TUHH
Hamburg University of Technology

ICS
Institute of Control Systems

Master's thesis

Accommodating Intention Uncertainty in General-Sum Games for Human-Robot Interaction

Author:
Lasse Peters

Supervisor: Patrick Göttsch, M.Sc.

Examiner: 1. Prof. Dr. Herbert Werner
2. Prof. Dr. Zachary Sunberg

22.04.2020

ICS
Institute of Control Systems

Master's thesis
for Mr. Lasse Peters

M — 8 — 2019
Student ID: 21486931

Study Course: IMPMEC

Title:
Accommodating Intention Uncertainty in General-Sum Games for Human-Robot Interaction

Project Description:
Operating in a shared environment with humans poses a challenging problem for autonomous systems since it involves reasoning about closed-loop response dynamics of multiple agents. The theoretical framework of game theory provides a wide range of methods for modeling and solving multi-agent interaction problems. Recent research has put forth a variety of approaches for utilizing game-theoretic reasoning for human-robot interaction tasks, [1, 2]. However, these methods are based on the modeling assumption that the planner has perfect information about the objective of other agents. The goal of this work is to design an algorithm that allows this assumption to be relaxed. After closely reviewing the existing approaches, an algorithmic extension is to be developed that explicitly reasons about uncertainty in the intention of other players. The performance of this algorithm is to be evaluated in simulation of a general-sum game and compared to a state-of-the-art baseline. Metrics for evaluation include computational tractability and the algorithms ability to meet specified control objectives.

Tasks:

1. Literature review
2. Selection of a suitable baseline for algorithmic extension
3. Implementation of a state-of-the-art game solver
4. Design and implementation of an algorithm for reasoning about uncertain objectives of other players in a general-sum game scenario
5. Evaluation and discussion

References:

[1] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[2] 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", *ArXiv:1909.04694*, 2019.

Supervisor: Patrick Göttsch, M.Sc.
Examiner: 1. Prof. Dr. Herbert Werner
2. Prof. Dr. Zachary Sunberg

External: University of Califonia, Berkeley

Start Date: 16.10.2019
Due Date: 22.04.2020

30.09.2019, Prof. Dr. H. Werner

Hereby I declare that I produced the present work myself only with the help of the indicated aids and sources.

Hamburg, 22.04.2020 Lasse Peters

Abstract

Safe and efficient interaction with humans requires an autonomous agent to understand the effects of its actions on their decisions. These coupled interactions of multiple players with differing objectives are well-described by a general-sum differential game. Recent work has put forth efficient approximations to differential games that admit noncooperative equilibrium solutions at real-time planning rates. However, an application of these methods to human-robot interaction (HRI) is impeded by the fact that, in the current formulation, they do not admit to model an important aspect of such problems, namely, intention uncertainty. This type of uncertainty arises if a robot does not know which equilibrium strategies humans may adapt to achieve their goal (equilibrium uncertainty) or if it has incomplete knowledge of their objectives (objective uncertainty). This work develops an approach for accommodating both types of intention uncertainty in game-theoretic formulations of HRI problems. For this purpose, a particle filtering technique is proposed that estimates the likelihood of possible human behaviors, i.e. their objectives and corresponding equilibrium strategies, from observations of the state. Based on this, planning is performed by choosing the robot strategy from the inferred most likely game solution. An evaluation of this method in simulations of multi-player intersection-driving scenarios shows that the developed approach clearly outperforms a game-theoretic planner that ignores intention uncertainty. By inferring the most likely human behavior, the robot is able to predict trajectories more accurately, and by invoking the strategy from the most likely game solution the robot is able to reduce the cost for *all* players. In addition to this main contribution, this work proposes an approach for fitting objective models to datasets of noise-corrupted partial state observations of multi-player interactions; i.e. the inverse differential game problem. This approach works by numerically inverting the solver proposed in [1] via differentiable programming and thus allows gradient-based optimization of the model parameters. A validation on synthetic datasets of observed multi-player interactions demonstrates that the proposed approach can reliably fit a model that matches the observations and accurately predicts the objective parameters of the model that was used to generate the data.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

DBN dynamic Bayesian network. 15

DDN dynamic decision network. 29

HMM hidden Markov model. 15

HRI human-robot interaction. 1, 5

ILQG iterative linear-quadratic games. 12

ILQR iterative linear-quadratic regular. 12

LQ linear-quadaratic. 4

LTV linear time-varying. 9

MAP maximum a posteriori. 35

POMDP partially observable Markov decision process. 2

SMC sequential Monte Carlo. 17

SUS stochastic universal sampling. 18

Chapter 1

Introduction

In order for a robot to be truly autonomous it needs to be capable of interacting with environments that are shared with other agents. Naturally, these shared environments are manipulated by multiple decision making agents at a time and different agents may pursue different objectives. Interactions in such shared environments are commonly referred to as multi-agent decision making problems. This work focuses on a special subclass of these problems: human-robot interaction (HRI).

Safe and efficient human-robot interaction (HRI) is one of the biggest technical challenges that has been encountered since early attempts at autonomy. A common way of approaching this problem is from an optimal control perspective; that is, by assuming a behavior model for humans and then choosing actions for the autonomous agent that minimize a cost function given this model. To simplify matters, the decision making process is often decoupled by first making an open-loop prediction of the decisions of other agents and subsequently optimizing the ego-agent's decisions based on these *fixed* predictions [2, 3]. However, for any non-trivial interaction scenario, the success of a given agent depends not only on its own actions but also on the decisions of others. By ignoring this dependence, an autonomous agent is incapable of discovering strategies which exploit reactions. Furthermore, trusting a nominal prediction may lead to unexpected or even unsafe behavior of a robot. The latter aspect is particularly important when interacting with humans. To address this problem within the optimal control framework, human reactions may be modeled by hand-specifying a feedback policy [4, 5] or by fitting behavior to data [6–9].

An alternative to specifying a behavior model directly is to model humans as rational agents and specify or learn a cost function that captures their objectives. Given this representation, the explicit behavior model is the solution of an equilibrium problem in which each agent seeks to minimize their respective cost. Formally, this choice of model corresponds to a dynamic game, played out between the robot and the humans. If the state evolves in continuous time — as is commonly the case in robotics — such problems are referred to as differential games. In this framework, the dynamics of the environment are described by a differential equation which depends upon each player's input, and the objectives of all players are expressed by their respective cost functions. By solving such

a game to an appropriate equilibrium concept — e.g. a Nash equilibrium in the case of noncooperative scenarios — one can recover complex interactive behaviors, even from simple and easy-to-understand objective functions [1].

Despite the expressiveness of differential games, the application of differential game theory to practical interaction problems is impeded by a key characteristic of real-world interaction scenarios: presence of uncertainty. This uncertainty may arise from incomplete knowledge of the state, the dynamics, the objectives of other players, and the equilibrium that other players aim for. While the first two aspects are inherent properties of the environment, the latter two are aspects of the intentions and the decision making process. Accommodating this *intention uncertainty* in differential games for HRI is the focus of this work.

1.1 Motivation

For illustration of issues arising from intention uncertainty, consider the example of a real-world interaction between a motorist and a cyclist at an intersection depicted in Figure 1.1.1.

At the initial time (Figure 1.1.1a) each road user may not know the exact objective, e.g. the goal location or the aggressiveness, of the other. Furthermore, even if they knew the other player's objective, the interaction may allow for multiple solutions. For example, if the driver of the car knew that the cyclist wishes to go straight, the problem still may allow for two different solutions: one in which the bicycle passes in front of the car, and one where it passes in the back.

However, as time progresses, they can gather information by observing the behavior of the other. At time $t = 2\,\mathrm{s}$ (Figure 1.1.1b) the driver can conclude that the cyclist is unlikely to pass the car in the front but may still be uncertain about her goal location. Finally, at time $t = 4\,\mathrm{s}$ the goal locations of both players become apparent.

Of course, in this example, both players are human agents who may solve this task with ease. However, if one of the players were replaced with a robot, the sources of uncertainty discussed above pose a challenging problem for autonomous decision making. If a robot in this example made wrong assumption about the intentions of other players, both players may incur a high cost, e.g. if the chosen pair of strategies resulted in a collision. Hence, understanding other players' intentions and considering the associated uncertainty is crucial to safe and efficient interaction.

1.2 Additional Background and Related Work

To put this work in context, this section provides an overview of related work that is concerned with modeling and planning in multi-agent interaction scenarios. For this purpose, both game-theoretic approaches as well as techniques that can be broadly categorized as partially observable Markov decision process (POMDP) approximations are surveyed.

(a) $t = 0\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$ (c) $t = 4\,\mathrm{s}$

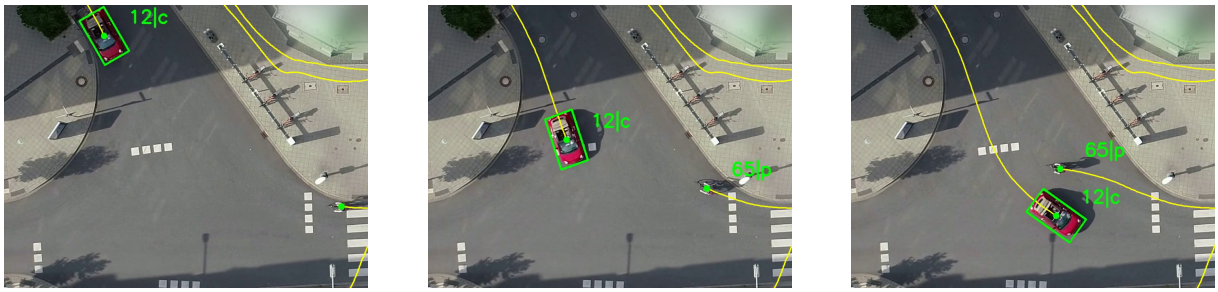


Figure 1.1.1: Example of a real-world interaction between a motorist (starting top left) and a cyclist (starting bottom right) at an intersection as recorded in [10].

1.2.1 Game-Theoretic Approaches

Various works have proposed methods for incorporating intention uncertainty by casting the interaction problem as a differential game of incomplete information. Existing game-theoretic models that explicitly consider incomplete information are limited to special cases of two-player zero-sum settings [11, 12] and application of such approaches has thus far only been demonstrated for a limited number of special examples; e.g. a special class of homing problems with linear dynamics and quadratic costs [13], games with special information structure that prohibit information gathering [14], and problems in which the only missing information is the initial state of the system [15, 16]. Due to the limitation to a narrow class of specialized examples, such approaches are not suitable for modeling complex HRI problems similar to the introductory example given in Section 1.1.

A common approach to dealing with intention uncertainty in complex interaction problems is to construct a controller by modeling players with uncertain intentions as adversaries, i.e. casting the planning problem as a zero-sum game by reasoning about worst case decisions of other players. Examples of this approach in which, from a robust control perspective, other players are treated as bounded disturbance to the system are given in [17, 18]. This technique can be suitable for problems in which safety is the main concern. However, as this approach ignores the fact that other players follow individual objectives, which in general may not be adversarial to the planner, it typically produces overly conservative strategies and is incapable of exploiting reactions of others. For example, a robot using this approach in a navigation problem like the example introduced in Section 1.1 can not impose a tight bound on the human worst case decision sequence and thus may conclude that *all* of its goal directed strategies are unsafe. This problem has been previously described as the *freezing robot problem* [19].

To avoid the issue of overly conservative strategies, general-sum formulations of interaction problems use the *true* — potentially non-adversarial — objectives of all players. However, solution methods to such formulations of differential games are only known for problems of complete information [12]. Furthermore, even in the case of fully observed games the dependence of each player's actions on the decisions of others poses a computational

challenge. Therefore, a common approach is to simplify the problem by establishing a leader-follower hierarchy amongst players; thus, converting the problem to a Stackelberg dynamic game [20]. Such approaches have been demonstrated in the context of HRI [21–23] but have been reported to yield undesirably aggressive behavior of the leader [24].

Other approaches avoid this pure leader-follower structure and aim for more symmetric roles of different players. In [24], the interaction is modeled in a hierarchical approach that solves a fully coupled dynamic game to inform a low-level controller. However, this approach solves the high-level Nash game through discretization of the state and input space and thus does not easily scale to multiple players.

To avoid the curse of dimensionality while maintaining symmetric roles of different players, recent work has focused on local approximations to Nash equilibria in differential games [1, 25, 26]. For example, Spica *et al.* [26] employ an iterative best response algorithm that successively updates each player's strategy by locally solving an optimal control problem in which the strategies of other players are fixed. Fridovich-Keil *et al.* [1] propose a method akin to differential dynamic programming that approximates a Nash solution by solving successive linear-quadaratic (LQ) approximations of the game.

1.2.2 POMDP Approximations

Several works address intention uncertainty by modeling certain aspects of human behavior as latent state variables and maintaining belief over these variables to compute optimized decisions.

In the field of autonomous driving, inference of behavioral parameters has been demonstrated to provide a significant benefit when interacting with other drivers [4] and a significant amount of work has focused on using this information in approximate POMDP schemes [4, 5, 27, 28]. However, these works typically use highly simplified models like IDM [29] and MOBIL [30] for the behaviors of other players. Similarly, Cunningham *et al.* [31], Galceran *et al.* [32], and Mehta *et al.* [33] use a library of hand-engineered feedback strategies to model the behavior of other agents. While these works demonstrate the benefit of employing behavioral inference, the policies used to model the behavior of other agents are somewhat arbitrarily chosen and may not specify suitable behavior for all cases.

Other works model humans as rational agents seeking to maximize their own objective function. In [34] and [35], human actions are predicted as the outcome of a noisily-rational decision process [36] with unknown goals. Therein, inference is used to reason about both the intentions of humans as well as the accuracy of the predictive model. However, these works treat agents as independently optimizing players and capture interaction between multiple agents only indirectly by reducing the confidence of predictions.

1.3 Contributions and Outline

This thesis has two main goals. First, to develop a planning approach that models intention uncertainty in game-theoretic formulations of HRI problems. Second, to investigate the utility of using such an approach in comparison to a game-theoretic planner that ignores this uncertainty. After providing the theoretical background that forms the foundations of this thesis in Chapter 2, the contributions of this work towards these goals are laid out in four chapters.

First, Chapter 3 presents an open-source software framework that has been developed as part of this thesis to aid the design and solution of differential games. This chapter discusses the design of the framework and provides benchmark results against existing implementations.

Thereafter, two types of intention uncertainty are examined in the context of HRI.
Chapter 4 analyzes scenarios in which a robot knows the human objectives but uncertainty arises from the fact that there are multiple strategies that humans may adopt to achieve their respective objectives. In this chapter, a new game-theoretic planning approach is proposed that addresses this type of uncertainty by reasoning about the likelihood of multiple equilibrium solutions to the underlying game.
Chapter 5 considers scenarios in which a robot has incomplete knowledge of human objectives. Here, the approach proposed in Chapter 4 is extended to infer human objectives while still considering multiple human strategies within each possible objective.
In both chapters, the proposed approaches are evaluated with respect to their utility for prediction and closed-loop planning in simulations of scenarios similar to the example given in Section 1.1.

Chapter 6 takes a first step towards future work in objective identification from large-scale datasets of observed behavior. To this end, an approach for objective identification from datasets of noise-corrupted partial state observations is proposed that works by numerically inverting the game solver presented in [1] via differentiable programming.

Finally, Chapter 7 summarizes the main results and provides an outlook towards future work.

Chapter 2

Fundamentals

This chapter provides the relevant theory used throughout this work to develop algorithms for game-theoretic planning under intention uncertainty. For this purpose, this chapter is divided into two parts.

Section 2.1 introduces the relevant fundamentals of differential games and discusses solution approaches to such problems. Section 2.2 gives a brief introduction to state estimation and discusses exact Bayesian inference as well as its approximation via particle filtering.

2.1 Noncooperative N-Player General-Sum Differential Games

Game theory is a conceptual framework that deals with the strategic interaction among multiple decision making agents, called *players*. Each player in a game-theoretic problem has an individual *objective function* that captures its preference ordering among multiple alternatives. For a non-trivial game, the objective function of a player depends not only on its own choices, but also on the decisions of other players. This mutual dependence of objective values induces a coupling between the *actions* of different players. Within the field of game theory there exists a fine-grained distinction of different problem classes, where each of these classes is characterized by various attributes. A thorough taxonomy of games and further theoretical background can be found in [37]. Following the nomenclature introduced in [37], this work is concerned with the class of *noncooperative N-player general-sum differential games.*

This problem class is characterized by the following properties:

Noncooperative: Players in this game are not allowed to form coalitions. That is, decisions can not be made collectively or with full trust that others may follow an a priori negotiated joint strategy that is beneficial for all players.

N-Player: The number of players, N, in the considered games may be an arbitrary

positive integer[1]. Specifically, the theory presented here holds beyond the 2-player setting — a special case for which many results have been published [2, 38–40].

General-Sum: The objective functions for different players are not constrained to a special structure but may encode arbitrary objectives that may be fully or partially competitive. Specifically, the sum of gains and losses of all payers may be unequal to zero. This contrasts with the well known special case of *zero-sum* games, where the objective functions of two players (or groups of players) always add up to zero and thus players are complete adversaries.

Differential Game: The problems studied belong to a special subclass of *dynamic* games, problems in which players observe each other's actions after every move and can react accordingly. Specifically, in *differential* games the decision process takes place in continuous-time and the *state* evolves according to a differential equation.

The remainder of this section proceeds as follows: First, the mathematical problem formulation for this class of games is presented (Section 2.1.1). Based on this, the notion of solution concepts for these problems are introduced (Section 2.1.2). Thereafter, solution methods are discussed, first showing the analytic solution for the special case of LQ games (Section 2.1.3) and subsequently presenting a recent approach that utilizes successive LQ approximations to solve nonlinear nonquadratic games.

2.1.1 Problem Formulation

A noncooperative N-player general-sum differential game is characterized by potentially nonlinear system dynamics

$$\dot{x} = f(t, x, u_{1:N})\,. \tag{2.1.1}$$

Here, each of N players is in control of an *input*, $u_i \in \mathbb{R}^{m_i}, i \in [N] \equiv \{1, \ldots, N\}$, to the system. By means of this input they gain control authority to influence the evolution of the joint *state*, $x \in \mathbb{R}^n$. Preference ordering among states and inputs for each player is captured by their respective cost functions, J_i, defined as an integral

$$J_i\left(u_{1:N}(\cdot)\right) \triangleq \int_0^T g_i\left(t, x(t), u_{1:N}(t)\right)\, dt, \forall i \in [N]\,, \tag{2.1.2}$$

of time-varying running costs, g_i, over a finite horizon, $t \in [0, T]$. Thus, player ith's cost, J_i, depends implicitly upon the state trajectory, $x(\cdot)$, which in turn depends upon initial state, $x(0)$, and control signals, $u_{1:N}(\cdot)$, as defined by Equation (2.1.1).

[1]Note, however, that for $N = 1$ the problem may not be considered but rather reduces to a single-player optimal control problem.

In this scenario, each player aims to minimize their respective cost function under the constraints of the dynamics by choosing a suitable state feedback strategy, $\gamma_i \in \Gamma_i$, i.e.

$$u_i(t) \equiv \gamma_i(t, x(t)) \,, \tag{2.1.3}$$

mapping time and state to their respective control input. Here, the strategy space, Γ_i, for player i is the set of measurable functions $\gamma_i : [0, T] \times \mathbb{R}^n \to \mathbb{R}^{m_i}$.

Note that, in this formulation, player ith's decision at time t only depends upon the current state, $x(t)$, and the time itself while the inputs of all other players, $u_{-i} \triangleq \{u_j\}_{j \in [N] \setminus i}$, remain unobserved. The cost that each player incurs when competing in this game does not only depend on their own strategy but rather on the choices of *all* players.

Additional Nomenclature

This work uses the term *strategy profile* to refer to the aggregate of strategies over all players and uses the shorthand $\gamma \triangleq \{\gamma_i\}_{i \in [N]}$. Furthermore, the $N - 1$ tuple of strategies including all strategies but the strategy of player j is denoted $\gamma_{-j} \triangleq \{\gamma_i\}_{i \in [N] \setminus j}$.

2.1.2 Solution Concepts

In general, there exist multiple different notions of optimal play for the problem formalized in Section 2.1.1 that are commonly referred to as *solution concepts* or *equilibrium concepts*. Different solution concepts are characterized by the approach they take to resolve potential conflicts between competing objectives of multiple players, and assumptions about the information players can access at different stages of the game.

One relatively intuitive approach is the solution for a social optimum, a strategy profile for which the sum of costs for all players is minimized. In this case, the problem may be viewed as a single-player optimal control problem with the decision variable being the joint strategy for all players. However, this solution concept requires an a priori negotiated and trusted agreement that strictly binds all players to this strategy; it neglects the noncooperative aspect of the problem. If a player can not fully trust others to adhere to the agreement, adopting a social optimum strategy poses the risk of incurring a high cost if another player decides to unilaterally deviate from this equilibrium to improve their payoff.

In a noncooperative scenario, rather than fully trusting each other, players must aim for solution points at which no player is unilaterally incentivized to change strategy. This noncooperative equilibrium concept is known as *Nash equilibrium*. Considering an overloaded notation of the player cost, J_i, to depend upon strategies rather than control signals, i.e. $J_i(\gamma_1; \ldots; \gamma_n) \equiv J_i(\gamma_1(\cdot, x(\cdot)); \ldots; \gamma_N(\cdot, x(\cdot)))$, formally, a Nash equilibrium is attained if each player follows a strategy, γ_i^*, such that the strategy profile satisfies

$$
\begin{aligned}
J_1^* &= J_1(\gamma_1^*; \gamma_2^*; \ldots; \gamma_N^*) \leq J_1(\gamma_1; \gamma_2^*; \ldots; \gamma_N^*), \\
J_2^* &= J_2(\gamma_1^*; \gamma_2^*; \ldots; \gamma_N^*) \leq J_2(\gamma_1^*; \gamma_2; \ldots; \gamma_N^*), \\
&\vdots \\
J_N^* &= J_N(\gamma_1^*; \gamma_2^*; \ldots; \gamma_N^*) \leq J_N(\gamma_1^*; \gamma_2^*; \ldots; \gamma_N),
\end{aligned}
\tag{2.1.4}
$$

for all $\gamma_i \in \Gamma_i, \forall i \in [N]$.

For a general game as defined in Section 2.1.1, finding Nash equilibria is known to be computationally intractable [41]. Therefore, several alternative equilibrium concepts have been proposed to approximate this problem. A common relaxation is to restrict the search to a *local Nash equilibrium* [25, 42–44]; that is, only requiring the conditions in Equation (2.1.4) to hold for γ_i in a local neighborhood of γ_i^*. Further simplification can be achieved by constraining the strategy space. This may be done by considering only open-loop strategies [43, 44], or by searching over a subspace of feedback strategies, e.g. considering only linear time-varying (LTV) feedback strategies[1, 45]. Other approaches involve simplification of the problem by pre-specifying an ordering amongst players, allowing earlier players to announce their strategies and expect later players to react accordingly[21, 46]. By establishing this kind of leader-follower structure, the problem is converted to a *Stackelberg* dynamic game and the corresponding solution concept is commonly known as *Stackelberg equilibrium* [20].

The behavior associated with each solution concept can be qualitatively different as it reflects the underlying assumptions about hierarchies, the information structure, and each player's decision-making process. Depending on the application domain, certain equilibrium concepts are more suitable than others to model the behavior of agents. For example, in the domain of human-robot interaction for navigation and collision avoidance problems, Stackelberg solutions have been reported to yield overly aggressive behavior of the leader [24]; or, equivalently, overly conservative behavior of the follower. Nash solutions, on the other hand, model symmetric roles and information structures and may be considered more suitable for this application domain as they encode shared responsibility for collision avoidance. Similarly, local approximations to Nash equilibria have been demonstrated to yield highly interactive strategies that qualitatively resemble global Nash strategies [1, 44, 47].

2.1.3 Analytic Solutions to Linear-Quadratic Games

In general, a differential game of the form described in Section 2.1.1 can not be solved tractably to a global Nash equilibrium [41]. However, a special subclass of these problems, called LQ games, characterized by linear dynamics and quadratic costs, allows for a closed-form solution via coupled Riccati differential equations [37, 48]. This section outlines a polynomial time algorithm for solving LQ games, closely following the presentation in [37].

Time Discretization

While the original problem has been formulated in continuous-time (c.f. Section 2.1.1), in practice, an implementation of a solver for online planning is executed in discrete time intervals. Therefore, here, the discrete-time LQ game solution is discussed where the discrete-time dynamics may be obtained from a zero-order hold of the inputs, u_i, over each time interval, Δt, i.e.

$$\begin{aligned} \hat{x}(t+\Delta t) = \hat{x}(t) + \int_t^{t+\Delta t} f(\tau, \hat{x}(\tau), \hat{u}_{1:N}(\tau))\, d\tau \\ \text{where } \hat{u}_i(\tau) = \gamma_i(t, \hat{x}(t)), \forall i \in [N], \text{ and} \\ \hat{x}(0) = x(0). \end{aligned} \tag{2.1.5}$$

Problem Formulation

We consider an discrete-time LQ version of the problem formulated in Section 2.1.1. Without loss of generality, we set $\Delta t = 1$, and use $t \in [T]$ as an index for the time step. Accordingly, the system is characterized by time-discrete LTV dynamics

$$x_{t+1} = A_t x_t + \sum_{i \in [N]} B_{i,t} u_{i,t}\,, \tag{2.1.6}$$

where $A_{i,t} \in \mathbb{R}^{n \times n}$ is the system matrix at time step t and $B_t \in \mathbb{R}^{n \times m}$ is the time-varying input matrix, column-wise constructed from per-player input matrices $B_{i,t} \in \mathbb{R}^{n \times m_i}$. Furthermore, each player's behavior is driven by an affine-quadratic objective

$$\begin{aligned} J_i(u_{1:N,0:T}) = &\sum_{t=0:T} g_{i,t}(x_{t+1}, u_{1:N,t}), \\ \text{where } g_{i,t}(x_{t+1}, u_{1:N,t}) = &\frac{1}{2} x_{t+1}^T Q_{i,t+1} x_{t+1} + x_{t+1}^T l_{i,t+1} \\ &+ \sum_{j \in [N]} \frac{1}{2} u_{j,t}^T R_{ij,t} u_{j,t} + u_{j,t}^T r_{ij,t}. \end{aligned} \tag{2.1.7}$$

Here, $Q_{i,t+1} \in \mathbb{R}^{n \times n}$, $l_{i,t+1} \in \mathbb{R}^n$ characterize the running state cost of the ith's player at time step $t+1$, and $R_{ij,t} \in \mathbb{R}^{m_j \times m_j}$, $r_{ij,t} \in \mathbb{R}^{m_j}$ parametrize the corresponding time-varying input costs.

