Arras, "Joint long-term prediction of human motion using a planning-based social force approach," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1–7.

[6] C. Muench and D. M. Gavrila, "Composable Q-functions for pedestrian car interactions," in *Proc. IEEE Intelli. Vehi. Sympo.*, 2019, pp. 905–912.

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

[8] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The Int J. Robot. Res.*, vol. 39, no. 8, pp. 895–935, 2020.

[9] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. Conf. Artif. Intelli.*, 2008, pp. 1433–1438.

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

[11] L. Yu, J. Song, and S. Ermon, "Multi-agent adversarial inverse reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, 2019, pp. 7194–7201.

[12] A. D. Dragan and S. S. Srinivasa, "Formalizing assistive teleoperation," *Robotics: Sci. Syst.*, vol. 8, pp. 73–80, 2013.

[13] T. Basar and G. J. Olsder, "Dynamic Noncooperative Game Theory, *2nd Edition.*" Philadelphia, PA, USA: SIAM, 1999, vol. 23.

[14] 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. Applicat.*, 2010, pp. 395–400.

[15] T. S. Reddy, V. Gopikrishna, G. Zaruba, and M. Huber, "Inverse reinforcement learning for decentralized non-cooperative multiagent systems," in *Proc. IEEE Int. Conf. Syst., Man, Cybernet.*, 2012, pp. 1930–1935.

[16] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," in *Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 3909–3917.

[17] X. Wang and D. Klabjan, "Competitive multi-agent inverse reinforcement learning with sub-optimal demonstrations," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 5143–5151.

[18] M. Fahad, Z. Chen, and Y. Guo, "Learning how pedestrians navigate: A deep inverse reinforcement learning approach," *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2018, pp. 819–826.

[19] 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.

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

[21] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *The Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1073–1087, 2017.

[22] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *Proc. Int. Conf. Learn. Representat.*, 2018.

[23] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 49–58.

[24] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," *Robotics: Sci. Syst.*, vol. 8, pp. 193–200, 2013.

[25] P. S. Laplace, "Memoir on the probability of the causes of events," *Stat. Sci.*, vol. 1, no. 3, pp. 364–378, 1986.

[26] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 1–14.

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

[28] P. J. Gmytrasiewicz and P. Doshi, "A framework for sequential planning in multi-agent settings," *J. Artif. Intell. Res.*, vol. 24, pp. 49–79, 2005.

[29] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, vol. 99, 1999, pp. 278–287.

[30] J. Bradbury *et al.*, "JAX: Composable Transformations of Python NumPy Programs," 2018.