Approach

For this LQ problem it can be shown that, if a Nash equilibrium exists, the corresponding equilibrium strategies take the time-varying affine form

$$\gamma_{i,t}^*(x_t) = -P_{i,t} x_t - \alpha_{i,t} \tag{2.1.8}$$

with feedback gain $P_{i,t} \in \mathbb{R}^{m_i \times n}$ and feed forward term $\alpha_{i,t} \in \mathbb{R}^{m_i}$ for player i at time step t. The sequence of $P_{i,t}$ and $\alpha_{i,t}$ must satisfy the following set of linear matrix equations:

$$[R_{ii,t} + B_{i,t}^T Z_{i,t+1} B_{i,t}] P_{i,t} + B_{i,t}^T Z_{i,t+1} \sum_{\substack{j \in [N] \\ j \neq i}} B_{j,t}^T P_{j,t} = B_{i,t}^T Z_{i,t+1} A_t, \text{ and} \tag{2.1.9}$$

$$[R_{ii,t} + B_{i,t}^T Z_{i,t+1} B_{i,t}] \alpha_{i,t} + B_{i,t}^T Z_{i,t+1} \sum_{\substack{j \in [N] \\ j \neq i}} B_{j,t}^T \alpha_{j,t} = B_{i,t}^T \zeta_{i,t+1} + r_{ii,t}\,, \tag{2.1.10}$$

$\forall i \in [N], \forall t \in [T]$. Here, $Z_{i,t} \in \mathbb{R}^{n \times n}$ represents the quadratic component of the cost-to-go obtained recursively from

$$Z_{i,t} = F_t^T Z_{i,t+1} F_t + \sum_{j \in [N]} P_{j,t}^T R_{ij,t} P_{j,t} + Q_{i,t}\,, \tag{2.1.11}$$

with the closed-loop system matrix, $F_t \in \mathbb{R}^{n \times n}$, defined as

$$F_t \triangleq A_t - \sum_{i \in [N]} B_{i,t} P_{i,t}, \forall t \in [T]. \tag{2.1.12}$$

Furthermore, $\zeta_{i,t} \in \mathbb{R}^n$ is the linear component of the cost-to-go obtained recursively from

$$\zeta_{i,t} = F_i^T (\zeta_{i,t+1} + Z_{i,t+1} \beta_t) + l_{i,t+1} + \sum_{j \in [N]} P_{j,t}^T (R_{ij,t} \alpha_{j,t} - r_{ij,t}) \tag{2.1.13}$$

with the state feed-forward term, $\beta_t \in \mathbb{R}^n$, defined as

$$\beta_t \triangleq - \sum_{j \in [N]} B_{j,t}^T \alpha_{j,t}. \tag{2.1.14}$$

Solution

We formulate a dynamic program based on the approach presented above to solve for the optimal strategies, γ_i^*, defined by $P_{i,t}$ and $\alpha_{i,t}$.

Starting at the final time, let $Z_{i,T+1} = Q_{i,T+1}$ and $\zeta_{i,T+1} = l_{i,T+1}$ be the parametrization of the final-state-cost, $g_{i,T}(x_{T+1}, u_{1:N,T})$.

Working backwards in time, for every time step we obtain the strategy profile segment, γ_t, that determines each player's feedback policy applied at time t and moves the system to x_{t+1}, by solving Equations (2.1.9) and (2.1.10). Numerically, this may be done efficiently by formulating a single matrix equation

$$S_t X_t = Y_t \tag{2.1.15}$$

with matrices

$$S_t \triangleq \begin{bmatrix} R_{11,t} + B_{1,t}^T Z_{1,t+1} B_{1,t} & B_{1,t}^T Z_{1,t+1} B_{2,t} & \cdots & B_{1,t}^T Z_{1,t+1} B_{N,t} \\ B_{2,t}^T Z_{2,t+1} B_{1,t} & R_{22,t} + B_{2,t}^T Z_{2,t+1} B_{2,t} & \cdots & B_{2,t}^T Z_{2,t+1} B_{N,t} \\ \vdots & \vdots & \ddots & \vdots \\ B_{N,t}^T Z_{N,t+1} B_{1,t} & B_{2,t}^T Z_{2,t+1} B_{2,t} & \cdots & R_{NN,t} + B_{2,t}^T Z_{2,t+1} B_{N,t} \end{bmatrix},$$

$$X_t \triangleq \begin{bmatrix} P_{1,t} & \alpha_{1,t} \\ \vdots & \vdots \\ P_{N,t} & \alpha_{N,t} \end{bmatrix}, \quad Y_t \triangleq \begin{bmatrix} B_{1,t}^T Z_{1,t+1} A_t & B_{1,t}^T \zeta_{1,t+1} \\ \vdots & \vdots \\ B_{N,t}^T Z_{N,t} A_t & B_{N,t}^T \zeta_{N,t+1} \end{bmatrix}, \tag{2.1.16}$$

where $S_t \in \mathbb{R}^{m \times m}$, $X_t \in \mathbb{R}^{m \times (n+1)}$ and $Y_t \in \mathbb{R}^{m \times (n+1)}$. Using this formulation, the system of equations may be solved via a LU decomposition of the dense square matrix S_t.

After obtaining the strategy profile segment for time step t, we recurse on Equations (2.1.11) and (2.1.13) to propagate the cost-to-go backwards in time. This procedure is repeated until we reach the initial time step, $t = 0$.

2.1.4 Iterative Linear-Quadratic Approximations for Nonlinear Nonquadratic Games

Many interactive scenarios relevant in practice are too complex to be well modeled by a single LQ game. However, LQ games may still be used to capture the local nature of these problems. Based on this idea, recent work introduced approximation methods that approach N-Player general-sum games[2] with nonlinear dynamics and nonquadratic costs from the perspective of classical LQ games [1, 45]. This section outlines one such method, the iterative linear-quadratic games (ILQG) algorithm presented in [1].

Algorithm

The high-level idea of ILQG is summarized in Algorithm 1. On an abstract level, the approach extends the idea of the well known iterative linear-quadratic regular (ILQR) algorithm [51, 52] — a method for smooth nonlinear single-player optimal control problems — to the domain of N-player general-sum differential games. Given the initial joint state, x_0, initial strategies, $\{\gamma_i^0\}_{i \in [N]}$, system dynamics, f, player costs, $\{g_i\}_{i \in [N]}$, and problem horizon, T, the algorithm iteratively updates the strategy profile until convergence using the following steps:

1. Rollout Integrate the dynamics, f, forward, starting at x_0 to simulate the joint state trajectory, ξ^k, of the nonlinear system when applying the current strategies (line 3).

[2] A related approach has been studied earlier in [49, 50]. However, results presented there are limit to two-player zero-sum scenarios.

2. **LQ Approximate** Compute the local LQ approximation to the game along the reference trajectory, ξ^k, by linearizing the dynamics and quadraticizing the cost (line 4 – 5).

3. **Solve LQ Game** Update the strategies using the analytic solution to the LQ approximation (line 6 – 7).

Like ILQR, ILQG requires the problem to be sufficiently smooth in order to admit local LQ approximations. That is, it requires the system dynamics, f, to be continuously differentiable in $\{x, u_i\}$ and requires the running costs, g_i, for each player to be twice differentiable in $\{x, u_i\}$, both uniformly in t, respectively.

To provide further insight, the remainder of this paragraph discusses the relevant details for some of the subroutines:

LinearizeDynamics computes the Jacobian linearization of the system dynamics, f, along the operating point ξ^k. Let $\delta x(t) \triangleq x(t) - \hat{x}(t)$ and $\delta u_i(t) \triangleq u_i(t) - \hat{u}_i(t)$ be the deviation from the operating point, x_i^k. The linear system approximation of f about ξ^k is given by

$$\delta\dot{x}(t) \approx A(t)\delta x(t) + \sum_{i\in[N]} B_i(t)\delta u_i(t), \tag{2.1.17}$$

with Jacobians $A(t) = D_x f\Big|_{\xi^k}$ and $B_i(t) = D_{\hat{u}_i} f\Big|_{\xi^k}$.

QuadraticizeCost computes the quadratic approximation of the running cost

$$\begin{aligned} g_i\left(t, x(t), u_{1:N}(t)\right) \approx g_i\left(t, \hat{x}(t), \hat{u_{1:N}}(t)\right) &+ \frac{1}{2}\delta x(t)^T \left(Q_i(t)\delta x(t) + 2l_i(t)\right) \\ &+ \frac{1}{2}\sum_{j\in[N]} \delta u_j(t)^T \left(R_{ij}(t)\delta u_j(t) + 2r_{ij}(t)\right), \end{aligned} \tag{2.1.18}$$

with gradients $l_i(t) = D_x g_i\Big|_{\xi^k}$, $r_{ij}(t) = D_{u_{ij}} g_i\Big|_{\xi^k}$ and Hessians $Q_i(t) = D^2_{xx} g_i\Big|_{\xi^k}$, $R_{ij}(t) = D^2_{u_j u_j} g_i\Big|_{\xi^k}$. Note that in this formulation, mixed partials $D^2_{u_j u_k} g_i$ and $D^2_{x u_j} g_i$ are neglected, as they rarely appear in cost structures of practical interest.

SolveLQGame solves the LQ game defined by the tuple $(A, \{B_i\}_{i\in[N]}, Q_i, l_i, \{R_{ij}\}_{i\in[N]}, \{r_{ij}\}_{i\in[N]}, T)$, using the dynamic program presented in Section 2.1.3. The result of this computation is an affine strategy for each player that describes the local behavior of each player for deviations from the current operating point, ξ^k,

$$\tilde{\gamma}_i^k(t, x(t)) = \hat{u}_i(t) - P_i^k \delta x(t) - \alpha_i^k(t). \tag{2.1.19}$$

StepTowards updates the strategies, $\{\gamma_i^{k-1}\}$, with a step towards the local equilibrium strategies,

$$\tilde{\gamma}_i^k(t, x(t)) = \hat{u}_i(t) - P_i^k \delta x(t) - \eta\alpha_i^k(t), \tag{2.1.20}$$

where $\eta \in (0, 1]$ is the relative step size.

Similar to (quasi-)Newton methods for single-objective optimization problems, the scaling of the step size is motivated by the fact that the solution to the current iterate is only valid for the *local* model (i.e. the LQ game) of the problem. However, in contrast to single objective optimization problems, a simple line search over step size η to ensure sufficient decrease in cost is not meaningful as each player has their own respective cost function and player costs may conflict. Instead, in [1] a prefixed small step size η has been reported to provide good convergence for many problems. Note, however, that for this choice of step size convergence may not be guaranteed.

CONVERGED determines convergence to a fixed point based upon the distance between the state trajectories traced out when applying $\{\gamma_i^k\}$ versus $\{\gamma_i^{k-1}\}$.

Approximate Local Nash Equilibria

If this algorithm converges at the Kth iteration, the resulting strategy profile, $\{\gamma_i^K\}$, reproduces the operating point of the previous iterate, ξ^{K-1}, and constitutes a global Nash equilibrium of the LQ approximation at that operating point. While it is tempting to presume that such fixed points also satisfy the *local* Nash equilibrium concept of the original problem, this is not always true because the LQ approximation neglects higher order coupling terms between each player's running cost, g_i, and other players' inputs, u_{-i} [1]. Therefore, this work uses the term *approximate* local Nash equilibrium for the fixed points of Algorithm 1. Despite the subtle differences between *approximate local Nash* equilibria and their non-approximate counterparts, they have been demonstrated to exhibit similarly interactive strategies [1].

Algorithm 1 iterative linear-quadratic games (ILQG)

procedure ILQG($x_0, \{\gamma_i^0\}_{i\in[N]}, f, \{g_i\}_{i\in[N]}, T$)
  for iteration $k = 1, 2, \ldots$ **do**
    $\xi^k \equiv \{\hat{x}(t), \hat{u}_{1:N}\} \leftarrow$ GETTRAJECTORY($f, x_0, \{\gamma_i^{k-1}\}_{i\in[N]}$)
    $\{A(t), B_i(t)\} \leftarrow$ LINEARIZEDYNAMICS(f, ξ^k)
    $\{Q_i(t), l_i(t), R_{ij}(t), r_{ij}(t)\} \leftarrow$ QUADRATICIZECOST(f, ξ^k)
    $\{\tilde{\gamma}_i^k\} \leftarrow$ SOLVELQGAME($A(t), B_i(t), Q_i(t), l_i(t), R_{ij}(t), r_{ij}(t), T$)
    $\{\gamma_i^k\} \leftarrow$ STEPTOWARDS($\{\tilde{\gamma}_i^k\}, \{\gamma_i^{k-1}\}$)
    if CONVERGED($\{\gamma_i^k\}, \{\gamma_i^{k-1}\}$) **then**
      return $\{\gamma_i^k\}$
    end if
  end for
end procedure

2.2 State Estimation for Hidden Markov Models

Environments are characterized by *state* and *state dynamics*. In this work, the term state implicitly refers to the notion of *complete* or *Markovian* state. The Markov property entails that future states are conditionally independent of past states, given the current state and input [53–55]. Hence, a Markovian state is the *best predictor* for the future evolution of the environment. Due to this predictive power, knowledge of the state is important for optimal decision making.

However, many problems do not allow agents to directly observe this state and thus do not admit access to perfect state information. Instead, the agent may receive observations, o_t, that are emissions of the latent state, s_t. Using this data, the agent can *infer* information about the state of the environment. The aggregate information that the agent recovers from observations is commonly represented by the *belief*,

$$b_t(s_t) \triangleq p(s_t|o_{1:t}), \tag{2.2.1}$$

which corresponds to the conditional probability of the current state, s_t, given all observations, $o_{1:t}$, received up to time t. In the case of Markovian state, the belief is a sufficient statistic to compute an optimal decision at time t [54]. Therefore, being able to accurately maintain this belief is crucial to solving problems characterized by state uncertainty.

A suitable model for these inference problems with Markovian state is the hidden Markov model (HMM) and the process of maintaining the belief is commonly referred to as *state estimation*. This section presents two approaches to state estimation for HMMs. First, Section 2.2.1 briefly presents the exact Bayesian update for recursive state estimation. Based on this, Section 2.2.2 presents an approximation to the Bayesian update, the particle filter.

2.2.1 Exact Bayesian Inference

This section briefly outlines the idea of Bayesian inference for HMMs to provide better understanding of the inference strategy developed throughout this work. A thorough discussion of fundamentals of probability theory and statistics can be found in [53, 56] and is beyond the scope of this work.

As a minimal example, consider the dynamic Bayesian network (DBN) representation of the HMM depicted in Figure 2.2.1. In this representation, shaded nodes correspond to the observed data, here $o_{1:t}$, while the other variables remain unobserved, here the sequence of latent states, $s_{1:t}$. For this statistical model, the recursive belief update may be derived as follows:

We begin by expanding the definition of conditional probability in the belief (c.f. Equation (2.2.1)), i.e.

$$b_t(s_t) \triangleq p(s_t|o_{1:t}) = \frac{p(s_t, o_{1:t})}{p(o_{1:t})}. \tag{2.2.2}$$

In this expression, we introduce the state at the previous time, s_{t-1}, by marginalizing the

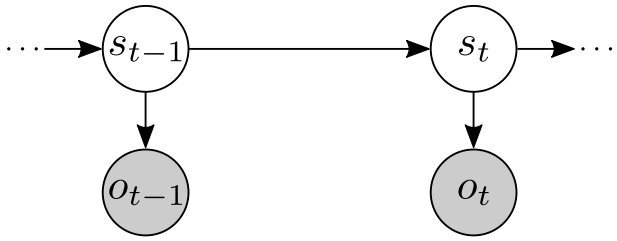


Figure 2.2.1: Visualization of a minimal HMM. Shaded nodes in the graph correspond to observed variables.

augmented joint probability, $p(s_t, s_{t-1}, o_{1:t})$, to obtain

$$b_t(s_t) = \frac{\int_{s_{t-1}\in\mathcal{S}} p(s_t, s_{t-1}, o_{1:t})\, ds_{t-1}}{p(o_{1:t})}. \tag{2.2.3}$$

Finally, we expand the numerator into a product of conditional probabilities and recover

$$b_t(s_t) = \frac{\int_{s_{t-1}\in\mathcal{S}} p(o_t|s_t, s_{t-1}, o_{1:t-1}) p(s_t|s_{t-1}, o_{1:t-1}) p(s_{t-1}|o_{1:t-1})\, ds_{t-1}}{p(o_t|o_{1:t-1})}. \tag{2.2.4}$$

By recognizing that o_t is conditionally independent of all past observations and states given s_t, and that s_t itself only depends upon the previous state, s_{t-1} (i.e. exploiting the Markov property), we can simplify Equation (2.2.4) to

$$b_t(s_t) = \frac{\int_{s_{t-1}\in\mathcal{S}} p(o_t|s_t) p(s_t|s_{t-1}) \overbrace{p(s_{t-1}|o_{1:t-1})}^{b_{t-1}}\, ds_{t-1}}{p(o_t|o_{1:t-1})}. \tag{2.2.5}$$

Note that in this expression the denominator, $p(o_t|o_{1:t-1})$, is the same for all $s_t \in \mathcal{S}$ and thus takes the role of a normalizing constant. Therefore, Equation (2.2.5) may alternatively be written as

$$b_t(s_t) \propto \int_{s_{t-1}\in\mathcal{S}} p(o_t|s_t) p(s_t|s_{t-1}) b_{t-1}(s_{t-1})\, ds_{t-1}. \tag{2.2.6}$$

This is the Bayesian update we initially seeked to derive. Intuitively, the Bayesian update may be understood as propagating the belief at the previous time, b_{t-1}, through the state dynamics, $p(s_t|s_{t-1})$, and weighting the propagated belief with the observation model, $p(o_t|s_t)$. Given initial information, $b_0(s_0) \triangleq p(s_0)$, this rule can be applied recursively to update the belief with new observations at every time step.

While this update rule is only valid for the minimal example of a HMM given in Figure 2.2.1, for more complicated models, the derivation follows a similar strategy: augment and marginalize the belief distribution, then expand the joint distribution into a product of conditional probabilities and exploit conditional independence.

2.2.2 Approximate Bayesian Inference via Particle Filtering

The exact update rule presented in Section 2.2.1 provides a theoretical formalism to perform Bayesian inference on HMMs. However, computing the exact update often

remains intractable, except for a few special cases — e.g. problems with Gaussian noise and linear dynamics admit exact inference via a Kalman filter [53, 57]. More complicated problems require marginalization over high-dimensional state spaces or exhibit stochastic dynamics that are not easily represented by parametric probability distributions. These cases typically require some form of approximation to the Bayesian update.

A possible approximation to the recursive Bayesian update presented in Section 2.2.1 is to replace the exact integral for marginalization with Monte Carlo integration. One class of inference methods that takes this approach are the so called *particle filters* or sequential Monte Carlo (SMC) methods. To date, a variety of particle filtering techniques have been proposed that use different sampling strategies and belief representations [53, 58]. All of these methods use a collection of sampled states, the so called *particles*, as a *nonparametric* representation to approximate the belief.

Here, the vanilla version of a *bootstrap particle filter* is presented [59]. This filter uses a *weighted particle belief* representation

$$\hat{b}(s) \triangleq \eta \sum_{m=1}^{M} w^{(m)} \mathbf{1}(s^{(m)} = s) \tag{2.2.7}$$

where M is the total number of particles, $s_t^{(m)} \in \mathcal{S}$ is the state associated with the mth particle, $w^{(m)} \in \mathbb{R}$ is the corresponding weight, $\eta = \sum_{m=1}^{M} w^{(m)}$ is a normalizing constant, and $\mathbf{1}(\cdot)$ denotes the indicator function that takes the value 1 iff the given condition is true:

$$\mathbf{1}(\mathcal{C}) \equiv \begin{cases} 1 & \text{if } \mathcal{C} \text{ is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.2.8}$$

As the number of particles, M, is increased, this belief representation can approximate the true belief b with arbitrary accuracy, i.e. $\hat{b}(s) \approx_{M \uparrow \infty} b(s)$.

Algorithm 2 summarizes the bootstrap particle filter that approximates the Bayesian update for the HMM presented in Section 2.2.1. A schematic visualization of this algorithm is provided in Figure 2.2.2. Input to the algorithm are the weighted particles of the posterior belief, $\mathcal{B}_{t-1} \equiv \{(s_{t-1}^{(m)}, w_{t-1}^{(m)})\}_{m \in [M]}$, as well as the current observation, o_t.

On an abstract level, the filter simulates the stochastic evolution of all particles — each a hypothetical true state of the environment — and evaluates the likelihood of the observation under each hypothesis to update the weights. As per the stages visualized in Figure 2.2.2 the algorithm proceeds as follows:

1. **Transition** Each particle is propagated through the state dynamics, $p(s_t|s_t^{(m)})$, to *propose* a new state, $s_t^{(m)}$, at the next time step (line 4).

2. **Observation Weighting** The weight of each particle is multiplied with the likelihood of receiving observation o_t if the true state of the environment was given by state particle $s_t^{(m)}$ (line 5).

3. **Resampling** A new set of particles is sampled with replacement from the belief defined by the reweighted particles, $\mathcal{B}_t$, to focus the samples on relevant regions of the state space (line 8).

One reason for the popularity of this method is the fact that sampling from the transition model (step 1, line 4 of Algorithm 2) does not require an explicit representation of the transition density $p(s_t|s_t^{(m)})$. Instead, it only requires access to a *generative model* of the state dynamics, i.e. a black box simulator of the environment.

Furthermore, note that there are several possible implementations of the RESAMPLE procedure. In the simplest case, resampling may be realized by resetting the weights, i.e. $w_i = 1, \forall i \in [M]$, and sampling the state particles independently from the posterior belief defined by the reweighted particles, $\bar{\mathcal{B}}_t$. However, a successful application of this algorithm to practical problems typically requires more sophisticated resampling strategies; particularly, if only a relatively small number of particles can be simulated due to computational constraints. Some of the considerations to make when designing a suitable resampling strategy are discussed hereafter.

First, note that resampling is optional; mathematically, the soundness of the algorithm does not depend on this step. In fact, for any finite number of particles resampling poses the risk of erroneously changing the posterior belief due to the effect of *sampling variance*. Intuitively, this variance arises from a loss of diversity as some particles may not be sampled. Therefore, while occasional resampling is important to maintain a high sampling density in important regions of the state space, it should only be performed if necessary. There exist several heuristics to determine the need for resampling [53]. A popular strategy is to trigger resampling if the variance of the importance weights, $\{w_t^{(m)}\}$, exceeds a predefined threshold. Additional mitigation of this problem may be achieved by employing an advanced *low variance resampling* strategy [53]. One such method is stochastic universal sampling (SUS) [60]. Rather than sampling particles independently, SUS chooses samples according to a sequential stochastic process that exhibits a lower sampling variance while keeping the probability of sampling a specific particle proportional to its weight. A thorough discussion of different low variance sampling strategies can be found in [61, 62].

Finally, it should be noted that the state transition model, $p(s_t|s_{t_1})$, is not the only distribution that may be used to propose new samples. For example, if the measurement model, $p(o_t|s_t)$, can be easily inverted it would pose a suitable proposal distribution if instead the samples are weighted with the probability of reaching the associated state from the posterior belief[3], i.e. $w_t^{(m)} = \int_{s_{t-1} \in \mathcal{S}} p(s_t^{(m)}|s_{t-1}^{(m)}) b(s_{t-1})\, ds_{t-1}$[53]. Similarly, a different factorization of the belief may be chosen to design alternative sampling densities, e.g. [63]. In practice, the suitability of a given approach depends upon the computational complexity of sampling from the corresponding proposal distribution and evaluating the weighting density. A detailed survey of proposal distributions for particle filtering can be found in [58].

[3]Note, however, that evaluating this integral is, in itself, a challenging problem that may not afford an efficient closed form solution.

Algorithm 2 Bootstrap particle filter corresponding to the Bayesian update presented in Equation (2.2.6).

procedure BOOTSTRAPPFUPDATE($\mathcal{B}_{t-1} \equiv \{(s_{t-1}^{(m)}, w_{t-1}^{(m)})\}_{m\in[M]}, o_t$)
  $\bar{\mathcal{B}}_t \leftarrow \emptyset$
  for particle index $m = 1 \dots M$ **do**
    $s_t^{(m)} \leftarrow$ sample from $p(s_t|s_{t-1}^{(m)})$
    $w_t^{(m)} \leftarrow w_{t-1}^{(m)} p(o_t|s_t^{(m)})$
    $\bar{\mathcal{B}}_t \leftarrow \bar{\mathcal{B}}_t \cup (s_t^{(m)}, w_t^{(m)})$
  end for
  $\mathcal{B}_t \leftarrow$ RESAMPLE($\bar{\mathcal{B}}_t$) ▷ may be skipped depending on the problem and belief
  return $\mathcal{B}_t$
end procedure

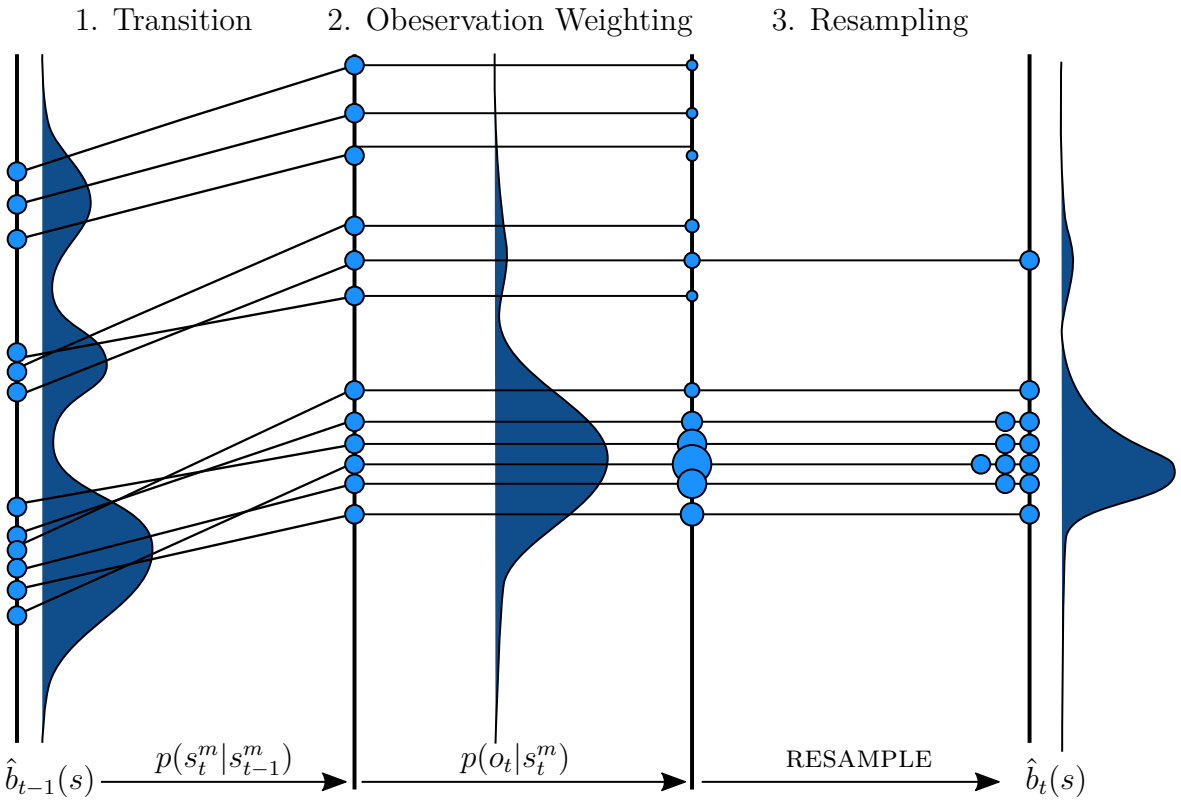


Figure 2.2.2: Schematic visualization of a bootstrap particle filter.

Chapter 3

A Julia Framework for General-Sum Differential Games

A key obstacle that impedes the application of differential games to applications characterized by high-dimensional states, real-time constraints, and fast planning rates — such as robotics — is the complexity involved with describing and solving these problems. Here, the term *complexity* is explicitly used with threefold meaning: First, the algorithmic time and space complexity of solution methods; Second, the challenges involved with implementing these algorithms efficiently; And third, the conceptual complexity of the interface used to describe and set up the problem. While the first two aspects are crucial for quick solution of the problem, the latter aspect is particularly important to admit quick iteration of different designs, e.g. to experiment with different cost structures that encode the behavior of each player.

As discussed in Section 1.2, in terms of computational complexity recent work offers efficient approximations to noncooperative games and several works have demonstrated real-time performance of these algorithms in C++ implementations [1, 26, 43, 44]. However, to the best of the author's knowledge, only Fridovich-Keil *et al.* [1] provide a publicly available implementation of their solver[1] and little work has focused on providing flexible interfaces and tools for the design phase of differential games.

This chapter describes iLQGames.jl, a software framework for designing and solving differential games. Using the ILQG algorithm presented in Section 2.1.4, iLQGames.jl solves *fully observed* instances of differential games; i.e., it solves a differential game with *known objectives* to a local equilibrium and does not per se consider intention uncertainty. The framework has been created as part of this thesis to aid the development of algorithms for intention uncertainty in differential games discussed in the subsequent chapters. Nonetheless, its design is focused on being a useful tool also for other areas of multi-agent interaction research.

iLQGames.jl is written in the Julia programming language [64] and makes use of the language's genericity to provide a flexible interface that admits quick iteration of different problem designs and keep up with execution times of a comparable C++ implementation.

[1] `https://github.com/HJReachability/ilqgames`

The iLQGames.jl open-source software is publicly available at `https://github.com/lassepe/iLQGames.jl`.

3.1 Architecture for Rapid Design and Solution

This section describes the key aspects of the framework that enable its flexibility and performance, and make it an effective tool for differential game research. Besides introducing the framework itself, this discussion aims to provide helpful insight for the implementation of other solvers, tools and problem interfaces.

Rapid Design

When modeling a practical scenario of multi-agent interaction as a differential game, it may not be immediately clear what are suitable dynamics and costs to describe the problem. Therefore, iLQGames.jl provides a lightweight interface for describing differential games that allows users to set up a model in few lines of code. Specifically, the user can define a differential game by providing the differential equation governing the system dynamics, a cost function for each player, the indices of the inputs that each player controls, and the horizon of the game.

Listing 1 shows the code for a minimal example that illustrates this setup procedure. In this example two players control a single unicycle with 4-dimensional state. Player-1 controls the steering angle of the unicycle and wishes to stay close to the origin. Player-2 controls the acceleration and wishes to keep the unicycle at a speed of $1\,\mathrm{m\,s^{-1}}$.

Listing 1 Implementation of a nonlinear two-player general-sum game in iLQGames.jl.

```
# constants: number of {states,inputs}, sampling time, horizon
nx, nu, ΔT, game_horizon = 4, 2, 0.1, 200

# setup a dynamical system
struct Unicycle <: ControlSystem{ΔT,nx,nu} end
dx(cs::Unicycle, x, u, t) = SVector(x[4]cos(x[3]),
                                    x[4]sin(x[3]),u[1],u[2])
dynamics = Unicycle()

# player-1 wants the unicycle to stay close to the origin,
# player-2 wants to keep the speed of the unicycle close to 1 m/s
costs = (FunctionPlayerCost((g,x,u,t) -> (x[1]^2+x[2]^2+u[1]^2)),
         FunctionPlayerCost((g,x,u,t) -> ((x[4]-1)^2+u[2]^2)))

# indices of inputs that each player controls
player_inputs = (SVector(1), SVector(2))
# the horizon of the game
g = GeneralGame(game_horizon, player_inputs, dynamics, costs)
```

With this light-weight problem description, the user can invoke the ILQG with given

initial conditions and initial strategies. For the minimal example introduced above, this step is shown in Listing 2. Most notably, even though ILQG is based on successive LQ approximations of the game, the user does *not* need to hand-specify linearization of the dynamics or quadratization of the costs. Instead, iLQGames.jl by default uses automatic differentiation to compute these LQ approximations efficiently [65].

Listing 2 Solving a game in iLQGames.jl.

```
solver = iLQSolver(g)
x0 = SVector(1, 1, 0, 0.5) # initial state
converged, trajectory, strategies = solve(g, solver, x0)
```

Furthermore, being written in pure Julia — a language with strong focus on scientific programming — iLQGames.jl can directly be used with various other packages from the ecosystem. By this means, iLQGames.jl natively supports visualization of the state and input trajectories of a game solution or projections of the cost landscape for a given player to support the design process.

For the minimal example, Figure 3.1.1 shows the noncooperative equilibriums solution computed and visualized by iLQGames.jl. This visualization shows the inputs applied by each player as well as the path traced out by the unicycle when simulating the equilibrium strategies computed above.

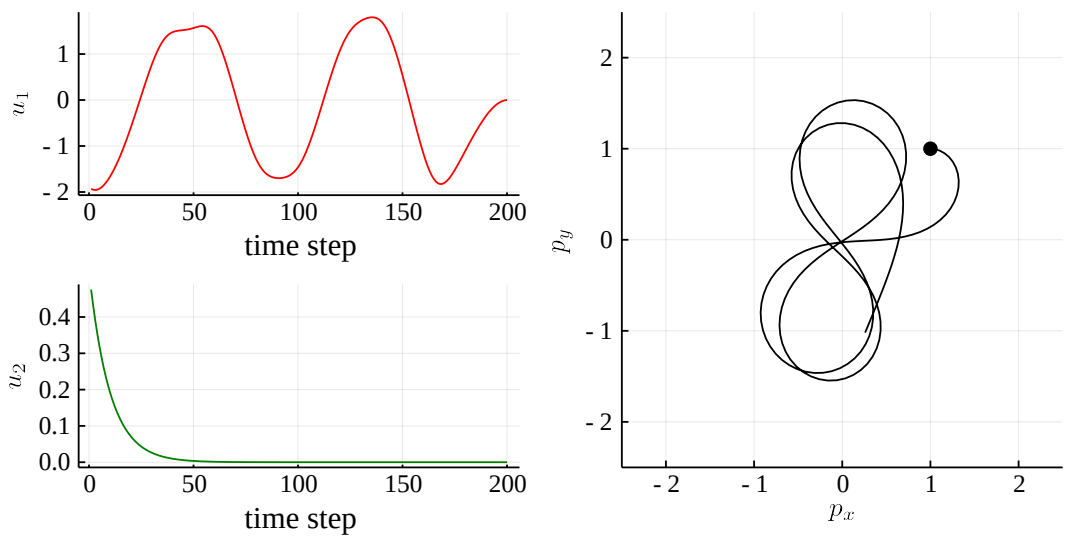


Figure 3.1.1: Visualization of the path traced out by the unicycle at the equilibrium solution of the game defined in Listing 1 Left: Player-1's control signal is highlighted in red; Player-2's control signal is highlighted in green. Right: path traced out by the unicycle when simulating the equilibrium strategies.

Rapid Solution

Despite being a high-level language that offers wide-ranging abstraction, the Julia compiler generates highly optimized code[2]. Most notably for the use case discussed here, this allows iLQGames.jl to solve each LQ iterate very efficiently via a fully stack-allocated dynamic program implemented in a readable high-level style in less than 70 lines of code. In fact, for moderately sized games, this optimization allows iLQGames.jl to outperform the C++ implementation presented in [1]. While it must be noted that Fridovich-Keil *et al.* [1] do not make this optimization and in theory a C++ implementation can achieve similar performance, making comparable optimizations in C++ may not be possible without sacrificing readability.

Flexibility

Furthermore, the generic function dispatch mechanism used in Julia allows users to overload default implementations at various levels of the solver to make problem specific optimizations. For example, users can specify a custom method to perform LQ approximations once they have chosen a design for the problem.

Beyond that, iLQGames.jl supports exploitation of special structure of the dynamics to speed up computation. This aspect is realized via the `LinearizationStyle` *trait* concept. By default, a dynamical system is attributed the `JacobianLinearization` trait and automatic differentiation or a user-defined linearization is used to obtain LQ approximations of the game. However, if the dynamics are feedback linearizable, the user can optionally specify the `FeedbackLinearization` trait for a model to invoke a specialized version of the solver presented in [47]. Furthermore, linear systems are assigned the `TrivialLinearization` trait and linearization is explicitly skipped. This trait concept can be easily extended to other special types of systems and thus allows users to seamlessly customize the solver with small local changes without the need to overload other parts of the routine.

3.2 Benchmark Results

The performance of iLQGames.jl is evaluated by benchmarking it on three problems against the C++ implementation presented in [1]. For additional reference, a Python implementation of an LQ solver is benchmarked as well. The benchmark problems are (a) a minimal LQ game (LQ), (b) a nonlinear nonquadratic collision avoidance problem (NL), and (c) a feedback linearized version of this collision avoidance game (FBL).

Table 3.1 summarizes the benchmark results. Here, LQ-Python denotes a Python implementation of the dynamic program used at the inner loop of ILQG, iLQGames-C++ refers to the implementation used in Fridovich-Keil *et al.* [1], and iLQGames.jl-MD and iLQGames.jl-AD refer to iLQGames.jl utilizing manual differentiation and automatic

[2]This has been prominently demonstrated in [66].

differentiation, respectively. Each game is solved over a horizon of 100 time steps on a standard laptop. The reported run times are the average over 100 successive calls to the respective implementation.

The Python implementation for the LQ case is multiple orders of magnitude slower than the C++ version and iLQGames.jl. Therefore, a Python implement would not scale well to nonlinear nonquadratic problems. In fact, preliminary tests not listed in Table 3.1 resulted in run times of more than 30 s for the nonlinear problem. iLQGames.jl with automatic differentiation, on the other hand, achieves moderate runtime and is sufficiently fast to evaluate different problem designs. When utilizing manually specified LQ approximation, as is done in the C++ version, iLQGames.jl outperforms the baseline.

Table 3.1: Benchmark results of differential game solvers. The tuple behind each benchmark problem indicates the number of players (P) and the dimensionality of the state (D). Colored bars indicate relative runtime and are normalized to (a) the highest runtime among all implementations (b,c) a runtime of 100 ms to allow real-time execution at 10 Hz.

	(a) LQ (2P, 2D)	(b) NL (3P, 12D)	(c) FBL (3P, 12D)
LQ-Python	20.800 ms	n/a	n/a
iLQGames-C++	0.3490 ms	16.27 ms	13.25 ms
iLQGames.jl-MD	0.0044 ms	7.19 ms	3.98 ms
iLQGames.jl-AD	n/a	63.57 ms	52.50 ms

Chapter 4

Planning with Equilibrium Uncertainty

This thesis's first investigation into intention uncertainty in HRI focuses on the issue of uncertain equilibrium selection. As an introductory example, recall the intersection navigation problem discussed earlier in Section 1.1. For this scenario, it is intuitively clear that the interaction admits multiple solutions. For example, one in which the cyclist passes the motorist in the front, and another where the car passed in the back. If the car in this example was controlled by an autonomous agent, this agent needs to understand which solution the human road user is aiming for to safely navigate the intersection. In differential games this *multimodality* of behavior manifests as a *multiplicity of solutions*; that is, a situation in which the problem admits multiple Nash equilibria. This chapter proposes an approach for tractably accommodating uncertainty arising from the presence of multiple equilibria in game-theoretic formulations of interactions between a single autonomous agent and one or multiple humans.

This chapter is structured as follows. First, Section 4.1 describes multimodality of behavior in the context of differential games and highlights the associated challenges that prohibit an a priori determination of a unique solution for interactions with humans. In view of these challenges, Section 4.2 proposes a new game-theoretic decision model for scenarios with equilibrium uncertainty. Section 4.3 then presents a tractable online planning approach based on this decision model. Finally, Section 4.4 evaluates the performance of the presented approach.

4.1 Multiplicity of Solutions in Differential Games

It is well known that noncooperative games, including differential games, can admit multiple equilibria [37]. From the perspective of a static analysis of interaction problems the ability to recover multiple solutions from a game formulation is a desirable property in that it allows to discover and examine different behaviors (see, for example, [67]). From the perspective of online planning, however, this multiplicity of solutions causes

uncertainty about how other players will behave and hence obstructs the predictive power of a game-theoretic planning approach.

Several methods have been proposed to eliminate Nash equilibria in an effort to obtain a unique solution point for decision making. An intuitively straight forward criterion for the selection of a solution point is given by the notion of *payoff dominance*, a condition in which there exists a solution that exhibits strictly higher payoffs (or lower costs) for all players than any other equilibrium of the game. Equilibria that fulfill this condition are commonly referred to as *Pareto-optimal* Nash equilibria.

Unfortunately, many games do not admit a Pareto ranking of equilibria and therefore payoff dominance alone is not a sufficient criterion to guarantee a unique solution. Moreover, even for problems in which a Pareto ranking of equilibria can be established, there exists substantial evidence that humans do not necessarily choose a payoff dominating strategy [68–70]. Instead, human interaction has been demonstrated to be often characterized by another equilibrium selection criterion: *risk dominance*. In risk dominance, not only the on-equilibrium payoffs are considered but also the cost that a player incurs if others deviate from that equilibrium [71]. Intuitively, a risk dominant equilibrium corresponds to the most stable solution point, i.e. the solution point from which other players are least likely to deviate due to large opportunity cost.

In practice, human interaction is likely to be characterized by a combination of both payoff and risk dominance as equilibrium selection criteria [68] and a significant amount of research has focused on providing theoretical models that rationalize observed selection schemes [71–73]. However, existing results are limited to studies of finite and often even static games and do not extend to the area of differential game theory. Furthermore, even for such structurally less complex games, behavioral studies evidence that human preferences for certain equilibria are significantly more complex and, e.g., may even depend on the interaction history [68, 74].

4.2 Problem Statement and Decision Model

This section proposes a new game-theoretic planning model that explicitly models uncertainty arising from multiplicity of solutions in scenarios of interaction between a single autonomous agent and one or multiple humans. On an abstract level, instead of trying to determine a single equilibrium for decision making a priori, in this model the planner casts a distribution over potential solutions and reasons about the likelihood of each solution as it observes human actions. This section describes the details of this decision model and illustrates its key ideas based on a running example introduced hereafter.

4.2.1 Introduction of a Running Example

For the sake of clarity, a running example similar to the scenario presented in Section 1.1 is introduced. This running example is used throughout this chapter to illustrate the

proposed decision model. While in a first step this example is only outlined abstractly, it is further extended as the problem statement and decision model is presented hereafter.

In this example, consider $N = 3$ agents — a robot and two human players — that navigate in a shared environment with no obstacles. This interaction problem may be envisioned as schematically shown in Figure 4.2.1.

The behavior of each agent is characterized by the following objectives:

1. Reach a preset goal state $x_{g,i}$ within the horizon T.
2. Avoid collisions with other players.
3. Minimize control effort.
4. Prefer low velocities.

As some of the aspects of this behavior (crucially collision avoidance) depend implicitly or explicitly on the joint state of at least two agents, this problem requires each player to reason about the decisions of other players.

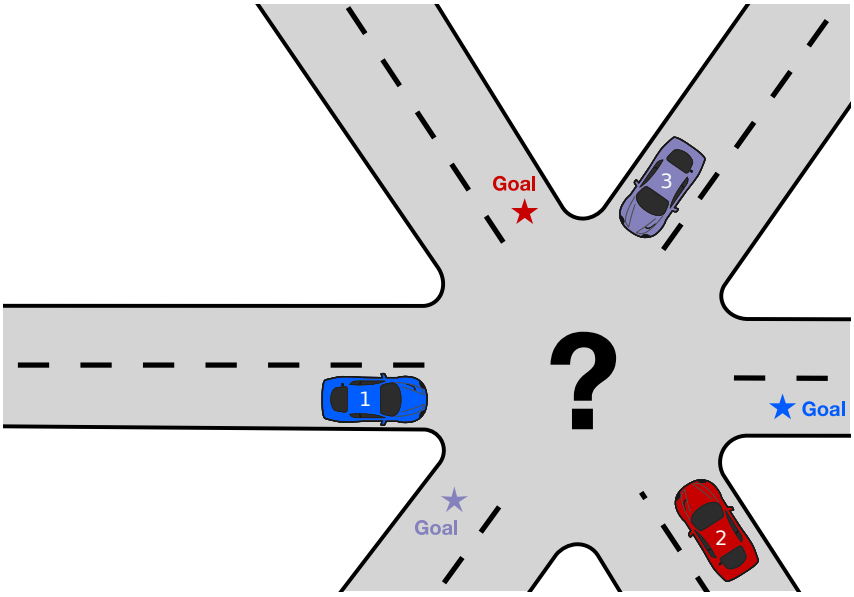


Figure 4.2.1: Schematic visualization of the running example. Each road user wishes to reach their goal on the other side of the intersection. The autonomous agent in this example controls the blue car starting on the left.

4.2.2 Game Formulation of a Human-Robot Interaction Problem

The interaction problem of a robot with $N - 1$ humans is cast as an N-player general-sum differential game as intruded in Section 2.1.1. Recall that in this formulation the evolution

of the joint state, x, of the environment is characterized by the dynamics,

$$\dot{x} = f(t, x, u_{1:N}) \tag{4.2.1}$$

the objectives are given by time-separable costs,

$$J_i\left(u_{1:N}(\cdot)\right) \triangleq \int_0^T g_i\left(t, x(t), u_{1:N}(t)\right)\, dt, \forall i \in [N]\,, \tag{4.2.2}$$

and each player seeks to find feedback strategy,

$$u_i(t) \triangleq \gamma_i(t, x(t))\,, \tag{4.2.3}$$

that minimizes their respective cost function. Furthermore, this work uses the convention that the autonomous agent corresponds to index 1, i.e. the robot is in control of input u_1 and seeks to minimize cost J_1.

Using this formulation, the decision model makes the following assumption. The decision process of human agents is modeled to attain an *approximate local Nash* equilibrium (c.f. Section 2.1.4) of the above game. By this choice of equilibrium concept the decision model allows humans to play suboptimal. While at first glance this may appear like a poor model for human decision making, it is argued that human players may not always be able to identify a global equilibrium; that is, if the interaction problem is sufficiently complex, human players are likely to utilize some form of approximation to tractably find an optimized strategy. Furthermore, it should be noted that global equilibria are a subset of local equilibria. Hence, when considering the set of local equilibria to predict human strategies, globally optimal play is implicitly contemplated as well. Conversely, if only global equilibria were allowed, the decision model would be less expressive and the autonomous agent using the proposed approach would not be able to discover and appropriately respond to suboptimal behavior.

Running Example. The running example introduced in Section 4.2.1 can be cast as an N-player general-sum differential game by specifying system dynamics and costs for all players.

Defining the joint state to be $x = [x_1; x_2; x_3]$, let the dynamics of each agent be modeled as those of a 4D unicycle

$$\dot{x}_i = \left[\dot{p}_{x,i}; \dot{p}_{y,i}; \dot{\theta}_i; \dot{v}_i\right] = \left[v_i \cos\theta_i; v_i \sin\theta_i; \omega_i; a_i\right]. \tag{4.2.4}$$

Here, agent i has position $p_i \triangleq (p_{x,i}, p_{y,i})$, orientation θ_i and velocity v_i and is in control of the longitudinal acceleration a_i as well as the steering rate ω_i of their respective subsystems.

For each player a nonquadratic cost function is formulated to encode the preferences given in Section 4.2.1 as the sum of the following:

$$\text{goal: } c_{\text{goal},i}\mathbf{1}(t > T - t_{\text{goal}})\|p_i - p_{\text{goal},i}\|^2 \tag{4.2.5}$$

$$\text{proximity: } c_{\text{prox},i}\mathbf{1}(\|p_i - p_j\| < d_{\text{prox}})(d_{\text{prox}} - \|p_i - p_j\|)^2 \tag{4.2.6}$$

$$\text{input: } c_{\text{input},i}u_i^T u_i \tag{4.2.7}$$

$$\text{velocity: } c_{\text{vel},i}v_i^2 \tag{4.2.8}$$

Here, $c_{\text{goal},i}, c_{\text{prox},i}, c_{\text{input},i}, \text{and} c_{\text{vel},i}$ are parameters that weight the importance of the respective components of the cost for player i, and $\mathbf{1}(\cdot)$ denotes the indicator function as defined in Equation (2.2.8). The distance of player i to the goal position $p_{\text{goal},i}$ is penalized for the last t_{goal} time units, and d_{prox} denotes the distance threshold at which proximity cost is activated.

By solving this problem to an approximate local Nash equilibrium of the game, the decision model encodes shared responsibility for collision avoidance between all players. Furthermore, as will be demonstrated later in this chapter, this example allows multiple equilibria due to the structure of the cost. For each pairwise encounter of two players the involved agents need to decide on which side they pass each other. While for some cases this conflict may be trivially resolved because both players have a strong preference for the same solution, other scenarios require some form of negotiation to resolve the ambiguity.

4.2.3 Strategy Alignment Problem

The discussion in Section 4.1 emphasizes that an agent using a game-theoretic planning model may face the challenge of selecting among multiple solutions and the running example introduced above provides an interpretation of this issue in a real world example. To address this issue, the agent is tasked solve the *strategy alignment* problem proposed hereafter.

First, it is assumed that there exists a local equilibrium solution of the game that determines the behavior of all humans. This agreement between humans can be thought of as the innate ability of humans to communicate through subtle cues that are difficult for robots to pick up on. However, humans can not be expected to be able to communicate with robots with similar clarity. Therefore, it is the robot's task to *infer* the equilibrium negotiated by the humans and *align* to their preferred local solution. By this means, the strategy alignment problem preserves local symmetry between players but allows humans to take a leading role in globally selecting a preferred equilibrium.

In accordance with these assumptions, mathematically, the strategy alignment problem is modeled as an inference problem (c.f. Section 2.2) in which the local equilibrium at which human players operate is a latent state, denoted k, that the agent can not observe directly. Instead, the agent receives observations of only the physical state, x. Using these observations, the agent can maintain a belief over the latent equilibrium state, i.e.

$$b_t(k_t) \triangleq p(k_t | x_{1:t}, u_{1,1:t}). \tag{4.2.9}$$

Note that in contrast to the formulation in Section 2.2.1, due to the structure of the problem the likelihood of the latent state, k_t, at time t depends not only upon past observations, $x_{1:t}$, but also on the decision of the agent, i.e. its inputs, $u_{1,1:t}$, to the system up to time t.

A dynamic decision network (DDN), i.e. a DBN with a decision node for the agent's action, that depicts the conditional independence assumptions for the equilibrium inference problem is shown in Figure 4.2.2. In accordance with the problem formulation, the agent's decision at time step $t+1$, $u_{1,t+1}$, is informed by past observations of the physical state,

$x_{1:t}$, while the equilibrium, k_t, generating the human behavior remains unobserved. At every time step, the next equilibrium, k_{t+1}, depends only upon the previous equilibrium, k_t, and the physical state, x_t. This dependence models the fact that human players update their strategies to account for the decision of the autonomous agent. Furthermore, $\hat{x}_{t+1}$ represents the predicted physical state that would be attained if humans behaved exactly according to the equilibrium strategy corresponding to k_{t+1} and there was no model mismatch in the dynamics. Finally, to account for deviations from the exact equilibrium strategy, the true physical state, x_{t+1}, is modeled as a probabilistic emission of the nominal prediction, $\hat{x}_{t+1}$.

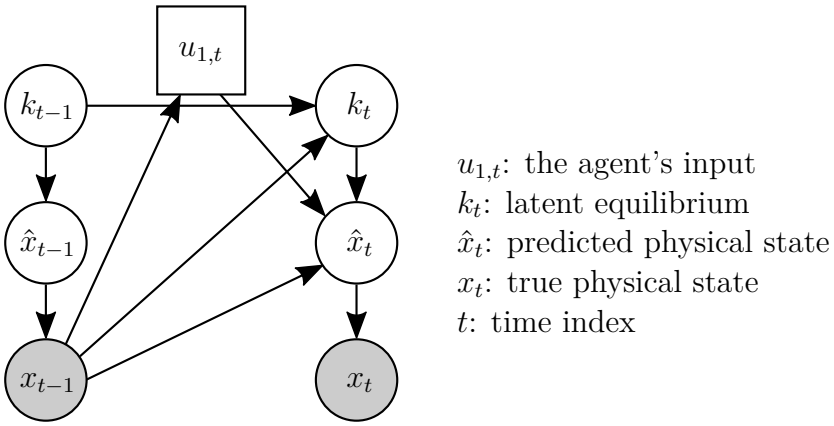


Figure 4.2.2: DDN used to model the equilibrium inference problem.

Running Example. By solving the strategy alignment problem in the running example introduced in Section 4.2.1, the robot allows humans to choose their preferred local solution. At the same time, since the agent only considers noncooperative equilibrium strategies, it maintains shared responsibility for collision avoidance. Furthermore, by inferring the equilibrium from observations of the human actions over time, the agent avoids making a priori assumptions about the equilibrium selection preferences of humans, e.g. ruling out payoff dominated or risk dominated solutions. This is particularly important as Section 4.1 highlights the challenges of modeling such preferences. For example, when approaching a human that comes straight at the robot, from a robots perspective it is reasonable to model humans as having symmetric preferences for passing on either side, if neither solution is risk or payoff dominant. However, humans may have cultural or otherwise difficult-to-model preferences for a specific solution.

4.3 Inference-Based Strategy Alignment

This section presents a general framework for tractable solution of the strategy alignment problem defined in Section 4.2. This approach is developed from the perspective of a fully observed differential game and uses ILQG to approximate local Nash equilibria of the game. By seeding the ILQG method with numerous samples from a prior distribution over initial strategy profiles, the method induces a distribution of local equilibria. At every

time step, the belief over latent equilibria is then updated with observations of the physical state using a particle filtering technique. Using this a-posteriori belief, the agent's strategy is aligned by extracting the most likely equilibrium.

4.3.1 Finding Local Equilibria

The strategy alignment approach presented in this section is, for the most part, agnostic to the solver used to compute local equilibria. There are two requirements for a solver to be used in this new framework. First, it must admit a form of persistent state that encodes a specific equilibrium, i.e. a representation of the latent equilibrium state, k. Second, because inference must be run at near real-time planning rates, the solver needs to be sufficiently fast to allow solving the game for multiple equilibria at every planning step.

Regarding the first point of latent equilibrium representation there are multiple choices of suitable local game solution algorithms. Local solution techniques typically admit a latent state representation via the initial seeding of the solver [1, 26, 45]. By seeding the solver with a given initial strategy profile, the equilibrium whose basin of attraction contains this seeding is recovered. Conversely, any strategy profile in the basin of attraction of a specific equilibrium, k, is a permissible representation for the latter.

From the perspective of runtime, however, only few solution techniques pose a suitable choice. Cleac'h *et al.* [45] report run times for their local solver in the order of multiple seconds even for a relatively small problems with 3 players. Spica *et al.* [26] and related approaches [43, 44] report significantly faster planning times ranging from 50 ms for a 2-player example to about 400 ms for a 6-player example. Finally, ILQG as implemented by Fridovich-Keil *et al.* [1] achieves planning times below 17 ms for a 3-player example.

In view of these results, this work uses ILQG for solution of the local equilibrium problem. A thoroughly optimized Julia implementation of this solver is made to further improve the runtime (c.f. Chapter 3). An additional advantage of ILQG is that it synthesizes *feedback* strategies for each player while Spica *et al.* [26] only generate *open-loop* controls. Thus, ILQG can achieve a more fine-grained prediction beyond the resolution of the replanning rate by invoking the local feedback laws between updates of the solution.

4.3.2 Inferring the Equilibrium

Since the robot does not know the equilibrium that humans have chosen, it can only attempt to infer how likely each equilibrium is. As the decision model assumes that the agent is able to measure the state perfectly, in a perfect mathematical abstraction, there would be a unique trajectory for each equilibrium, and the robot could eliminate solution hypotheses immediately if they did not match the observation exactly. However, in reality, there are multiple sources of uncertainty that have to be considered.

First, real human behavior can not be expected to exactly match an equilibrium strategy. Instead, they follow trajectories that are difficult to model perfectly but may approximately match the equilibrium. Second, the model used for the dynamics of the environment and

the costs for each player will always be an abstraction of the real-world dynamics and preferences of humans. Finally, there is some numerical noise in the game solution process. Thus, even if the model of dynamics and costs matched the real world perfectly, strategy profiles attained for a specific equilibrium with a local algorithm like ILQG may deviate within the tolerance of the convergence criterion.

Because of these sources of this uncertainty, Bayesian inference is used to maintain the equilibrium belief formulated in Equation (4.2.9).

Challenges for Exact Bayesian Inference

By exploiting the conditional independence assumption made by the strategy alignment model (c.f. Figure 4.2.2) the exact Bayesian update rule for Equation (4.2.9) can be derived. Following the procedure discussed in Section 2.2.1, the update can be written as

$$b_t(k_t) \propto \int_{k_{t-1}\in\mathcal{K}} \int_{\hat{x}_t\in\mathcal{X}} p(x_t|\hat{x}_t)p(k_t,\hat{x}_t|u_{1,t},k_{t-1},x_{t-1})b_{t-1}(k_{t-1})\,d\hat{x}_t\,dk_{t-1}. \tag{4.3.1}$$

Here, $\mathcal{X}$ is the space of physical states, x, and $\mathcal{K}$ is the latent space of equilibria, k.

Unfortunately, evaluating this update rule is computationally challenging for multiple reasons. First, enumerating all equilibria in the equilibrium space, $\mathcal{K}$, is generally intractable [41]. Second, an equilibrium in $\mathcal{K}$ is a complicated concept for which it is hard to find compact representation. In many cases, $\mathcal{K}$ can be expected to contain a finite number of equilibria. However, these equilibria change, arise, and vanish over time, depending on the physical state, x. As a result, while $\mathcal{K}$ typically is intrinsically low-dimensional and countably finite, the inability to enumerate equilibria a priori necessitates an indirect representation, e.g. here via a strategy profile in its basin of attraction (c.f. Section 4.3.1). Unfortunately, this representation, i.e. the space of *joint strategies*, is extrinsically high-dimensional and thus exact integration remains intractable.

Tractable Inference via Particle Filtering

While enumerating *all* equilibria in the equilibrium space $\mathcal{K}$ can not be done efficiently, one can recover a subset of $\mathcal{K}$ by sampling initial strategy profiles and solving the game at these points. Using this idea, a particle filtering technique is proposed to approximate the Bayesian update of the equilibrium belief given in Equation (4.3.1). This particle filtering technique is an adaptation of the algorithm presented in Section 2.2.2 to account for the structure of the DDN given in Figure 4.2.2.

Algorithm 3 summarizes the procedure of generating an initial belief for the particle filter. The algorithm proceeds in the following steps:

1. **Sample Strategy Profiles** First, K seed strategies are randomly sampled from a problem-specific distribution, $\mathcal{P}_\gamma$ (line 2 of Algorithm 3).

2. **Solve Game Instances** The game is solved at each of the sampled seeds to recover K equilibria. Each of the resulting equilibrium strategy profiles is referred to as

a particle which has a corresponding weight that is initially set to 1 (line 3 of Algorithm 3).

3. **Combine Duplicates** Some of the initial strategy profiles sampled from $\mathcal{P}_\gamma$ may result in the same solution. Thus, the weights for particles that represent the same equilibrium, as determined by measuring the distance between the equilibrium trajectories, are combined (line 4 of Algorithm 3).

In this algorithm, the seed distribution, $\mathcal{P}_\gamma$, is necessarily dependent on the specific state space and dynamics of the problem. However, as will be demonstrated in Section 4.4.2, it was not difficult to find suitable distributions for the experiments conducted here. In general, the seed distribution must cover the strategy space in a manner that allows to recover the relevant equilibria that human players may consider. Equilibria that are not attained by the solver for any sampled seed can not be inferred. Consequently, the performance of the proposed particle filtering technique depends on the ability to specify a suitable seed distribution.

Algorithm 3 Generation of an initial particle belief for equilibrium inference from a seed distribution $\mathcal{P}_\gamma$.

1: **procedure** GENERATEINITIALBELIEF($\mathcal{P}_\gamma$)
2: $\quad\{\bar{\gamma}_0^{(k)}\}_{k\in[K]} \leftarrow$ sample K strategy profiles from $\mathcal{P}_\gamma$
3: $\quad\bar{\mathcal{B}}_0 \leftarrow \{(\gamma_0^{(k)} \equiv \text{SOLVEGAME}(x_0, \bar{\gamma}_0^{(k)}), w_0^{(k)} \equiv 1)\}_{k\in[K]}$
4: $\quad\mathcal{B}_0 \leftarrow \text{COMBINEDUPLICATES}(\bar{\mathcal{B}}_0)$
5: $\quad$**return** $\mathcal{B}_0$
6: **end procedure**

Starting from the initial belief, the belief is updated using the particle filtering technique outlined in Algorithm 4. This algorithm proceeds in the following steps:

1. **Transition** The strategy profile, $\gamma_t^{(k)}$, associated with particle k is updated by re-solving the game from the last known physical state, x_{t-1}. With this updated strategy profile, each particle makes a prediction of the physical state, $\hat{x}^{(k)}$, by applying the human strategies corresponding to equilibrium k as well as the agent's input, $u_{1,t}$. From the perspective of a vanilla particle filter the process of re-solving the game and predicting the state constitutes a generative model that implicitly defines the transition model $p(k_t, \hat{x}_t | u_{1,t}, k_{t-1}, x_{t-1})$ in Equation (4.3.1) . These steps correspond to line 4 and 5 of Algorithm 4.

2. **Observation Weighting** The weight for each particle is then updated by evaluating the probability density $p(x_t | \hat{x}^{(k)})$ which captures potential deviations of humans from the exact strategy corresponding to equilibrium k (line 6 of Algorithm 4).

3. **Combine Duplicates** Following the logic discussed above for Algorithm 3, the weights for particles that represent the same equilibrium are combined (line 9 of Algorithm 4).

In this algorithm, the call to SolveGame corresponds to an invocation of ILQG.

It should be noted that the deviation model, $p(x_t|\hat{x}^{(k)})$, is somewhat arbitrary because it is meant to capture the three difficult-to-model sources of uncertainty mentioned above. In this work, a Gaussian distribution is used, as is commonly done in cases where uncertainty is difficult to model, but is expected to be unimodal. It would also be possible to incorporate domain knowledge into the algorithm by using a more complex distribution.

Algorithm 4 Particle filter for equilibrium inference.

procedure UpdateEQBelief($\mathcal{B}_{t-1} \equiv \{(\gamma_{t-1}^{(k)}, w_{t-1}^{(k)})\}_{k\in[K]}, x_{t-1}, x_t, u_{1,t}$)
  $\bar{\mathcal{B}}_t \leftarrow \emptyset$
  for particle index $k = 1 \dots K$ **do**
    $\gamma_t^{(k)} \leftarrow \text{SolveGame}\left(x_{t-1}, \gamma_{t-1}^{(k)}\right)$
    $\hat{x}_t^{(k)} \leftarrow x_{t-1} + \int_{t-1}^{t} f\left(\tau, x(\tau), u_{1,t}, \gamma_{-1}^{(k)}(x(\tau))\right) d\tau$
    $w^{(k)} \leftarrow w^{(k)} p(x(t)|\hat{x}^{(k)})$
    $\bar{\mathcal{B}}_t \leftarrow \bar{\mathcal{B}}_t \cup (\gamma_t^{(k)}, w_t^{(k)})$
  end for
  $\mathcal{B}_t \leftarrow \text{CombineDuplicates}(\bar{\mathcal{B}}_t)$
  return $\mathcal{B}_t$
end procedure

Running Example. Figure 4.3.1 schematically shows the idea of a particle based belief representation for the running example introduced in Section 4.2. Since duplicate particles are combined, each particle in the belief maintained by the autonomous agent (Player-1, blue) corresponds to a *different* equilibrium, i.e. a set of feedback strategies that the players may invoke to reach their goal while avoiding collisions with others. Here, the weight of each particle is visualized via its size and corresponds to its estimated likelihood. This likelihood estimate may change over time as a result of two different mechanisms in the belief update: first, via observation weighting of physical state measurements, and second, if two particles transition to the same equilibrium and hence their weights are combined. In the running example, weight changes of the latter kind can occur whenever agents have fully passed each other on the way to their respective goals. Here, the reason for merging is the fact that for the later part of the maneuver, i.e. when agents are close to their goal, each player's optimal plan does not depend upon the decisions of others and the problem is essentially reduced to three decoupled single-player optimal control problems.

4.3.3 Globally Aligned Closed-Loop Planning

After the inference algorithm has weighted each of the sampled solutions, the robot must decide which feedback strategy to apply. There are many possible candidates for the best strategy. If the weights are appropriately normalized, then they define a probability

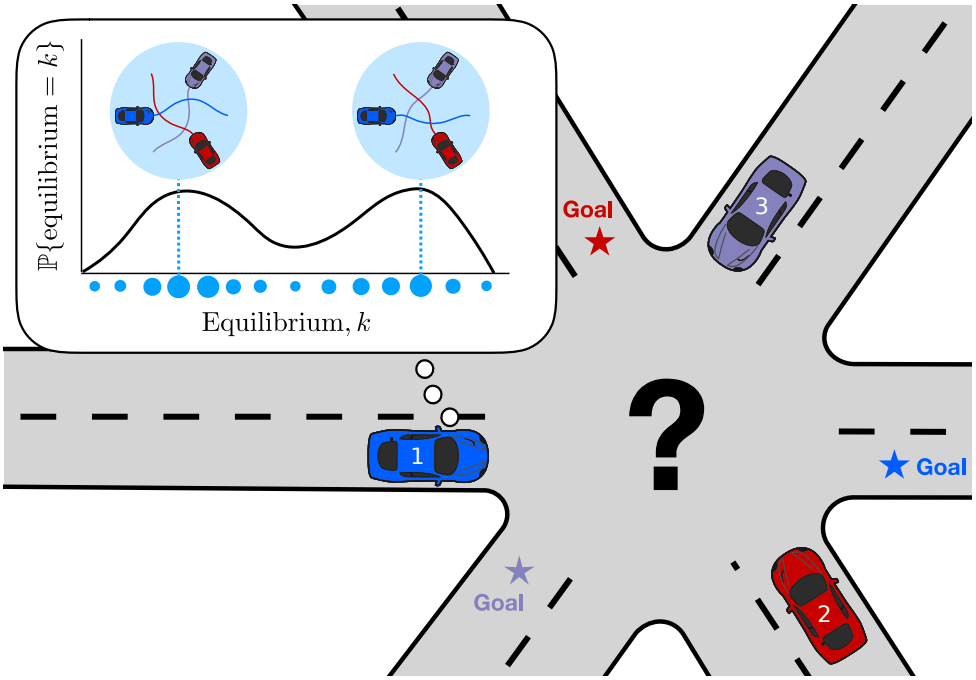


Figure 4.3.1: Schematic visualization of a particle base belief representation for the running example introduced in Section 4.2.

distribution over the sampled equilibrium particles, i.e.

$$P(k) = \frac{w^{(k)}}{\sum_{k \in [K]} w^{(k)}}. \tag{4.3.2}$$

This structure is analogous to a belief in a POMDP formulation [54, 75] where the human player's choice of equilibrium is the latent part of the state. Thus, solution concepts used for POMDPs are applicable here.

It is well known that to find the optimal solution of a POMDP, the agent must reason about the information they will receive in the future. However, this is computationally intractable in general [76], so approximations are usually employed. Commonly used approximations include generalized QMDP [4, 77] and planning assuming the most likely or mean latent state [4]. In the present setting, QMDP is difficult to apply because it is difficult to evaluate every possible control input against all of the strategies the other players might take. The mean latent state in this setting corresponds to the weighted average of several equilibrium strategies, which is usually not a strategy that a human would take. Maximum likelihood approximations involve little computational effort and previous work has demonstrated reasonable performance of such approaches in scenarios with moderate uncertainty [4, 78].

For these reasons, here, the maximum a posteriori (MAP) estimation of the equilibrium belief is used to make control decisions. Algorithm 5 summarizes the procedure for closed-loop planning with strategy alignment to the MAP equilibrium. Starting from the initial belief generated via the seed distribution $\mathcal{P}_\gamma$, at every time step the agent applies the control specified by its strategy in the most likely equilibrium. Subsequently, the agent

observes the resulting physical state which also depends upon the actions of the other players that follow an unobserved equilibrium of the game. Finally, the agent performs a belief update via the procedure outlined in Algorithm 4 to infer which equilibrium is likely to have generated the observed behavior. This updated belief is used to inform the next decision.

Algorithm 5 MAP Aligned Control

$\mathcal{B}_0 \leftarrow$ GenerateInitialBelief$(\mathcal{P}_\gamma)$
for each time step t **do**
    $k_{\text{MAP}} \leftarrow \underset{k \in [K]}{\text{argmax}} \left\{ w_{t-1}^{(k)} \text{ for particle index } k \text{ of } \mathcal{B}_{t-1} \right\}$
    $u_{1,t} \leftarrow \gamma_1^{(k_{\text{MAP}})}(x(t), t)$
    $x_t \leftarrow$ apply $u_{1,t}$ and receive observation x_t
    $\mathcal{B}_t \leftarrow$ UpdateEQBelief$(\mathcal{B}_{t-1}, x_{t-1}, x_t, u_{1,t})$
end for

4.4 Evaluation

This section analyzes the performance of inference-based strategy alignment by comparing the proposed equilibrium inference to a game-theoretic baseline using [1] that does not employ inference. The performance of both approaches is evaluated in simulations of the running example introduced in Section 4.2.

This section is structured as follows. Section 4.4.1 presents the implementation details for this evaluation and gives an overview over the parameters used throughout the simulated experiments. Thereafter, three types of evaluation experiments are conducted for which the results are presented and discussed in the following order. Section 4.4.2 demonstrates the multiplicity of solutions for the evaluation problem by performing a Monte-Carlo study of local equilibria. Section 4.4.3 evaluates the prediction performance of the equilibrium inference approach by isolating it from the control part in a purely observing setting. Finally, Section 4.4.4 studies the interaction dynamics and performance of the strategy alignment approach in a closed-loop planning scenario.

Note that each of these sections contains a discussion of the respective results, so that findings from earlier, conceptually more intuitive evaluations can assist in the interpretations of more complex experiments towards the end of this chapter. A summary of the main results as well a discussion of limitations and potential next steps is given in Section 4.4.5.

4.4.1 Experiment Details and Parameters

Evaluation Problems

Throughout this section, evaluations are performed on variants of the running example introduced in Section 4.2 with 2, 3 and 5 players, respectively. For each benchmark problem

a game formulation as defined by the tuple of dynamics (Equation (4.2.4)) and costs (Equations (4.2.5) to (4.2.8)) is implemented in the iLQGames.jl framework. Table 4.1 shows the weights and parameters for the cost terms given in Equations (4.2.5) to (4.2.8) as well as the time discretization and game horizon used for these experiments.

Table 4.1: Game parameters.

Parameter	Symbol	Value
Goal penalty start time	t_{goal}	9.9 s
Proximity cost activation distance	d_{prox}	1.2 m
Goal cost weight	c_{goal}	300
Proximity cost weight	c_{prox}	50
Input cost weight	c_{input}	10
Velocity cost weight	c_{vel}	30
Time step	Δt	0.1 s
Game horizon	T	10 s

Inference Algorithm

The algorithms for equilibrium inference and strategy alignment are implemented in the Julia programming language. These implementations closely follow the pseudo code representations in Algorithms 3 to 5. The equilibrium inference algorithm is implemented using `ParticleFilters.jl`[1] which is adapted to accommodate the CombineDuplicates procedure. Within this framework, the step of computing the game solution for each particle transition is realized via the Julia implementation of ILQG presented in Chapter 3.

To generate initial strategy profiles for this class of benchmark problems, open-loop strategies of the form

$$\gamma_{i,0}(t, x) = [\beta_\omega cos(t/T\pi); \beta_a cos(t/T\pi)], \tag{4.4.1}$$

with turn rate parameter, β_ω, and acceleration parameter, β_a, are used for all players. For each parameter combination, this initial strategy generates a different S-shaped path for the player. The seed distribution, $\mathcal{P}_\gamma$, for the evaluation problems is defined as the joint distribution over the N-tuple of parameterized open-loop strategies where the parameters β_ω and β_a are sampled independently from uniform distributions, $\mathcal{U}(\beta_{\omega,\min}, \beta_{\omega,\max})$ and $\mathcal{U}(\beta_{a,\min}, \beta_{a,\max})$, respectively. Table 4.2 shows the parameters of the seed distribution, $\mathcal{P}_\gamma$, used for the experiments conducted here.

[1] `https://github.com/JuliaPOMDP/ParticleFilters.jl`

Table 4.2: Seed distribution parameters.

Parameter	Symbol	Value
Min acceleration	$\beta_{a,\min}$	$1.5\,\mathrm{m\,s^{-2}}$
Max acceleration	$\beta_{a,\max}$	$2.5\,\mathrm{m\,s^{-2}}$
Min turn rate	$\beta_{\omega,\min}$	$-0.2\,\mathrm{rad\,s^{-1}}$
Max turn rate	$\beta_{\omega,\max}$	$0.2\,\mathrm{rad\,s^{-1}}$

Finally, within the equilibrium inference algorithm the deviation density, $p(x_t|\hat{x}^{(k)})$, is modeled as a multivariate Gaussian, $\mathcal{N}(\hat{x}^{(k)}, \epsilon_O \mathbf{I})$. The observation noise parameter, ϵ_O, for this distribution as well as the number of particles used for each problem are given in Table 4.3.

Table 4.3: Particle filter parameters.

Parameter	Symbol	Value
Number of particles (3-player)	K_3	50
Number of particles (5-player)	K_5	150
Observation noise	ϵ_O	0.1

4.4.2 Monte Carlo Study of Local Equilibria

This section analyzes the qualitatively different local equilibria that exist in a multi-player navigation problem. For each of the experiments presented in this section, first, a collection of equilibria is generated by solving the multi-player game for 100 random samples from the seed distribution $\mathcal{P}_\gamma$. Subsequently, spatial clustering is performed on the state trajectories traced out for each sample to discern the different types of encounter geometries. Here, clustering is performed using the DBSCAN algorithm [79] to avoid manual specification of the number of clusters [80].

Minimal Example: 2-Player Scenario

For the sake of clarity, first, a 2-player version of the navigation problem is considered.

Figure 4.4.1 shows the two clusters of solutions that can be found in this setting as well as the initial strategy profiles sampled from $\mathcal{P}_\gamma$ that generate these clusters. Here, Player-1's trajectory is shown in blue, while Player-2's trajectory is shown in red. Additionally, the clusters are sorted from left to right in ascending order of cost incurred by Player-1. To give an intuition for time, for each player trajectory the initial positions as well the position closest to the other player are highlighted with circular marks.

This simple example shows that the game allows for two qualitatively different equilibria. In the equilibrium corresponding to the first cluster Player-1 (blue) accelerates while Player-2 (red) slowly approaches the conflict area to let the other player pass first. Additionally, both players slightly deviate from the straight path to their respective goal to make room for the other player as they share responsibility for collision avoidance. In the equilibrium corresponding to the second cluster the players take opposite roles. The examination of the cost for each cluster reveals that the player passing the conflict area first incurs a slightly lower cost than the player who is forced to decelerate and wait at the intersection. Thus, it is clear that neither equilibrium is payoff dominant. Rather, each cluster corresponds to different plausible mode of interaction.

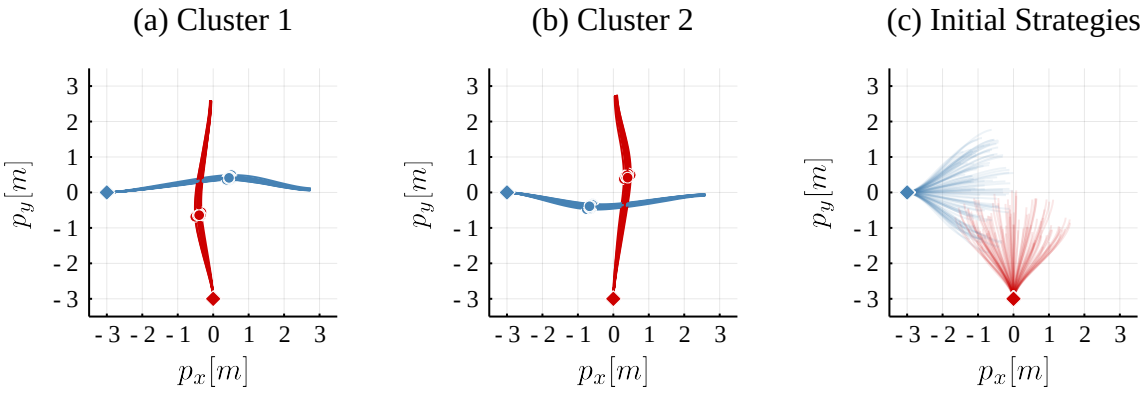


Figure 4.4.1: (a,b) Clustered local equilibria for a 2-player navigation problem. Initial positions and positions closest to the other player are highlighted, respectively, with rectangular and circular marks. (a) blue goes first; (b) red goes first. (c) Initial strategy samples from $\mathcal{P}_\gamma$.

This simple example demonstrates that even in the minimally complex case of a 2-player encounter a unique dominating strategy profile may not be found and thus some form of strategy alignment is required. In order to safely navigate the intersection, each agent must understand which equilibrium the other player is aiming for. If players fail to agree on an equilibrium, multiple players may incur a high cost, e.g. if two players accelerate to pass the conflict area first.

Running Example: 3-Player Scenario

A third player is added to the navigation problem to recover the original running example as introduced in Section 4.2. The clustered local equilibria for this scenario are depicted in Figure 4.4.2. Here, the third player's trajectories are shown in purple. Again, clusters are ordered from left to right and top to bottom in ascending order of cost incurred by Player-1 (blue). The clustering reveals a total number of eight qualitatively different local equilibria. The two solutions with the lowest cost for all players — shown in the first two sub-figures of Figure 4.4.2 — correspond to a clockwise or counter-clockwise circular motion in which all players equally deviate from their straight path to the goal while

maintaining almost constant speed. The remaining solutions can be understood as all possible sequential orders in which players can pass the conflict zone. That is, in each of these cases one player accelerates on the straight path to its goal to pass the conflict zone first. Another player follows with moderate speed and slightly deviates from its straight path to the goal to avoid a collision. The last player approaches the conflict zone at low speed and waits for the others to pass before continuing on a straight path to its goal.

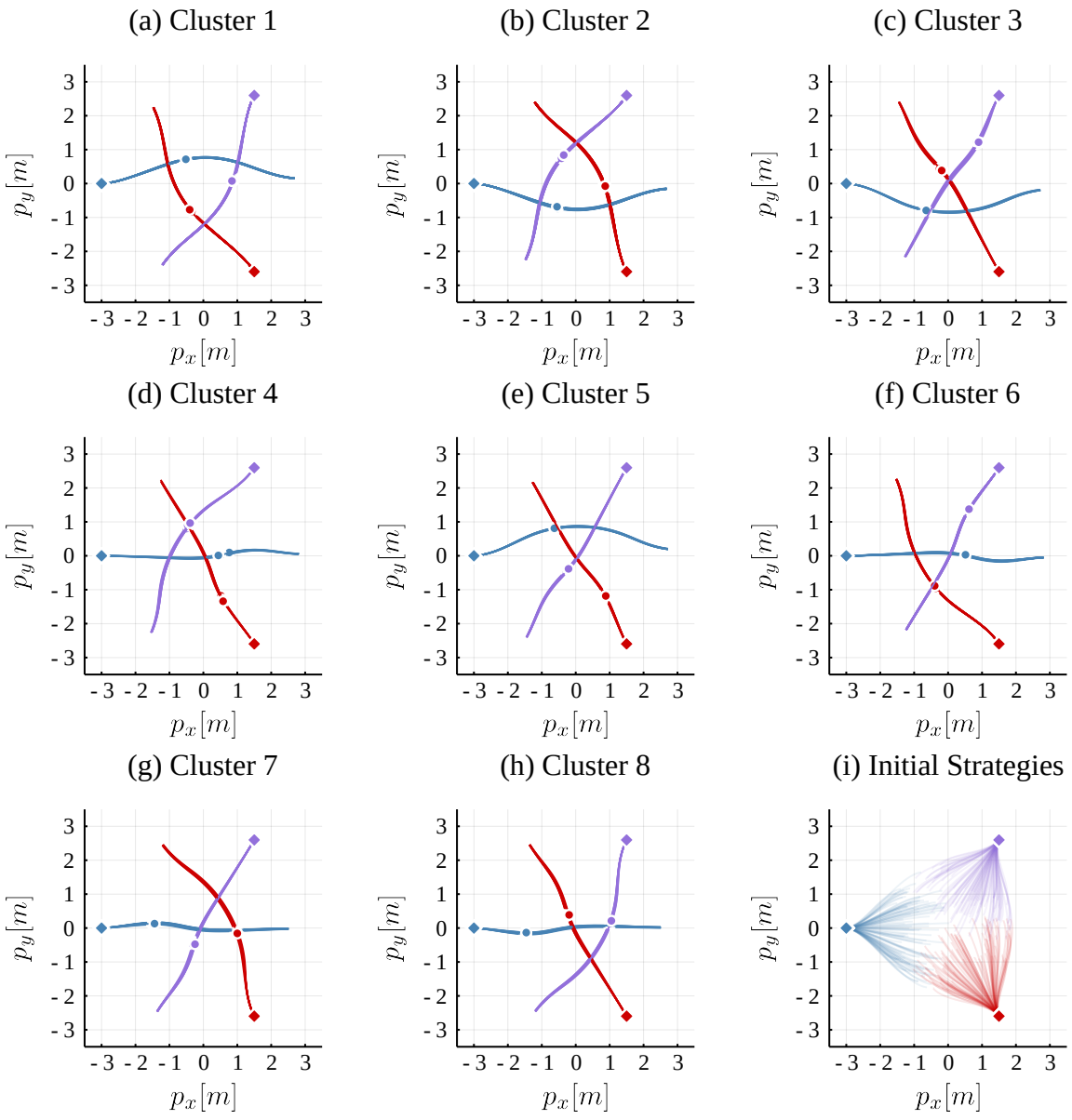


Figure 4.4.2: (a-h) Clustered local equilibria for a 3-player navigation problem. Initial positions and positions at half the simulation horizon highlighted, respectively, with rectangular and circular markers. (i) Initial strategy samples from $\mathcal{P}_\gamma$.

The examination of qualitatively different approximate local Nash equilibria for the 3-player navigation scenario shows that the problem immediately becomes more complex,

merely through the number of combinations in which the players may pass the conflict zone. Therefore, it is clear that with increasing number of players hand-specifying suitable behaviors for all possible cases is undesirable and thus a more principled approach should be taken.

4.4.3 Prediction Performance

In this section the performance of the proposed inference method is examined independently from closed-loop interaction dynamics. For this purpose, again, the running example of a 3-player navigation problem is considered. However, in this experiment the agent does not control any inputs to the system but rather observes the interaction of three players as they follow a set of strategies corresponding to a local equilibrium. At every time step, the agent receives an observation of the physical state x. With this information, it is the agent's task to accurately predict the trajectories of all players over the remaining game horizon.

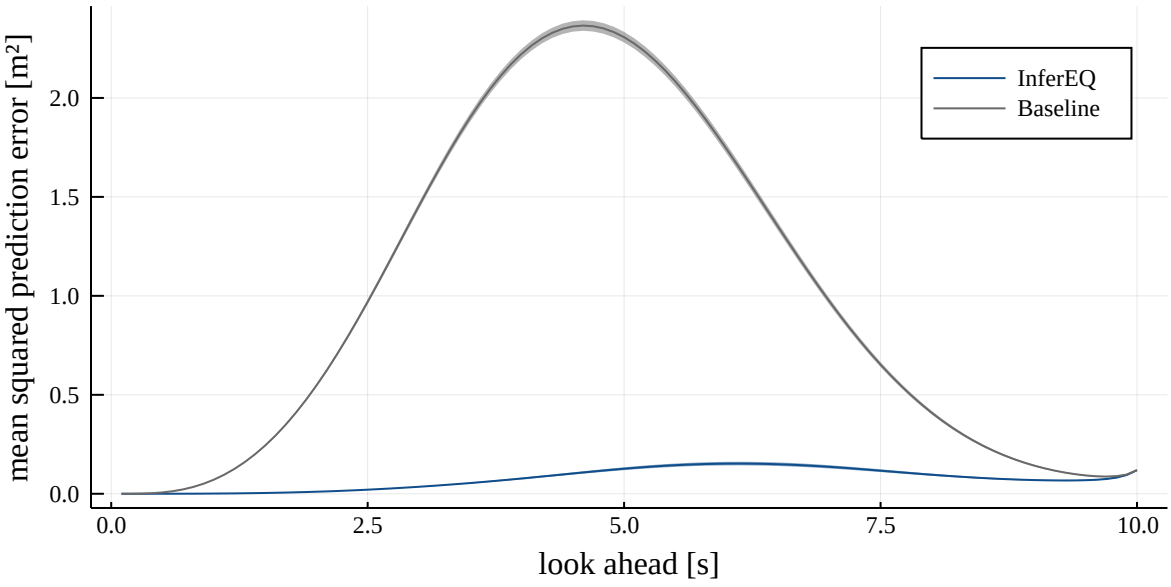
The equilibrium inference method proposed in Section 4.3.2 is compared to a baseline that uses ILQG but does not actively reason about different solutions. Instead, it uses a fixed local equilibrium to predict the trajectories of all players. For the baseline, the seed that generates the local equilibrium is sampled from the same seed distribution, $\mathcal{P}_\gamma$, as used by the equilibrium inference approach. Note that while the baseline does not actively reason about the equilibrium that the players operate at, it still benefits from state-feedback as it re-solves the game for the remaining game horizon after every observation of the physical state. Each method is evaluated in 200 simulations of the 3-player running example where for each run the simulated human behavior is generated by sampling a seed from $\mathcal{P}_\gamma$.

Figure 4.4.3a shows the mean squared prediction error $E\left[||p - \hat{p}||^2\right]$ of the position $p = [p_{x,1}; p_{y,1}; p_{x,2}; p_{y,2}; p_{x,3}; p_{y,3}]$ over the moving prediction horizon for the inference method and the baseline. Note that in this evaluation, prediction errors are grouped by the *time offset* from the respective current simulation time; i.e. the data point at time t_p for a specific method corresponds the average over all predictions made by this method looking t_p time units into the future. Since the agent receives an exact observation of the physical state at each time step and furthermore knows the goal locations $x_{g,i}$ of all players, both methods achieve a low prediction error at the beginning and the end of the prediction horizon. However, in the intermediate range, which is crucial to predict how conflicts are resolved, the inference approach is able to significantly reduce the prediction error.

To provide further insight into the prediction performance, Figure 4.4.3a shows the evolution of the mean squared prediction error over the simulation horizon instead of the prediction horizon. Here, in contrast to the previous evaluation, each point on the graph corresponds to the averaged prediction error for all predictions made *at* a specific time step of the simulation. In this evaluation, the prediction error for both approaches is initially high. However, as the interaction of players is observed over time the equilibrium inference approach is able to quickly reduce the prediction error and after 1.5 s predicts the player trajectories almost perfectly for the rest of the game. The baseline, on the other hand, maintains a high average prediction error until 4 s. At around half the simulation

horizon the ambiguity of multiple equilibria is trivially resolved by the fact that all initial strategies now converge to the same equilibrium which corresponds to the solution of three fully decoupled single-player optimal control problems.

(a) Error statistics over prediction horizon.



(b) Error statistics over simulation horizon.

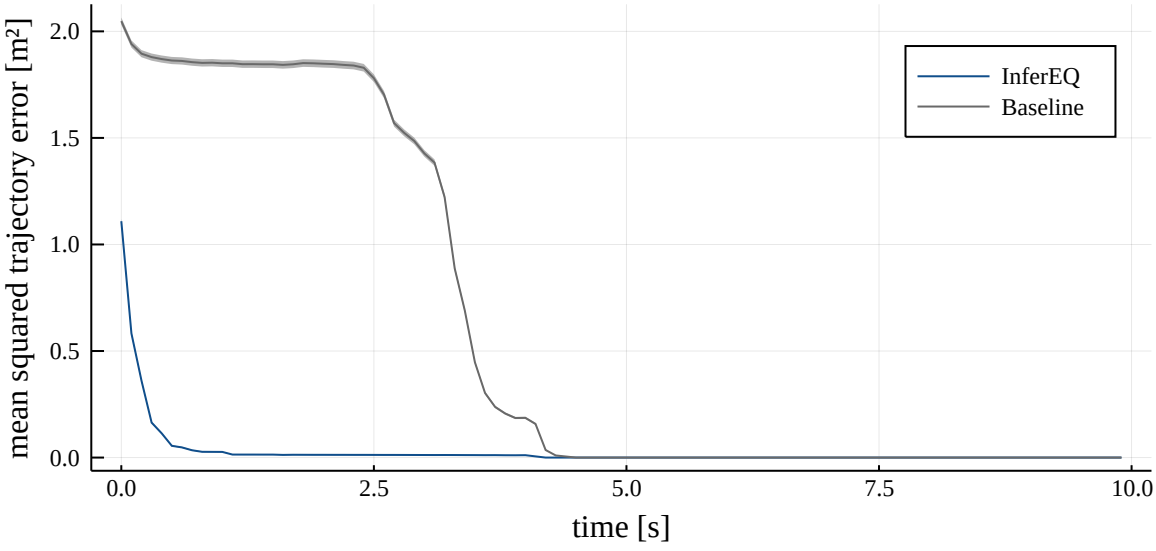


Figure 4.4.3: Mean squared prediction error for the 3-player running example under equilibrium uncertainty. InferEQ: prediction with equilibrium inference. Baseline: prediction when equilibrium uncertainty is ignored. Ribbons indicate the standard error of the mean.

4.4.4 Closed-Loop Interaction

The performance of the full inference and planning framework is tested in a closed-loop interaction scenario. In this experiment, the behavior for all human players is generated by solving the game for a randomly selected equilibrium. All players generate their respective strategies over the remaining game horizon, i.e. replanning after every time step to close the loop.

The MAP aligned planner is compared to a baseline using the planning approach of Fridovich-Keil *et al.* [1]. As for the prediction experiments, the baseline does not actively reason about different solutions but uses a fixed local equilibrium to make control decision. Again, this local equilibrium is initially sampled from the seed distribution, $\mathcal{P}_\gamma$, and updated over the remaining game horizon by re-solving the game after each observation of the physical state x.

3-Player Scenario

Figure 4.4.4 shows the distribution of costs incurred by each player over 200 simulations of the 3-player scenario for both planning approaches. From the cost distribution it is clear that the MAP aligned planner performs significantly better than the non-adaptive baseline. By actively aligning to the equilibrium chosen by the human players, the robot not only reduces its own cost but also the cost the remaining players. For the robot (Player-1), the gap in performance results from the improved ability to compute efficient plans as the robot has a better understanding of how humans will react. Conversely, from the perspective of the human players, the strategy aligned robot matches their prediction more closely and thus renders their own plans more efficient.

Further insight is gained by analyzing the qualitative difference between both methods in closed-loop planning. For this purpose, player trajectories for each approach are analyzed in simulations of the same human behavior. In each simulation, the random seed generating the human behavior is fixed to the same value for both methods. Note that fixing the random seed only fixes the *strategy*, but *not* necessarily the trajectory. That is, humans will take the same action when presented with the same physical state x at time t but their actual decision depends implicitly upon the actions of the robot as these in turn affect the state.

For the baseline, the analysis reveals two qualitatively different cases which are reflected in the bimodality of the cost distributions in Figure 4.4.4 (right). In the first case, the robot using the baseline controller initially samples the same equilibrium as the human and thus is trivially aligned. Hence, the interaction plays out as planned by all players and the costs incurred by each player in closed-loop interaction closely matches the cost as predicted by their plan. These scenarios compose the low-cost mode of the player cost distribution for the baseline. In the second, more critical case, the robot does not initially sample the same equilibrium as the human players and hence invokes a misaligned strategy. An example that demonstrates the effects of misalignment is depicted Figure 4.4.5. Here, the current player positions are highlighted with circular marks. The robot (Player-1, blue) starts on the left. The robot's predictions, as generated by the baseline planner, are shown

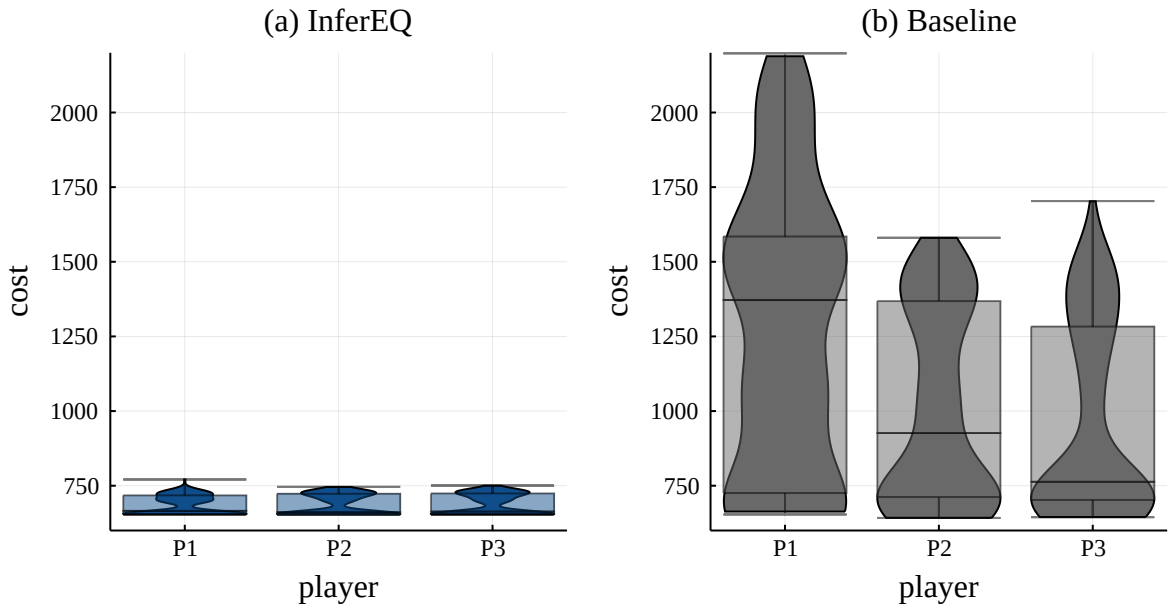


Figure 4.4.4: Distribution of costs incurred by each player in the 3-player running example. (a) InferEQ: Player-1 uses strategy alignment. (b) Baseline: Player-1 ignores equilibrium uncertainty.

as black solid lines while the human plans are shown as red dashed lines. Initially, the robot's prediction matches the plan of the human coming from the top (purple, Player-3) but is misaligned with respect to the strategy of the player coming from the bottom (red, Player-2). Comparing these strategies to the Monte Carlo study in Section 4.4.2, the robot's strategy corresponds to Cluster 7 (c.f. Figure 4.4.2 (g)) while the human behavior is generated from Cluster 1 (c.f. Figure 4.4.2 (a)). Due to this misalignment, the robot predicts that Player-2 will go first and thus approaches the conflict area to let the human pass. Player-2 in turn adapts to the speed of the robot and plans to pass the conflict area at the same speed as the robot in a clockwise circular motion. Consequently, both players slowly inch forward expecting the other to accelerate (Figure 4.4.5 (a-d)). This mutual deadlock continues until $t = 6\,\mathrm{s}$, where the goal cost now dominates the proximity cost and both players accelerate to avoid the penalty for not reaching their respective goal position in time. Due to the close proximity and the large acceleration needed to reach the goal in time, Player-1 and Player-2 incur a high cost of 1169 and 1094 cost units, respectively. Note that in this example Player-3 remains mostly unaffected by the misalignment and only incurs a cost of 670.

For the MAP aligned planner, the variance in performance is significantly lower. Again, there exists a case where the robot is trivially aligned, i.e. if the human chooses the most likely equilibrium from the prior equilibrium distribution induced by $\mathcal{P}_\gamma$. In all other cases the robot has to actively infer the equilibrium from the observed human actions and the incurred cost depends on the time it takes to recover the correct equilibrium. As an example of this inference process, Figure 4.4.6 visualizes the MAP planning procedure for the same random seed (i.e. the same human behavior) as discussed above for the baseline (Figure 4.4.5). Again, the robot starts at the left, the robot's plan (the MAP estimate

of the equilibrium belief) is shown as black solid lines and the human plans are shown as red dashed lines. Additionally, blue lines visualize the equilibrium trajectories of all other particles in the belief where higher opacity corresponds to a higher particle weight. A histogram of the weight distribution over particle indices for the *posterior* belief is provided for each visualized time step. The example shows that initially the robot's MAP estimate corresponds to Cluster 2 (c.f. Figure 4.4.2 (b)) and hence the robot's strategy is not aligned with the human equilibrium (Cluster 1, Figure 4.4.2 (a)). However, already at $t = 2\,\mathrm{s}$ the belief assigns 80 % of the probability mass to the correct equilibrium and the robot invokes the aligned strategy. While at this time step there are still three other equilibria in the belief that have a significant weight, at $t = 4\,\mathrm{s}$ the belief is fully converged. Due to the alignment of the robot to the human equilibrium, no player is forced to deviate from their locally optimal equilibrium strategy and all players reach their goal efficiently, incurring a cost of $[c_1; c_2; c_3] = [648; 638; 645]$, where c_i is the cost of player i.

Remarks on Runtime The 3-player version of the strategy alignment problem is solved at real-time planning rates. The runtime of the strategy alignment approach is dominated by the time it takes to update the game solution for each particle. Solving for a single approximate local Nash equilibrium from scratch takes on average 7.5 ms. However, if the game solver is warm-started with the solution of the previous time step, a single particle update can be computed with an average runtime of just 1.5 ms. Accordingly, an update of all 50 particles in the belief is done in less than 80 ms. Therefore, after having initialized all particles at the first time step, the MAP aligned planner runs in real-time when used with a re-planning scheme that utilizes warm-starting.

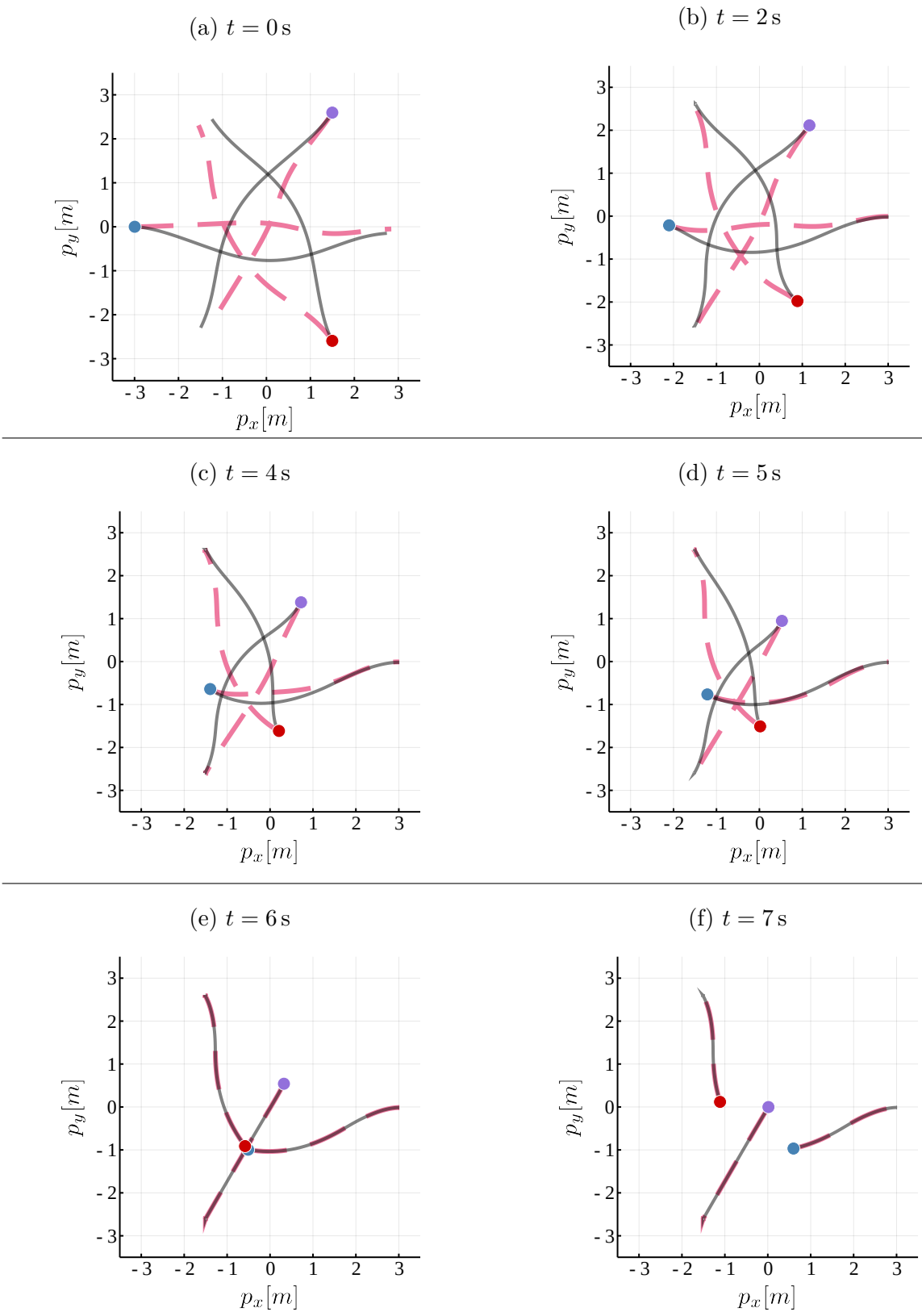


Figure 4.4.5: Example of misalignment in closed-loop planning when using the baseline controller. The robot (blue) starts at the left. Robot plans are shown with black solid lines. Human plans are shown with red dashed lines.

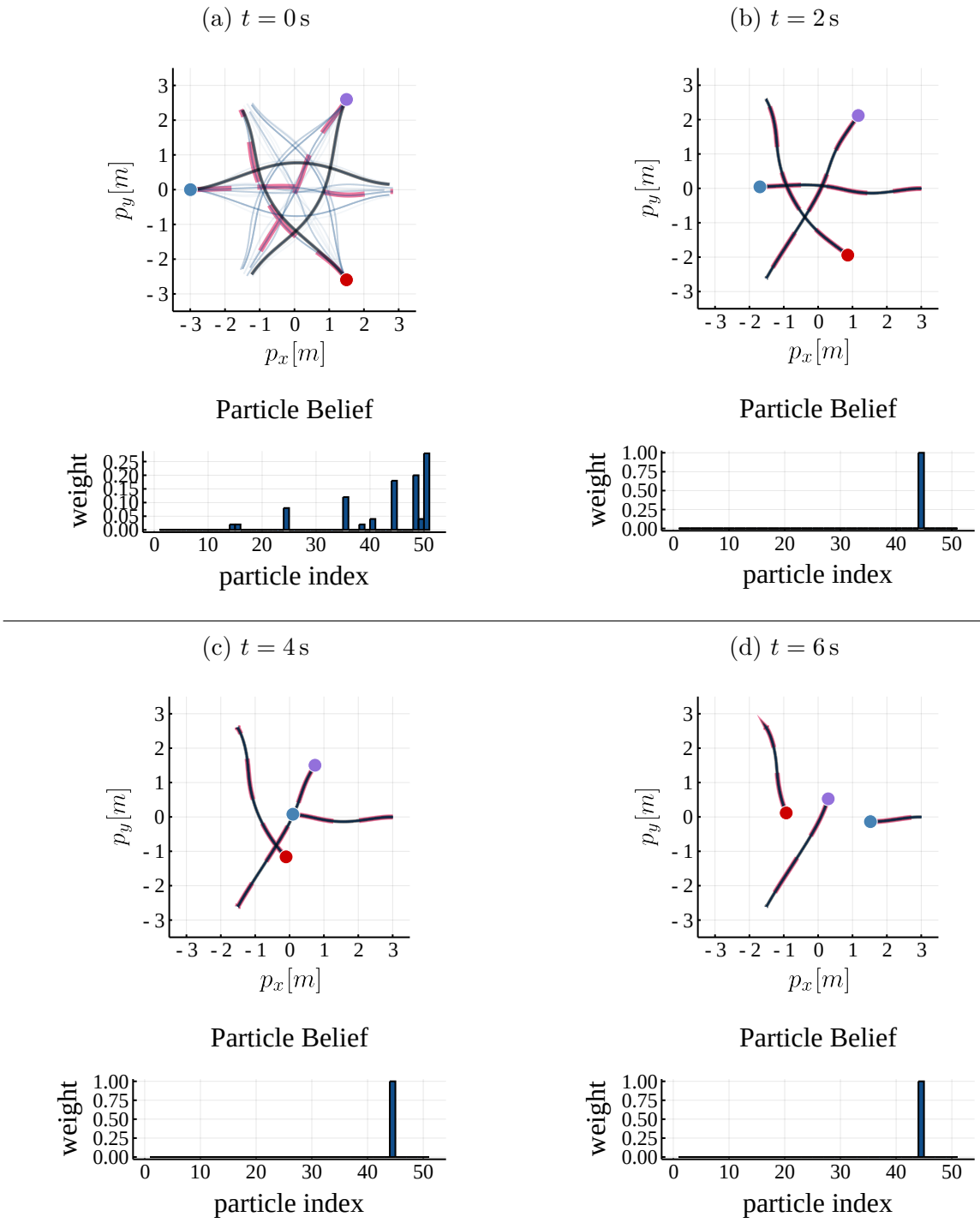


Figure 4.4.6: Example of MAP aligned planning. The robot (blue) starts at the left. Robot plans (MAP estimate of the equilibrium belief) are shown with black solid lines. Human plans are shown with red dashed lines. Blue lines visualize particle trajectories where the opacity is proportional to the particle weight. The particle weight distribution of the posterior belief is shown as a histogram over particle indices.

Scalability: 5-Player Scenario

Finally, scalability of the MAP aligned planner is tested by applying it to a 5-player version of the navigation problem. Examples of equilibrium trajectories for this larger version of the problem are shown Figure 4.4.8.

Figure 4.4.7 shows the distribution of costs incurred by each player over 200 simulations of the 5-player scenario for MAP aligned approach as well as the baseline. Again, one can observe a significant improvement by inferring the local equilibrium and aligning the robot's strategy accordingly.

This improvement is most noticeable for the autonomous agent (Player-1). The reason for this larger gap in performance is the increased complexity of the problem which reduces the chance of choosing an aligned strategy without actively reasoning about the equilibrium chosen by other players. Human players, on the other hand, only incur a high cost if the robot's misalignment causes conflict with their selected strategy. Another indicator for this finding is the fact that those human players initially located close to the robot, i.e. Player-2 and Player-5, exhibit the largest improvement of their cost distribution if the robot uses strategy alignment instead of the baseline approach.

Note, however, that the inference problem becomes harder as the number of players and equilibria in the problem are increased. Hence, the variance of the cost is significantly higher as compared to the 3-player example and the cost distributions exhibit a longer tail. By examining scenarios in the tail end of the distribution it can be observed that these instances typically correspond to cases where there are multiple high-likelihood equilibria for a large part of the game horizon. If in such a case the robot commits to a misaligned equilibrium, it takes a suboptimal path and has to apply larger inputs once the MAP estimate changes and eventually matches the true latent state. Furthermore, there is an increased chance that humans select an equilibrium which is not represented by any particle in the robot's belief. Note, however, that the latter problem can be addressed by increasing the number of particles. Figure A.1.1 in the appendix of this thesis demonstrates that the planning performance is further improved if a larger number of particles is used. In practice, the number of particles is to be chosen based on the available computational resources and the required level of safety and efficiency for the application domain.

Remarks on Runtime The increased number of particles and the higher computational complexity of finding equilibria significantly increase the run-time for this problem. In the implementation of the 5-player navigation problem, solving for a single local equilibrium from scratch takes on average 85 ms. When utilizing warm-starting, this time can be reduced to 9 ms. Accordingly, a planning step takes 1.35 s when updating all 150 particles using a single thread. However, many particles converge to the same equilibrium. Thus, by not only combining the weights but also sharing solutions between particles that have been identified to represent the same equilibrium as per the call to CombineDuplicates, the performance can be significantly improved. Employing this optimization, the average runtime is reduced to 196 ms for this example. Thus, while in the current implementation the 5-player problem can not be solved in real-time, the runtime still remains moderate.

Furthermore, it should be noted that for the experiments conducted here, inference only uses a single thread. However, solutions for individual particles can be computed independently and thus large parts of the inference procedure are trivially parallelizable. As each particle update is computed in well below 100 ms, a multi-threaded particle filter implementation poses a promising approach to achieve real-time planning-rates above 10 Hz. Beyond that, elimination of low-likelihood particles, e.g. via standard resampling or simple thresholding, may help to further reduce the runtime.

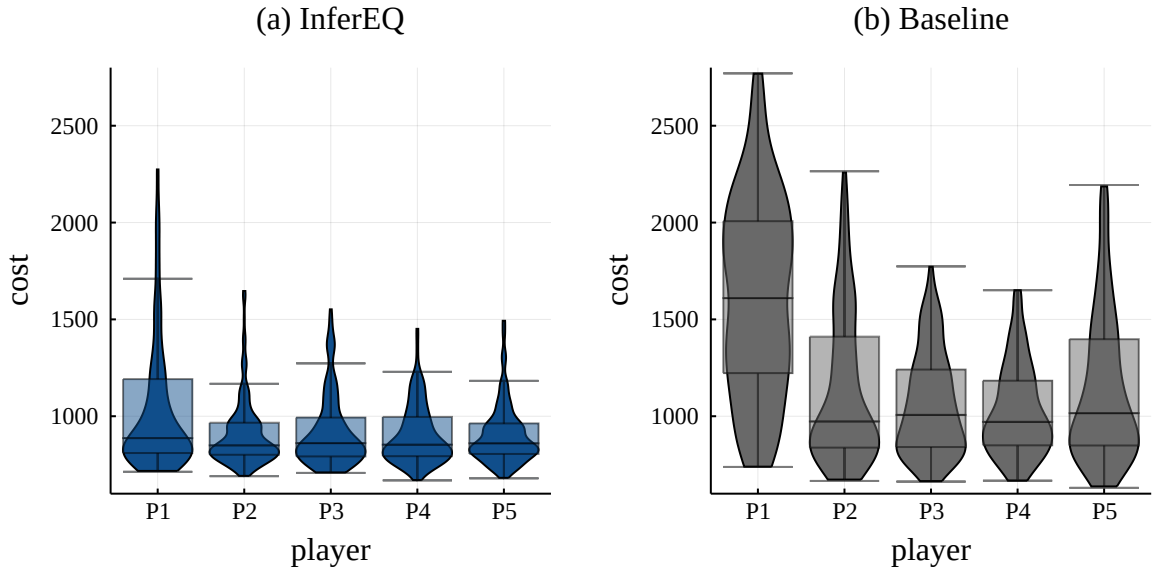


Figure 4.4.7: Distribution of costs incurred by each player in a 5-player navigation problem. (a) InferEQ: Player-1 uses strategy alignment. (b) Baseline: Player-1 ignores equilibrium uncertainty.

4.4.5 Final Discussion and Conclusion

Before this chapter is closed, this section summarizes the main results of the evaluation, discusses limitations and provides possible directions to overcome some of the shortcomings of the approach.

The Monte Carlo study presented in Section 4.4.2 demonstrates the expressiveness of a game-theoretic interaction model and shows that approximate local Nash equilibria have an intuitive interpretation in the studied navigation problem: each of them corresponds to a different plausible mode in which players may seek to avoid collisions with others. The fact that this large variety of local solutions could be recovered from a simple seed distribution additionally highlights the utility of the sampling based approach as a tool for practitioners.

The prediction experiments in Section 4.4.3 demonstrate that the proposed equilibrium inference method significantly improves upon the predictive power of a game solver that only considers a single local equilibrium. With respect to the focus of this work — closed-loop

interaction of robots with humans — accurate prediction is crucial in order to find safe and efficient plans. However, the demonstrated predictive power makes equilibrium inference an interesting approach for other application domains, too. For example, a predictor using equilibrium inference may be used for traffic monitoring to detect misaligned, thus unsafe behavior.

The closed-loop interaction experiments conducted in Section 4.4.4 show that by inferring the equilibrium that determines the human behavior, the robot is able to reduce not only its own cost, but also the cost of all *other* players. A case study of a scenario in which the robot invokes a misaligned strategy provides the following insight: if the robot fails to align with the human equilibrium, the locally optimized strategies of two or more players may be rendered highly inefficient. In the example studied here, this misalignment results in a coordination failure where two players expect their opponent to pass the conflict area first.

Of course, a real-world scenario, human players are likely not bound to a fixed equilibrium and may align to the robot strategy if they detect the coordination failure. Thus, in practice, the performance gap between a strategy aligned planner and non-aligning baseline may be less severe. However, expecting humans to resolve such conflicts would result in overly aggressive behavior of the robot and can result in inefficient or even unsafe behavior in interaction with less attentive humans. Nonetheless, studying equilibrium alignment in experiments with real human behavior remains an important direction of future work.

With respect to real-time performance and scalability of the approach, the conducted experiments show mixed results. For the 3-player example studied in this work, the approach achieves real-time performance. When scaled to five players, the runtime remains moderate, but real-time performance is not achieved. This experiment shows that the number of particles required to cover a larger number of equilibria and the computational complexity of solving high-dimensional games are limitations of the proposed approach. Possible directions to address this problem are to consider only a limited number of relevant players, or to decouple the problem into independent groups of players. Furthermore, the number of samples required to recover all equilibria may be reduced by incorporating prior knowledge in the seed distribution. If the basin of attraction for different equilibria is well understood, in the best case, only a single sample is needed to recover each equilibrium. One possible approach to the design of a more advanced seed distribution is to solve a large number of games offline and perform spatial clustering on the solution (c.f. Section 4.4.2) to select only one seed per equilibrium; i.e. casting a categorical distribution over seeds that are likely to converge to different equilibria. In some special cases, it may also be possible to construct suitable seed strategies systematically, for example, via braid theory [81].

This chapter only considered problems in which the robot has full knowledge of the objectives of other players. This shortcoming is addressed in the next chapter by extending the inference framework to consider a *distribution* of possible objectives for other players.

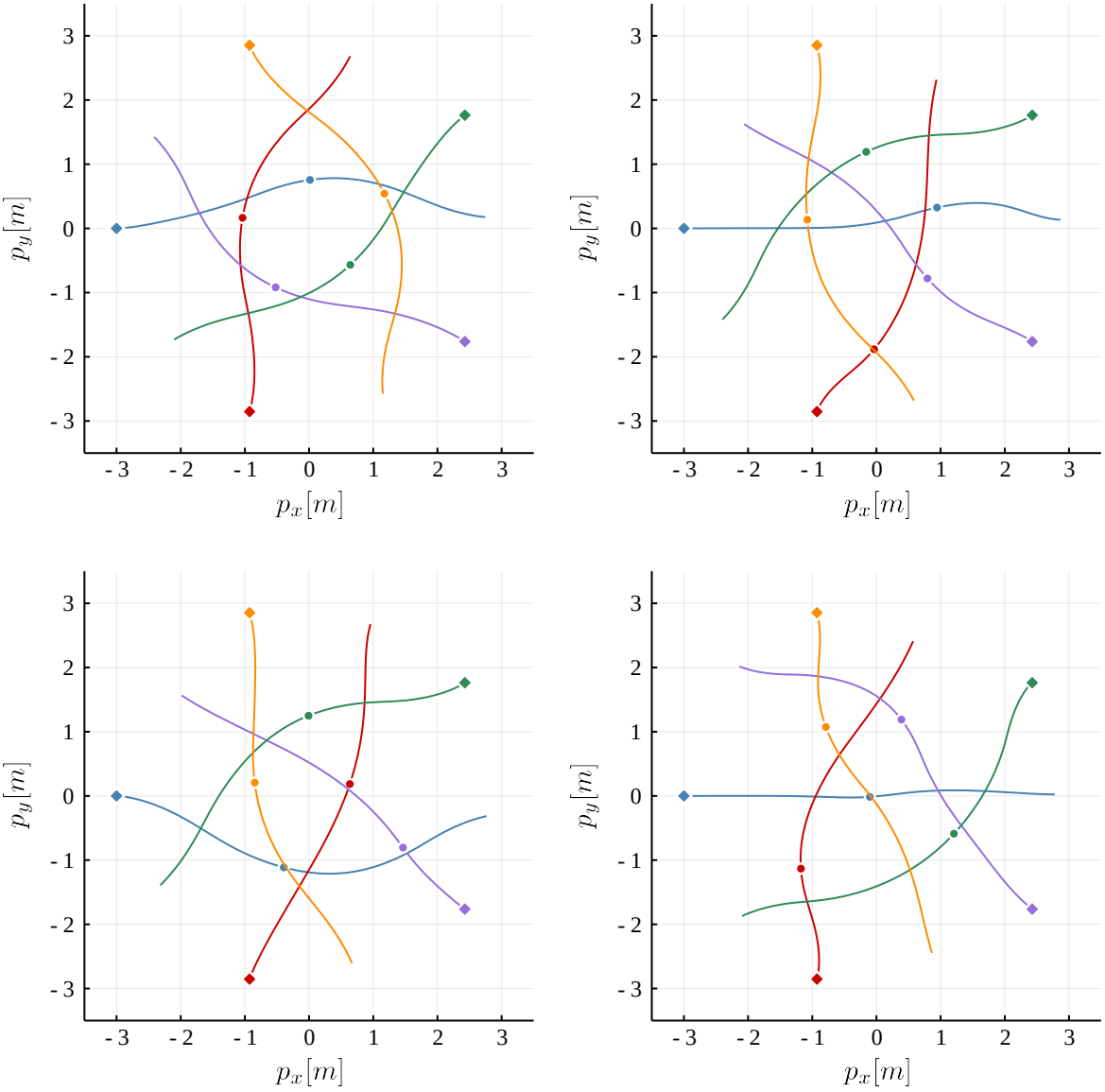


Figure 4.4.8: Four examples of equilibria in a 5-player navigation problem. Initial positions and positions at half the simulation horizon highlighted, respectively, with rectangular and circular markers.

Chapter 5

Planning with Objective Uncertainty

Chapter 4 proposed an inference algorithm for reasoning about the equilibrium solution that other players operate at and introduced a planning scheme for aligning the robot's strategy accordingly. However, this decision model considered only problems with a priori known objectives of all players. Examples of objective uncertainty in the problems discussed in the preceding chapter are given by the fact that the robot may not know the desired goal location of human players or their aggressiveness; i.e. the extend to which each player is willing to make room for others to avoid a collision. As a result, in the face of objective uncertainty, in addition to not knowing *which equilibrium* is playing out, the robot does not know *which game* human players are solving to generate their behavior. This chapter extends the approach presented in Chapter 4 to such scenarios with partially observed objectives and investigates the value of reasoning about objective uncertainty in HRI scenarios.

This chapter is structured as follows. First, Section 5.1 extends the problem formulation and decision model presented in Chapter 4 to scenarios with partially observed objectives. Section 5.2 then discusses how the particle filtering technique for equilibrium inference can extended to make use of this decision model. Since large parts of adapted inference method are closely related to the approach discussed in Chapter 5, these theoretical discussions are kept brief and seek to highlight only the relevant differences. Finally, Section 5.3 evaluates the extended inference approach in simulation and compares it to the method developed in Chapter 4.

5.1 Problem Statement and Decision Model

In this section the problem statement given in Section 4.2 is extended to scenarios with objective uncertainty and the decision model is formulated accordingly.

5.1.1 Game Formulation with Partially Observed Objectives

Again, the interaction of a single robot with $N-1$ humans in an N-player general-sum differential game is considered. Furthermore, the fully observed game formulation, the equilibrium concept, and conventions about player indexing are directly inherited from the formulation in Section 4.2.2.

To incorporate aspects of objective uncertainty, the following augmentation is made. Instead of giving the autonomous agent (Player-1) direct access the cost functions, $J_{2:N}$, of other players, an asymmetric information pattern is chosen in which the robot only knows the *distribution* from which these objectives are drawn. For this purpose, a distribution is defined from which the cost functions of all humans are jointly sampled. This distribution of player costs induces a distribution over games that human players may solve to generate their behavior.

The cost function distribution is defined via a parametric family of cost functions. For each human player, $i \in [N] \setminus \{1\}$, a cost function

$$J_i\left(u_{1:N}(\cdot); \Theta_i\right) \triangleq \int_0^T g_i\left(t, x(t), u_{1:N}(t); \Theta_i\right)\, dt, \forall i \in [N] \setminus \{1\}\,, \tag{5.1.1}$$

is defined, which augments the formulation in Equation (2.1.2) with a vector of $n_{p,i}$ parameters, i.e. $\Theta_i \in \mathbb{R}^{n_{p,i}}$. Using this parametric formulation, the distribution of cost functions is implicitly defined by specifying the distribution of the joint parameter vector, $\Theta = [\Theta_2; \ldots; \Theta_N]$. This parameter distribution is denoted $\mathcal{P}_\Theta$.

Note that this probabilistic description of the game is only from the perspective of the autonomous agent, while the human players still generate their behavior from the equilibrium solution of a fully observed game with deterministic objectives of all players.

Running Example. The running example of a HRI problem originally introduced in Section 4.2.1 is revisited and extended to scenarios with partially observed objectives. For this purpose, the navigation problem is modified such that the robot does not know the *aggressiveness* of other players.

The system dynamics given in Equation (4.2.4) and the structure of the cost given in Equations (4.2.5) to (4.2.8) are directly inherited from the running example used in Chapter 4. To model partial observability of the aggressiveness of each player, the proximity cost weight, $c_{\text{prox},i}$, is modeled as an unknown parameter of the cost function of each human, $i \in [N] \setminus \{1\}$. Two different values for this proximity cost weight are considered: $\bar{c}_{\text{prox}}$ and $\underline{c}_{\text{prox}}$, with $\bar{c}_{\text{prox}} \gg \underline{c}_{\text{prox}}$. A high proximity cost weight for player i, i.e. $c_{\text{prox},i} = \bar{c}_{\text{prox}}$, causes that player to give a wider berth to make room for others. Conversely, a low proximity cost weight, i.e. $c_{\text{prox},i} = \underline{c}_{\text{prox}}$, causes player i to act more aggressively in that this player takes only little effort to make room for others.

Finally, the objective distribution of the game is defined by choosing $\Theta_i = c_{\text{prox},i}$ as the parameterization of cost function J_i and defining a distribution over that parameter. For simplicity, in this example the discrete uniform distribution $\mathcal{U}\{\underline{c}_{\text{prox}}, \bar{c}_{\text{prox}}\}$ is used and parameters for different human players are drawn independently from this distribution.

5.1.2 Objective Inference Model

Using the augmented problem formulation presented above, the strategy alignment problem is extended to incorporate objective inference. Following a similar line of arguments as in Section 4.2.3, it is assumed that there exists a differential game with deterministic objectives whose solution to a local equilibrium determines the behavior of *all* humans. Hence, humans are assumed to be able to fully observe the objectives of other players.

The robot, however, can not directly observe the objectives of other players. Thus, it needs to solve the joint problem of both: inferring the objectives $J_{2:N}$ (i.e. the parameter vector Θ), *and* inferring the human equilibrium selection within the corresponding game.

In general, both inference problems need to be addressed jointly as the equilibria in the game depend upon the objectives. Hence, the robot may maintain a joint belief over both, the latent cost parameter vector Θ, and the latent equilibrium k_t, i.e.

$$b_t(\Theta, k_t) \triangleq p(\Theta, k_t | x_{1:t}, u_{1,1:t}). \tag{5.1.2}$$

Note that in this formulation the cost parameter is assumed stationary and thus the cost parameter is denoted without time index.

The augmented DDN for this problem is given in Figure 5.1.1. In contrast to the formulation in Figure 4.2.2, here, the latent equilibrium k_t at time t additionally depends on the time-invariant parameterization Θ of the human objectives.

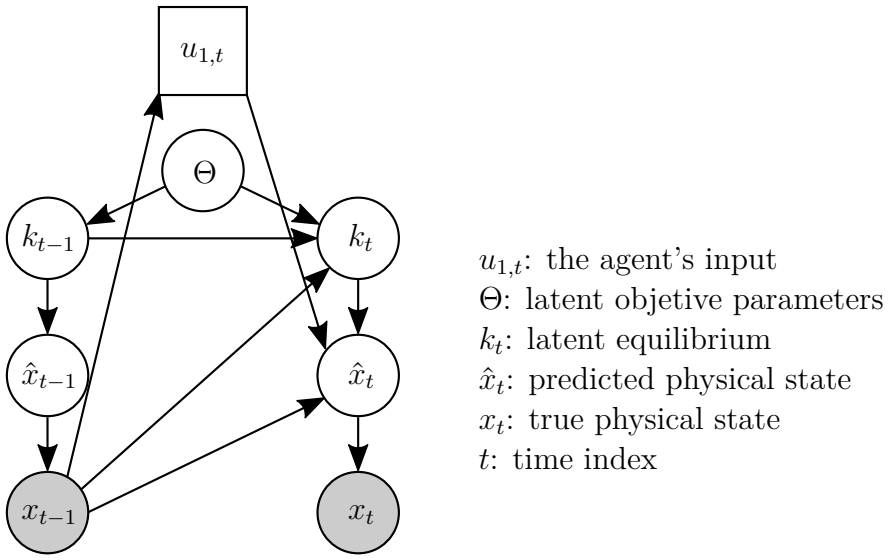


Figure 5.1.1: DDN used to model the joint inference of human objectives and equilibria.

5.2 Strategy Alignment in Games with Partially Observed Objectives

Since any problem with fully observed objectives can be formulated using the augmented decision model with a deterministic distribution of objective parameters, it is obvious that

the inference problems considered here are at least as complex as the problem class studied in Chapter 4. Consequently, performing an exact belief update of Equation (5.1.2) remains impractical for this class of problems. Instead, the particle filtering technique presented in Chapter 4 is augmented to accommodate objective uncertainty. With this adapted belief update, aligned closed-loop planning can then be realized using the same approach as presented in Chapter 4, i.e. by invoking the agents strategy from the MAP equilibrium.

Given the DDN in Figure 5.1.1 the belief update rule for Equation (5.1.2) can be written as

$$b_t(\Theta, k_t) \propto \int_{k_{t-1} \in \mathcal{K}} \int_{\hat{x}_t \in \mathcal{X}} p(x_t|\hat{x}_t) p(k_t, \hat{x}_t|u_{1,t}, k_{t-1}, x_{t-1}, \Theta) b_{t-1}(\Theta, k_{t-1}) \, d\hat{x}_t \, dk_{t-1}. \quad (5.2.1)$$

The approximation of this update rule via a particle filter can be achieved by making two essential changes to the approach presented in Section 4.3.

First, in order to approximate the joint belief over objectives and equilibria, a particle in this problem corresponds to a tuple of a parameter sample, $\Theta^{(k)}$, and a strategy profile, $\gamma^{(k)}$. Hence, the generation of the initial particle belief described in Algorithm 3 needs to be adjusted to draw samples accordingly. The adapted sampling procedure for the initial particle belief is summarized in Algorithm 6. In this work, particles are drawn from the product of the parameter distribution, $\mathcal{P}_\Theta$, and the seed distribution, $\mathcal{P}_\gamma$. This step corresponds to line 2 and 3 of Algorithm 6. Note, however, that for some applications a joint distribution that does not assume statistical independence of the seed and the objective may pose a useful extension that is easily implemented. In contrast to the procedure described in Algorithm 3, here the call to SolveGame additionally takes the objective parameter $\Theta^{(k)}$ when solving the game for particle index k (line 4 of Algorithm 6).

Algorithm 6 Generation of an initial particle belief for objective inference from the product of $\mathcal{P}_\Theta$ and $\mathcal{P}_\gamma$.

1: **procedure** GenerateInitialBelief($\mathcal{P}_\Theta, \mathcal{P}_\gamma$)
2: $\quad \{\Theta^{(k)}\}_{k \in [K]} \leftarrow$ sample K objective parametrizations from $\mathcal{P}_\Theta$
3: $\quad \{\bar{\gamma}_0^{(k)}\}_{k \in [K]} \leftarrow$ sample K strategy profiles from $\mathcal{P}_\gamma$
4: $\quad \bar{\mathcal{B}}_0 \leftarrow \{(\Theta^{(k)}, \gamma_0^{(k)} \equiv \text{SolveGame}(x_0, \bar{\gamma}_0^{(k)}; \Theta^{(k)}), w_0^{(k)} \equiv 1)\}_{k \in [K]}$
5: $\quad \mathcal{B}_0 \leftarrow \text{CombineDuplicates}(\bar{\mathcal{B}}_0)$
6: $\quad$ **return** $\mathcal{B}_0$
7: **end procedure**

Second, the transition step of the particle filter needs to consider the dependence of the transition model, $p(k_t, \hat{x}_t|u_{1,t}, k_{t-1}, x_{t-1}, \Theta)$, on the objective parameter Θ (c.f. Equation (5.2.1)). The resulting algorithm for joint inference of objectives and equilibria is summarized in Algorithm 7. Here, SolveGame again is invoked with parameter, $\Theta^{(k)}$, for particle index k (c.f. line 4 of Algorithm 7). Since the objective parametrization is assumed to be stationary, $\Theta^{(k)}$ is copied without modification to the updated belief (line 7 of Algorithm 7).

Algorithm 7 Particle filter for joint inference of objectives and equilibria.

procedure UPDATEEQOBBELIEF($\mathcal{B}_{t-1} \equiv \{(\Theta^{(k)}, \gamma_{t-1}^{(k)}, w_{t-1}^{(k)})\}_{k\in[K]}, x_{t-1}, x_t, u_{1,t}$)
  $\bar{\mathcal{B}}_t \leftarrow \emptyset$
  for particle index $k = 1 \dots K$ **do**
    $\gamma_t^{(k)} \leftarrow \text{SOLVEGAME}\left(x_{t-1}, \gamma_{t-1}^{(k)}; \Theta^{(k)}\right)$
    $\hat{x}_t^{(k)} \leftarrow x_{t-1} + \int_{t-1}^{t} f\left(\tau, x(\tau), u_{1,t}, \gamma_{-1}^{(k)}(x(\tau))\right) d\tau$
    $w^{(k)} \leftarrow w^{(k)} p(x(t)|\hat{x}^{(k)})$
    $\bar{\mathcal{B}}_t \leftarrow \bar{\mathcal{B}}_t \cup (\Theta^{(k)}, \gamma_t^{(k)}, w_t^{(k)})$
  end for
  $\mathcal{B}_t \leftarrow \text{COMBINEDUPLICATES}(\bar{\mathcal{B}}_t)$
  return $\mathcal{B}_t$
end procedure

5.3 Evaluation

This section examines the value of objective inference for prediction and planning. For this purpose, the objective inference approach presented above is compared to a method that uses *only* equilibrium inference while assuming a fixed objective, and a baseline that assumes a fixed value for both latent variables. The evaluation is performed in simulation of two different scenarios: a problem with unknown aggressiveness of other players, and a problem with unknown goal locations.

This section is structured as follows. Section 5.3.1 describes the simulated scenarios and introduces the compared methods. Section 5.3.2 performs a Monte Carlo study of equilibria for the different cost parameters in the support of the objective distribution. Section 5.3.3 compares the prediction performance of the different approaches in a setting where the robot observes the interaction of multiple agents. Section 5.3.4 investigates the value of objective inference in a closed-loop interaction scenario. Finally, Section 5.3.5 concludes by summarizing the main results and discussing their implications.

5.3.1 Experiment Details and Parameters

Evaluation Problems

Two different problems with partially observed objectives are studied to evaluate the performance of the proposed inference method: first, a problem in which the aggressiveness of other players is unknown, and second, a problem with unknown goal positions of other players.

Uncertain Aggressiveness The scenario with uncertain aggressiveness of other players is modeled as described in the running example discussed in Section 5.1; i.e. by casting a discrete uniform distribution over the proximity cost parameter, $c_{\text{prox},i}$, for each player. The two values of the proximity cost weight used to model aggressive and non-aggressive

behavior are given in Table 5.1. All other game parameters are directly inherited from the experiments in Chapter 4 (c.f. Table 4.1).

Table 5.1: Proximity cost weights used to model levels of aggressiveness.

Parameter	Symbol	Value
aggressive proximity cost weight	$\underline{c}_{\text{prox}}$	10
non-aggressive proximity cost weight	$\bar{c}_{\text{prox}}$	50

Uncertain Goal Positions A problem with uncertain goals of human players is modeled by making the goal position, $x_{g,i}$, for each human player, $i \in [N] \setminus \{1\}$ an unknown parameter of their cost function; i.e. choosing the parametrization $\Theta_i = x_{g,i}$ in J_i. Here, two different goal positions for each human player are considered. The first goal candidate for each player is on the side opposite to their starting position, i.e. the player goes straight over the intersection as in Chapter 4. The second goal candidate is given by a left turn of 60°. That is, Player-2 may decide to turn left towards the initial position of Player-1, and Player-3 may decide to turn left towards the initial position of Player-2. All other game parameters are directly inherited from the experiments in Chapter 4 (c.f. Table 4.1).

Approaches

In order to examine the value of objective inference, two different methods are compared to the objective inference approach presented in Section 5.2:

Baseline: A fixed hypothesis for both, the equilibrium and the objective, is used without actively reasoning about the likelihood of either latent variable.

InferEQ: Equilibrium inference as presented in Section 4.3 is used by assuming a fixed objective.

For the evaluation problem with uncertain goal positions, these approaches randomly select an assumed objective for other players. For the experiments with uncertain aggressiveness of human players, two variants of each approach are considered: an *optimistic* version, denoted with postfix "+", that assumes non-aggressive behavior for human players, and a *pessimistic* approach, denoted with postfix "-", that always assumes aggressive behavior.

Finally, all particle filtering approaches are simulated with $K = 150$ particles in the experiments conducted here. The remaining filter parameters, including the observation noise and the seed distribution parametrization, are inherited from Chapter 4 (c.f. Tables 4.2 and 4.3).

5.3.2 Monte Carlo Study of Local Equilibria

This section analyzes the qualitatively different local equilibria that exist for each evaluation problem. For this purpose, the possible objectives that may be attained by sampling a parametrization from $\mathcal{P}_\Theta$ are enumerated and the resulting game is solved for 300 random samples from the seed distribution $\mathcal{P}_\gamma$. Since each of the human players in the studied evaluation problems has two alternatives to choose from, each problem allows for a total of four different objective assignments. Each objective assignment corresponds to a different *true* game that may play out. Within each of these true games there are multiple equilibria which are identified by performing spatial clustering on the sampled solutions. The clustered local equilibria for both problem are examined hereafter.

Uncertain Aggressiveness

For the problem with uncertain aggressiveness of human players an overview of all possible objective assignment is given in Figure 5.3.1. Here, the equilibrium with the lowest cost incurred by the robot is shown for each objective parametrization.

The first objective parametrization, depicted in Figure 5.3.1 (a), corresponds to a scenario in which all players have the same *high* incentive to avoid each other, i.e. both humans are non-aggressive. This scenario corresponds to the problem discussed in the previous chapter and hence the same equilibria as shown in Figure 4.4.2 are attained (c.f. Figure B.1.1).

In the scenario depicted in Figure 5.3.1 (b), Player-2 (red) has aggressive while Player-3 (purple) has a non-aggressive objective parametrization. Accordingly, it can be observed that Player-2 takes only little effort to avoid others while the remaining players are forced to take a wider berth to avoid a collision. Figure 5.3.1 (c) shows a similar situation, now with opposite cost parametrization of both human player.

Finally, Figure 5.3.1 (d) shows an exemplary equilibrium for the scenario in which both human players have a low incentive to avoid other players. Here, only the autonomous agent (blue) takes a wider berth to avoid a collision.

In the interest of brevity, the full Monte Carlo study for this problem is provided in the appendix of this thesis, Figures B.1.1 to B.1.4. This study shows that the equilibria attained within the different objective parametrizations are structurally similar to the problem discussed in the previous chapter; i.e. an equilibrium can be recovered for each possible order in which players may wish to pass the conflict area. Hence, a total number of 32 equilibria is recovered for this problem, 8 for each possible objective parametrization. As discussed above at an example for each parametrization, within each objective assignment the local equilibrium trajectory varies slightly depending on the aggressiveness of each human player.

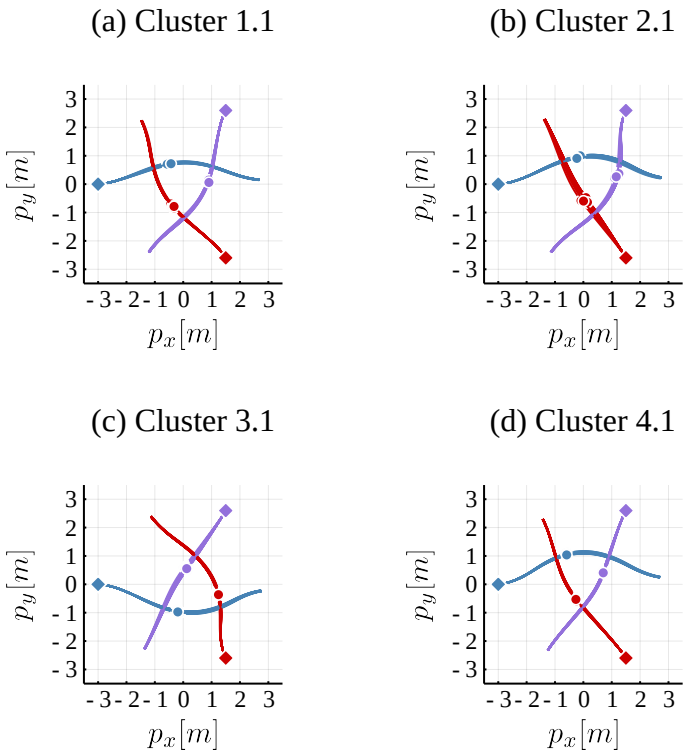


Figure 5.3.1: Equilibrium clusters with the lowest cost incurred by the robot for all possible proximity cost parametrizations of human players. (a) Both humans are non-aggressive. (b) Player-2 (red) is aggressive, Player-3 (purple) is non-aggressive. (c) Player-2 is non-aggressive, Player-3 is aggressive. (d) Both humans are aggressive.

Uncertain Goal Positions

Figure 5.3.2 gives an overview of all possible objectives of humans in the evaluation problem with uncertain goals. Here, each of sub-figure shows a different goal assignment for human players. The first case (a) corresponds to the original running example of Chapter 4 in which all players go straight. In the remaining three cases are scenarios in which (b) Player-1 , or (c) Player-2, or (d) both human players take a left turn.

A full Monte Carlo study for each case is provided in the appendix of this thesis, Figures B.2.1 to B.2.4. In this study, again, a total number of 32 equilibria are recovered, with 8 equilibria for each of the four objective parametrizations. As for the other problem studied in this work, these equilibria correspond to different orders in which players may wish to pass the conflict area.

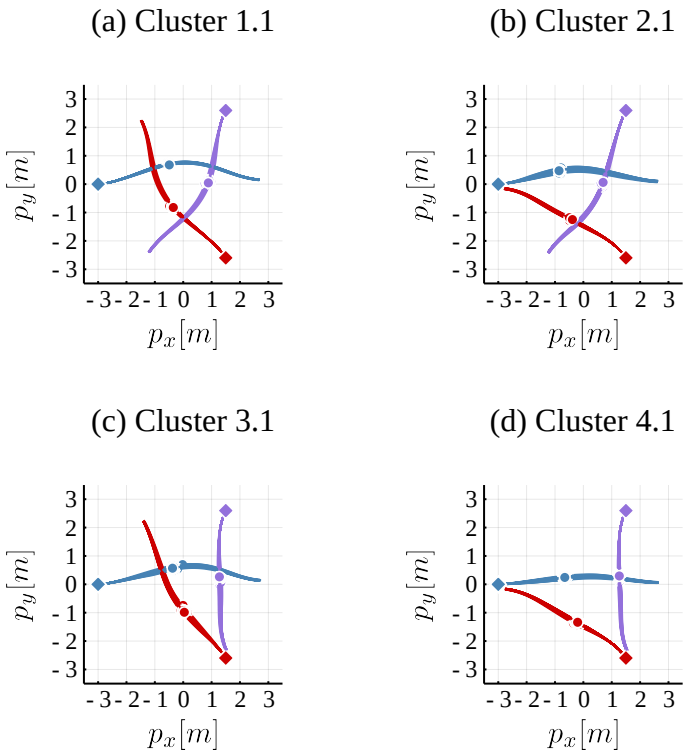


Figure 5.3.2: Equilibrium clusters with the lowest incurred by the robot for all possible combinations of human goal positions. (a) Both human players want to go straight. (b) Player-2 (red) wants to turn left, Player-3 (purple) wants to go straight. (c) Player-2 wants to go straight, Player-3 wants to turn left. (d) Both human players want to turn left.

5.3.3 Prediction Performance

This section examines the utility of the proposed objective inference approach for prediction of observed behavior. The experiments conducted here are of the same form as the prediction experiments presented in Section 4.4.3. That is, the agent *observes* a game playing out at an unknown local equilibrium and is tasked to *predict* the system trajectory for the remaining game horizon. For each evaluation problem the prediction error is then evaluated by examining its statistical evolution over the prediction and the simulation horizon. The presented data is collected from 200 simulations of each problem. For each simulation run an objective parametrization and a strategy seed are sampled to generate the observed game solution.

Uncertain Aggressiveness

Figure 5.3.3a shows the prediction error statistics over the *prediction horizon* for the evaluation problem with uncertain aggressiveness of human players. As for the problem discussed in the previous chapter, the position at the current time is perfectly observed

and the goal positions of all players are known. Hence, the prediction error at the beginning and the end of the prediction horizon are trivially low. In the intermediate range, however, there is a notable gap in performance for the different approaches. The baseline approaches show a poor prediction performance and are clearly dominated by all other methods. Equilibrium inference with fixed objective (optimistic InferEQ+ and pessimistic InferEQ-) exhibit a considerably lower average prediction error. The objective inference approach (InferEQOB) shows the lowest prediction error in this range.

An evaluation of the error statistics for the same data over the *simulation horizon* is shown in Figure 5.3.3b. Here, it is apparent that the baselines are only able to make an accurate prediction after about half the simulation horizon; i.e. when the conflict has been resolved and the problem essentially degenerates to three decoupled optimal control problems. Equilibrium inference, on the other hand, achieves a noticeably reduced prediction error after observing about 1 s of the game. After this initial fast decay of the prediction error for equilibrium inference, the error plateaus and only slowly decays. Finally, the objective inference approach exhibits a quickly decaying prediction error and makes almost perfect predictions after 2 s into the game.

In summary, the performance gap between the baselines and equilibrium inference is large compared to the gap between equilibrium inference and objective inference. These results indicate that, despite potential objective mismatch, the equilibrium inference approaches are still able to identify a qualitatively similar equilibrium that allows moderate prediction errors. The reason for this is the fact that a mismatched objective in this problem only causes a small deformation of the equilibrium trajectories (c.f. Figure 5.3.1). Hence, from the perspective of prediction performance in this problem, equilibrium inference alone already provides a significant improvement.

Uncertain Goal Positions

Figure 5.3.4a shows the prediction error statistics over the *prediction horizon* for the evaluation problem with uncertain human goal positions. For the approach using objective inference, the prediction error grows slower than for all other approaches. Furthermore, equilibrium inference exhibits a lower prediction error than the baseline approach. Since only the position at the current time step is known, all methods exhibit a large prediction error towards the end of the prediction horizon. Note that the *later* regime of the error statistics is composed of predictions made at the *earlier* stage of the simulation. For example, predictions for a 10 s look ahead are only ever made at the first time step of the simulation. At the next time step the remaining game horizon is already 0.1 s shorter and hence predictions only contribute to error statistics up to a 9.9 s look ahead. Consequently, even for the approach using objective inference (InferEQOB), the error in the later regime of the prediction horizon increases since these predictions are based on a smaller number of observations.

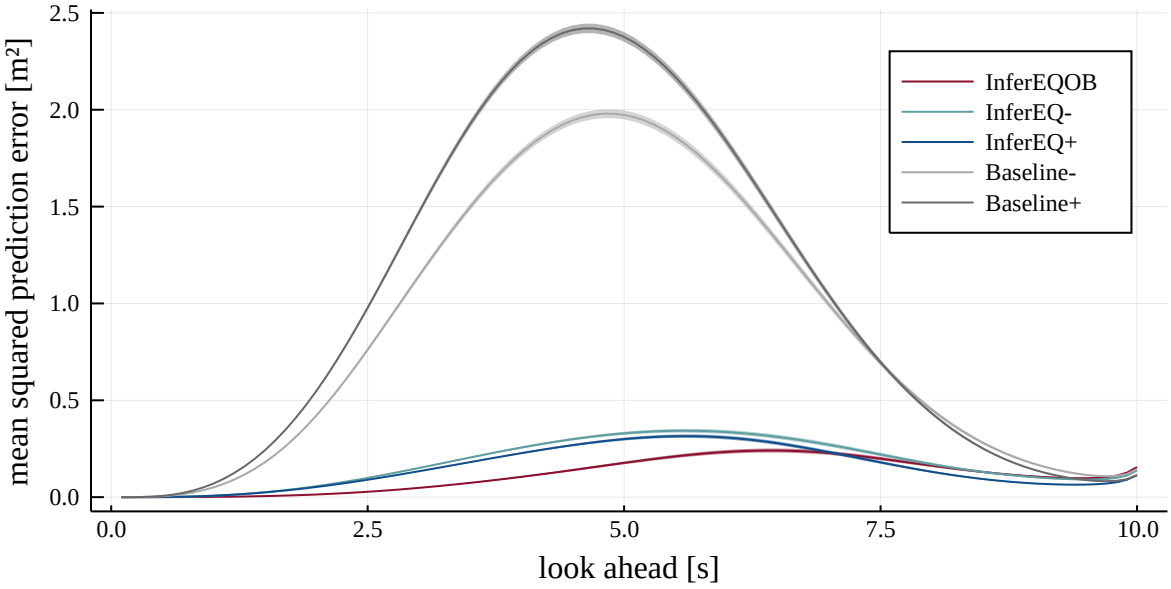
The evolution of error over the *simulation horizon* for the same data is shown in Figure 5.3.4b. Here, the prediction error of the baseline initially plateaus at a high value and

only decays after about half the simulation horizon. Equilibrium inference (InferEQ) is able to reduce the prediction error within the first second of the simulation after which the prediction error plateaus in a lower regime. After half the simulation horizon both the baseline and the equilibrium inference approach achieve the same performance but still maintain a comparably high prediction error which only decreases towards the end of the experiment. Objective inference, on the other hand, exhibits a steadily decreasing prediction error and makes almost perfect predictions after about 2.5 s of the observed game.

The final decay of the prediction error for equilibrium inference and the baseline approach is a direct consequence of the fact that the size of the reachable state space decreases as the remaining time approaches zero. Furthermore, the performance gap between these approaches within the first half of the experiments indicates that equilibrium inference, despite potential objective mismatch, is still able to identify a solution that matches the observed behavior more accurately. The fact that both equilibrium inference and the baseline achieve the same prediction performance in the second half of the simulation supports this finding. That is, as soon as conflicts between the objectives of different players are resolved, equilibrium alignment is trivially achieved and the remaining prediction error is a result of objective mismatch alone.

In summary, objective inference clearly dominates all other approaches. In comparison to the problem with uncertain aggressiveness of human players, for this scenario, inferring the objective parameters in addition to the latent equilibrium provides has a greater value for accurate prediction of behavior.

(a) Error statistics over prediction horizon.



(b) Error statistics over simulation horizon.

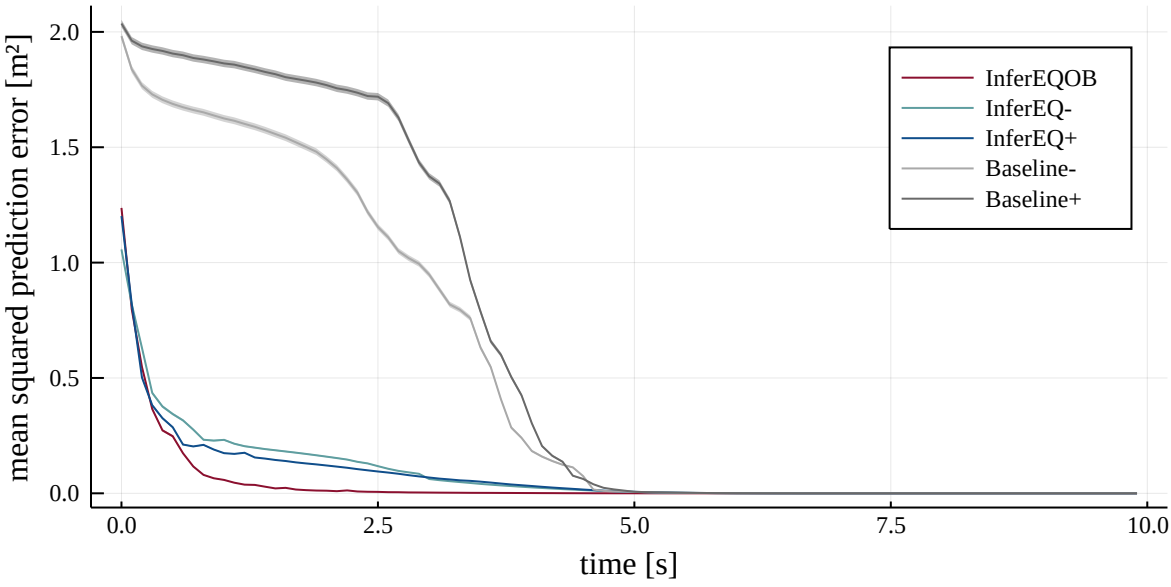
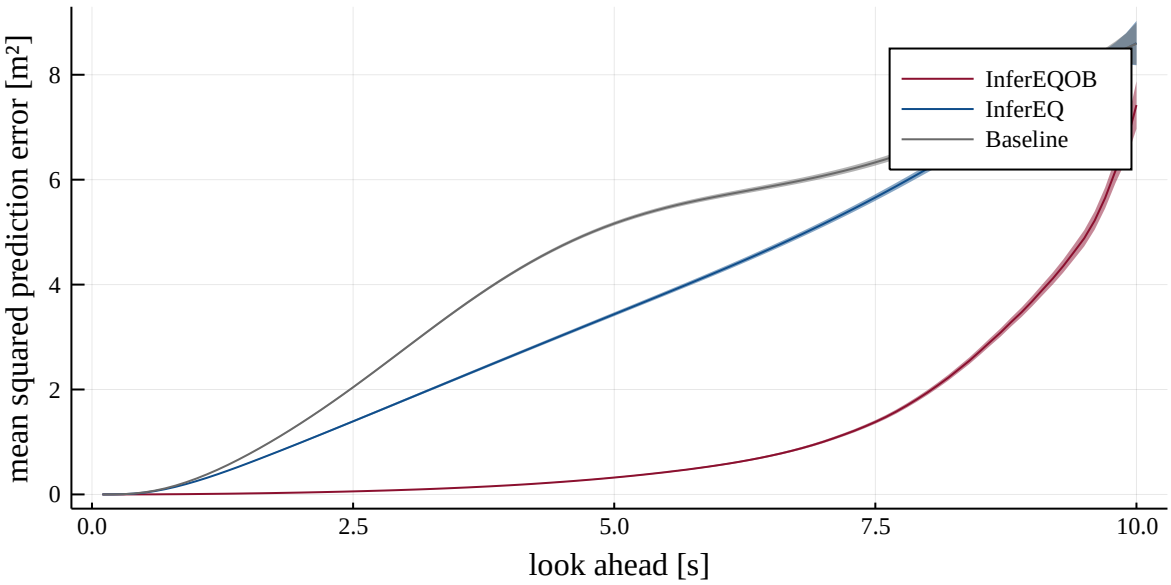


Figure 5.3.3: Mean squared prediction error for uncertain aggressiveness of human players. InferEQOB: inference of objectives and equilibria. InferEQ: inference of equilibria, ignoring objective uncertainty. Baseline: ignoring intention uncertainty. Postfix "+": assuming non-aggressive human behavior (optimistic). Postfix "-": assuming aggressive human behavior. Ribbons indicate the standard error of the mean.

(a) Error statistics over prediction horizon.



(b) Error statistics over simulation horizon.

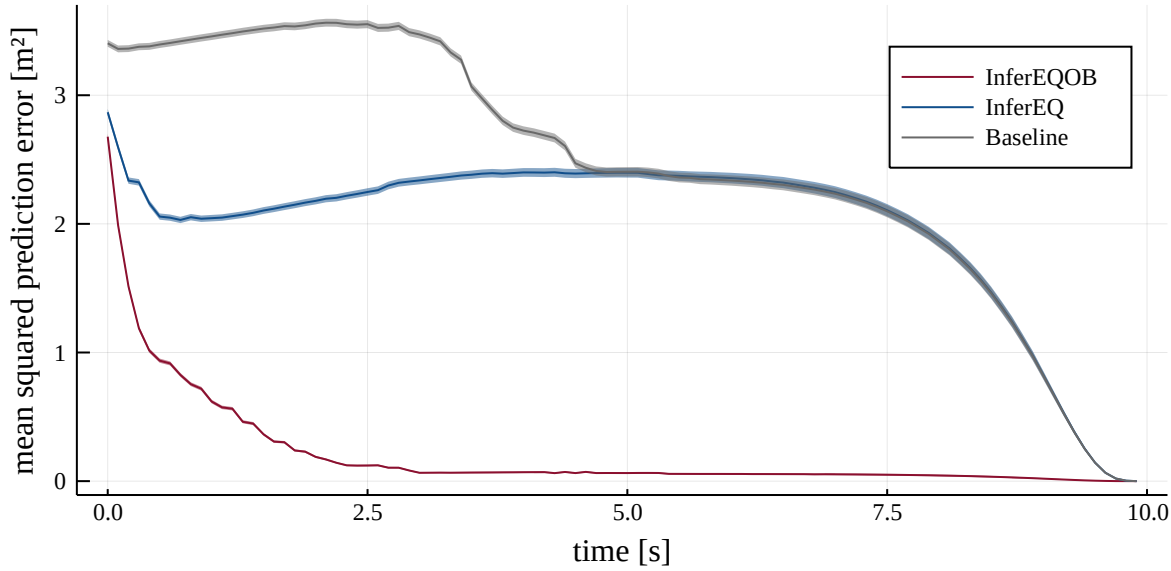


Figure 5.3.4: Mean squared prediction error for uncertain human goal positions. InferEQOB: inference of objectives and equilibria. InferEQ: inference of equilibria, ignoring objective uncertainty. Baseline: ignoring intention uncertainty. Ribbons indicate the standard error of the mean.

5.3.4 Closed-Loop Interaction

This section examines the utility of objective inference in closed-loop interaction scenarios. For each evaluation problem the costs incurred by all player are recorded for 200 simulations. The human behavior in each simulation run is generated by randomly sampling an objective parametrization from $\mathcal{P}_\Theta$ and solving the game at a randomly sampled seed from $\mathcal{P}_\gamma$.

Uncertain Aggressiveness

Figure 5.3.5 shows the distribution of costs incurred by each player for the evaluation problem with uncertain aggressiveness of humans. Additionally, Table 5.2 shows the numerical values of the mean cost incurred by each player.

In this evaluation it is clear that if the autonomous agent uses one of the baseline approaches, the cost for all player is significantly higher than if the robot utilizes one of the three inference approaches. Within the baseline approaches, the pessimistic variant (Baseline-) shows a slightly better performance than the optimistic approach (Baseline+). Within the inference approaches, the performance gaps are smaller and require a closer examination.

First, consider only the cost incurred by the robot (Player-1). Taking the median of the cost distributions for the robot, objective inference (InferEQOB) and optimistic equilibrium inference (InferEQ+) achieve a slightly better performance than pessimistic equilibrium inference (InferEQ-). Taking into account the upper tail end of the agent's cost distribution, i.e. focusing on the worst case outcomes, optimistic equilibrium inference shows the worst performance among the inference approaches. For pessimistic equilibrium inference, the worst case cost is similar to the one for the optimistic variant but the frequency of these outcomes is clearly reduced. Objective inference, in contrast, results in a smaller worst case cost for the robot than all other approaches and the upper tail end of the agent's cost distribution is noticeably more confined. Furthermore, considering the best case costs for the robot, it can be observed that the pessimistic approaches do not achieve the performance of their optimistic counterparts and the objective inference approach. Focusing on the costs incurred by human players (Player-2 and Player-3), it can be observed that pessimistic equilibrium inference admits a marginally lower cost for human players than optimistic equilibrium inference or the objective inference approach.

Finally, considering the mean of the player cost distributions for each approach as shown in Table 5.2, it is clear that objective inference results in the lowest average cost for the robot, while for the human, the average cost is lowest if the robot utilizes pessimistic equilibrium inference.

The presented results admit multiple insightful conclusions. First, it can be concluded that being pessimistic about human behavior in this scenario, i.e. assuming aggressive behavior of other players, helps to reduce the worst case outcomes for the robot. The reason for this is the fact that this planning approach results in more conservative plans. The reduced cost for human players in this case is a direct consequence of this suboptimal play by the robot. If the robot takes a wide berth because it assumes that humans behave aggressively, it admits a more efficient strategy for non-aggressive humans. Finally, pessimistic planning

can not achieve the best case performance of the other inference approaches since the pessimistic strategies are suboptimal if the true human behavior instead is non-aggressive. Conversely, optimistic equilibrium inference is able to exploit more efficient strategies if humans are non-aggressive but can result in more costly maneuvers if the true human behavior corresponds to an aggressive parametrization.

In summary, objective inference achieves the best planning performance among all discussed approaches. By reasoning about both, the objective parametrization and the equilibrium, the worst case and the mean costs are reduced while achieving the same best case performance as the optimistic competitors.

Table 5.2: Mean and standard error of the mean of player costs for uncertain aggressiveness of human players. The lowest cost for each player is marked bold.

	mean cost P1	mean cost P2	mean cost P3
InferEQOB	$\mathbf{840.0 \pm 10.6}$	674.9 ± 5.5	671.1 ± 5.2
InferEQ-	877.3 ± 12.9	$\mathbf{667.0 \pm 4.9}$	$\mathbf{662.1 \pm 4.6}$
InferEQ+	897.7 ± 18.3	684.1 ± 5.1	683.7 ± 4.8
Baseline-	1094.0 ± 20.0	780.0 ± 19.0	779.8 ± 18.5
Baseline+	1214.1 ± 28.2	851.7 ± 23.5	835.8 ± 22.2

Uncertain Goal Positions

Figure 5.3.6 shows the distribution of costs incurred by each player for the evaluation problem with uncertain human goal positions. Additionally, Table 5.3 summarizes the numerical values of the mean cost incurred by each player.

The baseline exhibits a significantly higher cost for all players and is clearly dominated by both inference approaches. Among the inference approaches, the difference in performance is less severe. The only noticeably differences between the corresponding cost distributions is the fact that objective inference exhibits a shorter tail towards high costs for the robot; i.e. the worst case cost and the frequency of these unfavorable outcomes is reduced.

The numerical values of the mean cost incurred by each player in Table 5.3 show that, on average, objective inference exhibits the lowest cost for *all* players. This gap in performance is most noticeable for the autonomous agent (Player-1).

The results presented above show that reasoning about the goal position in addition to the latent equilibrium of human players improves the planning performance in this scenario. However, perhaps surprisingly, the gained utility for closed-loop interaction is small compared to the gained performance for prediction (c.f. Section 5.3.3). A hypothesis which explains this finding is that even for mismatched objectives, i.e. assuming an incorrect goal location of for another player, there still exists an equilibrium that allows an accurate *short-term* prediction of the human behavior.

In order to test this hypothesis, the prediction error for this problem is examined again, but now considering only predictions up to 2 s into the future. Figure 5.3.7 shows the

mean squared error of these short-term predictions. Here, it is clear that the short-term prediction error for equilibrium inference grows only slowly in the first half of the simulation and stays close the performance of objective inference. Since the resolution of conflicts between players occurs in the first half of the simulation, from the perspective of closed-loop planning, accurate prediction of decisions of other players is predominantly relevant in the earlier phase of the problem. Hence, despite the fact that equilibrium inference exhibits a high long-term prediction error if human goal positions are unknown (c.f. Figure 5.3.4b), the accurate short-term prediction in the early phase of the problem still allows the agent to identify an efficient strategy for closed-loop interaction.

In summary, objective inference improves the performance of the autonomous agent in the closed-loop interaction problem with uncertain human goal positions. However, equilibrium inference neglecting goal uncertainty still achieves similar performance in many cases since the short-term prediction error in the relevant regime remains low.

Table 5.3: Mean and standard error of the mean for player costs for uncertain human goal positions. The lowest cost for each player is marked bold.

	mean cost P1	mean cost P2	mean cost P3
InferEQOB	$\mathbf{656.2 \pm 5.0}$	$\mathbf{587.7 \pm 9.0}$	$\mathbf{578.6 \pm 8.3}$
InferEQ	683.3 ± 9.2	589.7 ± 9.9	590.5 ± 10.6
Baseline	1028.2 ± 28.4	743.6 ± 20.8	759.5 ± 25.0

Remarks on Runtime

As discussed in the previous chapter, the runtime of the equilibrium inference approach is dominated by the time it takes to compute a single particle transition and therefore is linear in the number of particles. Hence, for the 3-player problems studied in this chapter, a single game instance is still solved within 7.5 ms, or 1.5 ms when using warm-starting (c.f. Section 4.4.4). However, objective uncertainty increases the number of equilibria in the problem and therefore requires the use of a higher number of particles to cover the joint latent space of equilibria and objectives. Therefore, even if warm-starting is used, an update of all 150 particles takes approximately 230 ms. Thus, real-time planning at 10 Hz is not achieved in this implementation. Still, the runtime remains moderate and the gap to real-time performance may be closed with the measures discussed in the previous chapter (c.f. Sections 4.4.4 and 4.4.5).

5.3.5 Final Discussion and Conclusion

To conclude this chapter on planning with uncertain equilibria of human players, this section summarizes the main results and discusses their implications.

The Monte Carlo study in Section 5.3.2 demonstrates that, in the presence of objective uncertainty, the number of possible equilibria is significantly increased. In the problems

studied here, it is found that even for vastly different objectives, equilibria exhibit a striking structural similarity; for each objective parametrization, the same number of equilibria is recovered and different objectives cause only *continuous deformations* of the resulting equilibrium trajectories. Note, however, that this similarity is not an inherent property of local equilibria per se, but rather results from the structure of the problem and objectives [81].

The prediction experiments conducted in Section 5.3.3 show that the robot is able to predict trajectories more accurately if objective parameters are inferred. However, it is found that the gain in performance depends upon the type of objective uncertainty. While in the case of unknown aggressiveness of human players, the prediction accuracy is only slightly improved by employing objective inference, in the case of uncertain goal positions, it provides a significant benefit. Note that these results are in line with the results of the Monte Carlo study; compared to the goal position of a player, their aggressiveness parameter has only a small impact on the deformation of the equilibrium trajectories.

The closed-loop interaction experiments in Section 5.3.4 show that objective inference enables the robot to identify more efficient plans when faced with incomplete knowledge of others players' objectives. However, it is observed that an equilibrium inference approach that ignores objective uncertainty still performs well in many cases since the *short-term* prediction error in the critical phase of interaction is less effected by objective mismatch in these experiments. This result indicates that, even if the goal location of a human is modeled incorrectly, equilibrium inference still finds the solution computed from this incorrect objective model that best approximates the observed behavior. In the examples studied here, this is enabled by the fact that equilibria for different objectives exhibit the structural similarity discussed above. Due to this structure, even if the robot assumes an incorrect goal location for a human player, equilibrium inference is still able to predict whether this human will wait or whether she wants to pass first since this behavior is also captured by the incorrect objective model. Nonetheless, while objective inference does not drastically improve the *mean* performance of the robot, it improves the *worst-case* performance significantly. Hence, in particular for safety critical applications, the objective inference approach compares favorably with approaches that ignore objective uncertainty. Furthermore, it should be noted that there likely exist scenarios in which understanding the objectives of other players is significantly more critical to safety and efficiency than for the examined navigation problem. In the examples studied here, objectives of different players are only weakly coupled through the proximity cost and other aspects, such as the goal positions of humans, are only indirectly relevant to the payoff for each player. However, other applications domains, such as robot assisted surgery [82] or collaborative manufacturing [83], may naturally exhibit a much stronger coupling of objectives of different players. In such safety critical scenario of close interaction, the objective inference approach can be expected to provide an even greater impact on the performance of the system.

Despite these promising results, in the current formulation, the presented inference approach is only tractable for a small number of possible human objectives. With each additional objective that the robot must consider, the number of possible human strategies — and hence, the number of particles required to cover the latent space — is further increased.

Therefore, applications with many unknown parameters relevant to the safety of the system require further improvement of the proposed method.

One approach that can help to alleviate this problem is to allow latent transitions of the objective parameters. For example, the inference model could allow a small probability for objective parameterizations to transition to a random sample from the parameter distribution, $\mathcal{P}_\Theta$. By this means, even if initially a specific parametrization is not sampled, the robot has a small chance of recovering the objective with every transition of the latent objective parameter. Nonetheless, such modifications can only provide limited alleviation to this problem and scaling to high-dimensional latent parameter spaces remains an open challenge. The next chapter takes a first step towards estimation of high-dimensional objective models by pursuing a different approach: numerical inversion of ILQG via differentiable programming.

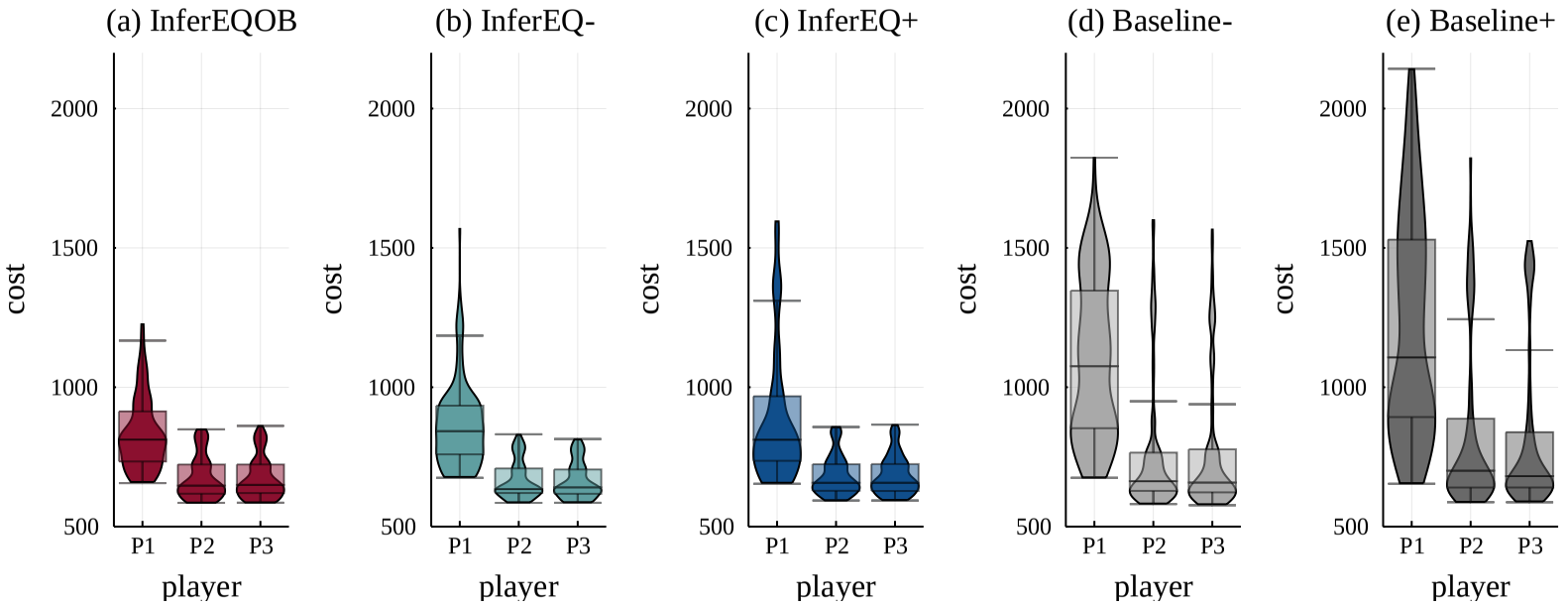


Figure 5.3.5: Distribution of costs incurred by each player for uncertain aggressiveness of human players. InferEQOB: inference of objectives and equilibria. InferEQ: inference of equilibria, ignoring objective uncertainty. Baseline: ignoring intention uncertainty. Postfix "+": assuming non-aggressive human behavior (optimistic). Postfix "-": assuming aggressive human behavior (pessimistic).

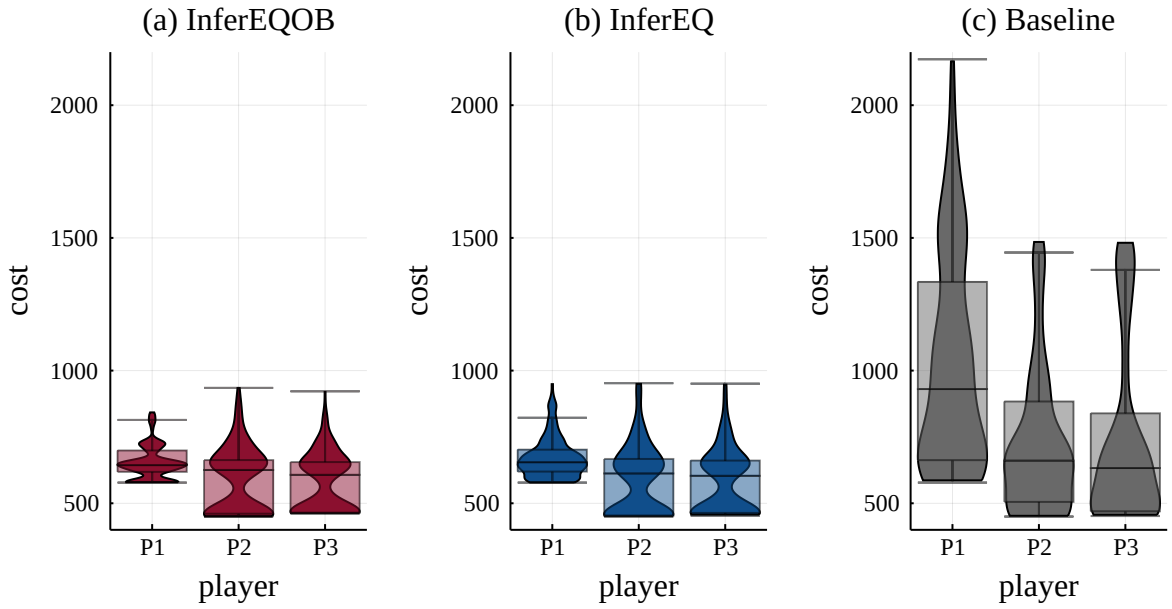


Figure 5.3.6: (a) InferEQOB: inference of objectives and equilibria. (b) InferEQ: inference of equilibria, ignoring objective uncertainty. Baseline: ignoring intention uncertainty.

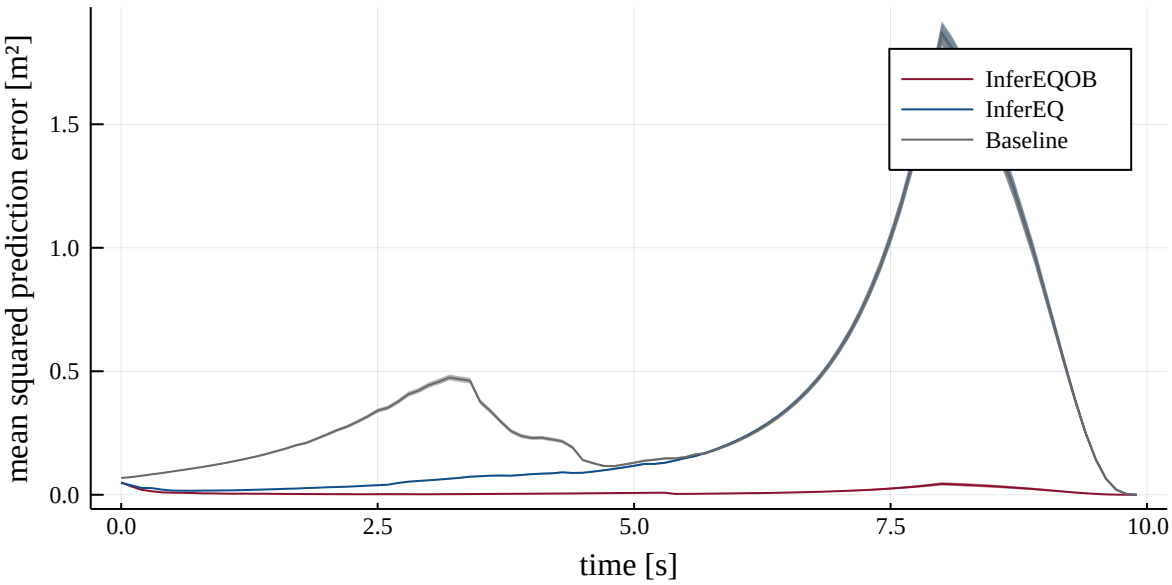


Figure 5.3.7: Mean squared prediction error over simulation time for short-term predictions in the evaluation problem with uncertain human goal positions. InferEQOB: inference of objectives and equilibria. InferEQ: inference of equilibria, ignoring objective uncertainty. Baseline: ignoring intention uncertainty. At every time step, predictions up to 2 s into the future are considered. Ribbons indicate the standard error of the mean.

Chapter 6

Towards an Inverse Iterative Linear-Quadratic Game Solver

The preceding chapters have focused on online methods for handling intention uncertainty in HRI problems, using the framework of differential games. An important requirement for application of these approaches in practice is the ability to express a given interaction problem as a differential game. That is, one needs to identify the differential equation which characterizes the dynamics of the system and find the cost function that describes each player's behavior. While in many cases this may be done manually, doing so for problems with complex objectives or poorly understood dynamics can be tedious or even impractical. Instead, it is desirable to learn these aspects of a game from data.

A significant amount of work has focused on recovering system dynamics from state and input observations, a problem commonly referred to as *system identification* [84]. In the context of single-player optimal control, identification of objectives from observed behavior has also been addressed with great success by recent work [6, 85]. In a single-player setting, this problem is commonly referred to as *inverse optimal control* or *inverse reinforcement learning*. Similarly, in a multi-player setting, the problem of recovering objectives from a given game solution poses an *inverse differential game*. Literature on this class of problems, however, is rather scarce since the topic has only recently received attention [86, 87].

This chapter proposes a solution approach to the inverse differential game problem based on numerical inversion of the iterative linear-quadratic method presented in [1]. In contrast to the problems discussed in Chapter 5, this chapter is concerned with *ex post* recovery of player objectives from a previously recorded dataset; i.e. it is concerned with offline, rather than online estimation. Since, naturally, datasets of human behavior compose of imperfect state information [10, 88], the proposed approach is designed to handle both noise corrupted and incomplete state observations.

Note that this chapter is meant to take a first step towards future work and only validates the feasibility of the proposed inversion approach. Hence, the discussion in this chapter is limited to a first proof of concept. A thorough evaluation of the approach on large-scale datasets of human behavior such as [10] is beyond the scope of this thesis and is left for future work.

This chapter is structured as follows. Section 6.1 formally introduces the inverse differential game problem. Section 6.2 proposes a solution approach for this problem which numerically inverts the ILQG algorithm. Finally, Section 6.3 validates the feasibility of the proposed approach based on a minimal example and discusses possible future directions.

6.1 Problem Statement and Approach

The inverse differential game problem studied in this chapter considers a dataset,

$$Z = \{y(t) \mid t = 0, \dots, T\}, \tag{6.1.1}$$

of state observations, y, that correspond to a sampled state trajectory traced out by a system with known dynamics,

$$\dot{x} = f(t, x, u_{1:N}), \tag{6.1.2}$$

which is controlled by N players, where the ith player is in control of inputs u_i.

Let the observations in the dataset Z be related to the state vector x via a known observation model, h, with

$$y(t) \triangleq h(x(t)) + v(t), \tag{6.1.3}$$

where $v(t)$ is a zero-mean white noise process.

Furthermore, let

$$J_i\left(u_{1:N}(\cdot); \Theta_i\right) \triangleq \int_0^T g_i\left(t, x(t), u_{1:N}(t); \Theta_i\right) dt, \forall i \in [N], \tag{6.1.4}$$

denote a parametric family of cost functions for each player, with parameters $\Theta = [\Theta_1; \dots; \Theta_N]$, where $\Theta \in \mathbb{R}^r$.

Then the inverse differential game problem is to find the parameter vector $\hat{\Theta}$ that best fits the data of observed behavior, Z, presuming that the N players compete in an N-player general-sum differential game in which their behavior is characterizes by an approximate local Nash equilibrium. Here, "best fit" is determined by the loss function,

$$L(\Theta, Z) \triangleq \sum_{t=0}^{T} ||(y(t) - \hat{y}(t; \Theta))||^2, \tag{6.1.5}$$

which rates the utility of parameters Θ based on the sum of squared Euclidean distances between the observations in the dataset, $y(t)$, and the expected observations, $\hat{y}(t; \Theta) \triangleq h(x(t; \Theta))$, which would be received if the game was solved with player objectives corresponding to Θ.

Given this loss function, the objective parametrization that best explains the observed behavior is then

$$\hat{\Theta} = \underset{\Theta \in \mathcal{D}_{\mathcal{J}}}{\arg\min}\, L(\Theta, Z), \tag{6.1.6}$$

where $\mathcal{D}_{\mathcal{J}} \subseteq \mathbb{R}^r$ is the subspace of the parameter space for which a solution to the game exists.

6.2 Numerical Inversion of Iterative Linear-Quadratic Games

There are several approaches that may be taken to solve the optimization problem defined in Section 6.1. For example, one may take a derivative-free approach for numerical optimization of the loss in Equation (6.1.6). Possible methods that may be implemented as a direct extension of the Monte Carlo method presented in Chapters 4 and 5 are the cross-entropy method [89] or particle swarm optimization [90]. While the feasibility of derivative-free approaches has been demonstrated in the related field of inverse optimal control [91], such approaches have been reported to converge poorly compared to gradient-based approaches for smooth parametrizations of the player cost models, J_i [87].

Instead, here, a gradient-based approach is taken to find a (local) minimizer of the loss function, L. This approach poses the challenge of computing the partial derivative of the loss with respect to the player cost parameters. However, if differentiation of the loss is realized, it enables the use of flexible model structures with many parameters, such as artificial neural networks, for which first-order gradient-based parameter optimization has been demonstrated with great success by a wide range of recent work [92].

Differentiation of the Loss Function

The computation of the gradient of the loss function, L, requires expansion of the chain rule until the parameter Θ appears explicitly. For some components of L this might be easily done manually. For example, the differentiation of the observation model, h, with respect to the states, x, may be easily obtained. Further down the chain, however, this process involves a rather unintuitive step: differentiation of the game solution, γ, with respect to the cost parameters Θ. In this work, the game solution is an approximate local Nash equilibrium, i.e. a fixed point of the ILQG algorithm. Accordingly, differentiation of the game solution involves computation of the partial derivative of the procedure outlined in Algorithm 1 with respect to the parameter vector Θ.

From the above discussion it is clear that the partial derivatives of the loss function are not trivially computed manually. However, they can be obtained via automatic differentiation [93]. For this purpose, the loss function, the observation model and the entire game solution pipeline must be implemented as a differentiable problem. In this work, this is achieved in a Julia implementation of these components using the iLQGames.jl framework presented in Chapter 3. In this implementation, automatic differentiation is realized using ForwardDiff.jl which utilizes dual numbers to compute the derivatives in a forward pass of the algorithm [65].

Optimization

Given the differentiable implementation of Algorithm 1, the minimizer, $\hat{\Theta}$, of the loss function, L, can be approximated using a numerical first-order optimization method of choice. In this work, gradient-descent is used to solve this problem where the step-size is chosen adaptively using backtracking line search. However, various other methods are applicable, such as Gauss-Newton methods or more advanced gradient methods including momentum terms [94–96].

6.3 Validation

In order to validate the proposed approach, it is tested on synthetically generated observations of interaction scenarios. This data is generated from games with known values of the parameters, Θ. Section 6.3.1 describes the details of how these validation datasets are generated. Using these synthetic datasets for which the respective true models are known, the inverse ILQG approach is tested with respect to its ability to recover the true model parameters. The results of these experiments are given in Section 6.3.2. Finally, Section 6.3.3 concludes with a brief discussion and outlines potential next steps.

6.3.1 Validation Scenario

Validation is performed on 2-player versions of the evaluation problem with uncertain aggressiveness of human players in Chapter 5. Accordingly, the dynamics are given by Equation (4.2.4), the cost structure follows Equation (4.2.5), and the proximity cost weight $c_{\text{prox},2}$ is the unknown parameter in the parameterization of J_2. However, in contrast to the evaluation problem considered in Chapter 5, here, the unknown parameter is not constrained to a discrete set of values (i.e. aggressive vs. non-aggressive), but rather is sampled from a continuous uniform distribution, $\mathcal{P}_\Theta = \mathcal{U}[\underline{c}_{\text{prox}}, \bar{c}_{\text{prox}}]$. Here, the lower and upper bound on the proximity cost parameter, $\underline{c}_{\text{prox}}$ and $\bar{c}_{\text{prox}}$, are chosen as 1 and 100, respectively. All other game parameters are identical to the experiments conducted in Chapter 4 (c.f. Table 4.1).

A total of 200 datasets, Z_m, are generated by performing the following steps for each of them:

1. Sample a true model parameterization, Θ^*, from $\mathcal{P}_\Theta$.
2. Solve the game with objectives given by Θ^* using ILQG and extract the resulting state trajectory, $x(t)$.
3. Map the state trajectory to an observation sequence, $y(t)$, by propagating each state, $x(t)$, through the observation model, h, and adding noise samples, $w(t)$.

For the experiments conducted here, the observation model h is the function that extracts only the player positions from x; i.e. $h(x) = [p_{x,1}; p_{y,1}; p_{x,2}; p_{y,2}]$. The observation noise,

w, is sampled from a multivariate Gaussian with diagonal covariance matrix, $\Sigma_w = \sigma_w^2 \mathbf{I}$, where the standard deviation is chosen as, $\sigma_w = 0.1\,\text{m}$.

To give an intuition for the data, Figure 6.3.1 shows an example of a dataset generated via the procedure outlined above. In this example, the solid lines correspond to the true game solution and the scattered points indicate the observed positions, y, which compose a dataset Z.

For the inverse ILQG method, gradient descent is performed with an initial parameter guess that is sampled from $\mathcal{P}_\Theta$. Finally, note that the inverse ILQG approach uses the same initial strategy seed as the true model to consider only effects of unknown objectives and eliminate effects of mismatched equilibria for this validation.

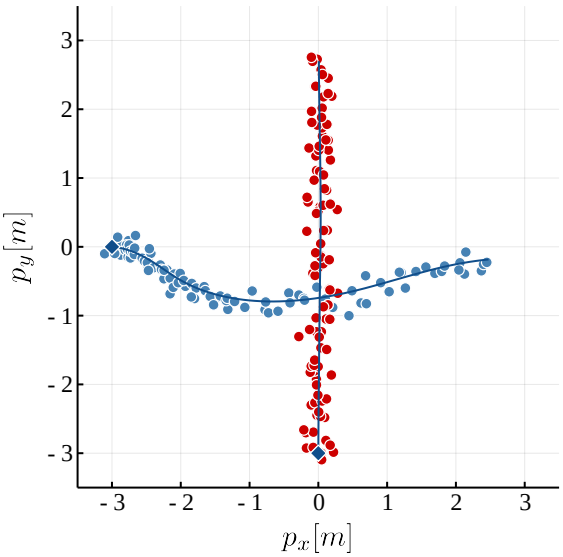


Figure 6.3.1: Example of a synthetically generated dataset. Solid lines indicate the true game solution. Scattered points correspond to the observations contained in the dataset.

6.3.2 Results

The inverse ILQG approach is validated on 200 synthetic datasets as discussed in Section 6.3.1. For each experiment, the evolution of the loss, L, and the parameter error, $\Delta\Theta = ||\hat{\Theta} - \Theta^*||$, are recorded for each iteration of gradient descent.

Figure 6.3.2 shows the evolution of the distribution of the parameter error and the loss for the 200 simulations. Here, ribbons indicate the first and the third quartile of the distributions. Furthermore, Figure 6.3.3 shows the convergence statistics for these experiments. This evaluation shows that the inverse ILQG approach converges reliably for the experiments conducted here. In all 200 simulations, the solver converged within 20 iterations (see Figure 6.3.3 (a)) to a parameter estimate that results in a low loss (see Figure 6.3.3 (b)), while simultaneously accurately recovering the parametrization of the true model (see Figure 6.3.3 (c)). To given an intuition for the quality of fit achieved by

the approach, Figure C.1.1 in the appendix of this figure shows the descent experiment with the highest final parameter error out of all 200 simulations. Note that even in this case, the recovered trajectory matches the observations well and deviates only by a small margin from the true game solution.

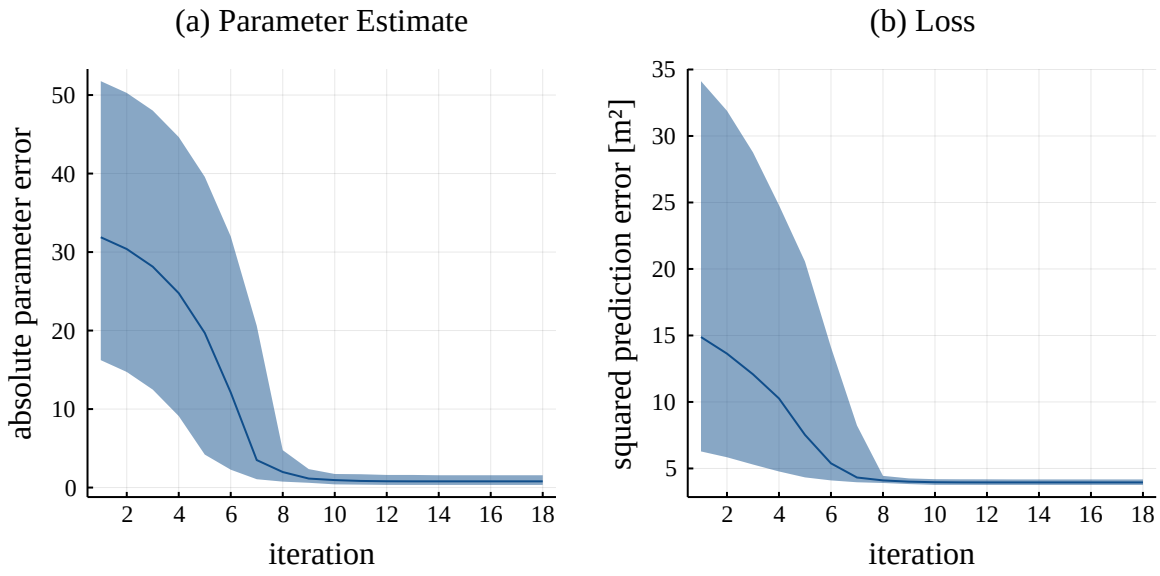


Figure 6.3.2: Distribution of the parameter error and the loss over iterations of gradient descent in inverse ILQG. Solid lines indicate the median of the distribution. Ribbons indicate the first and the third quartile.

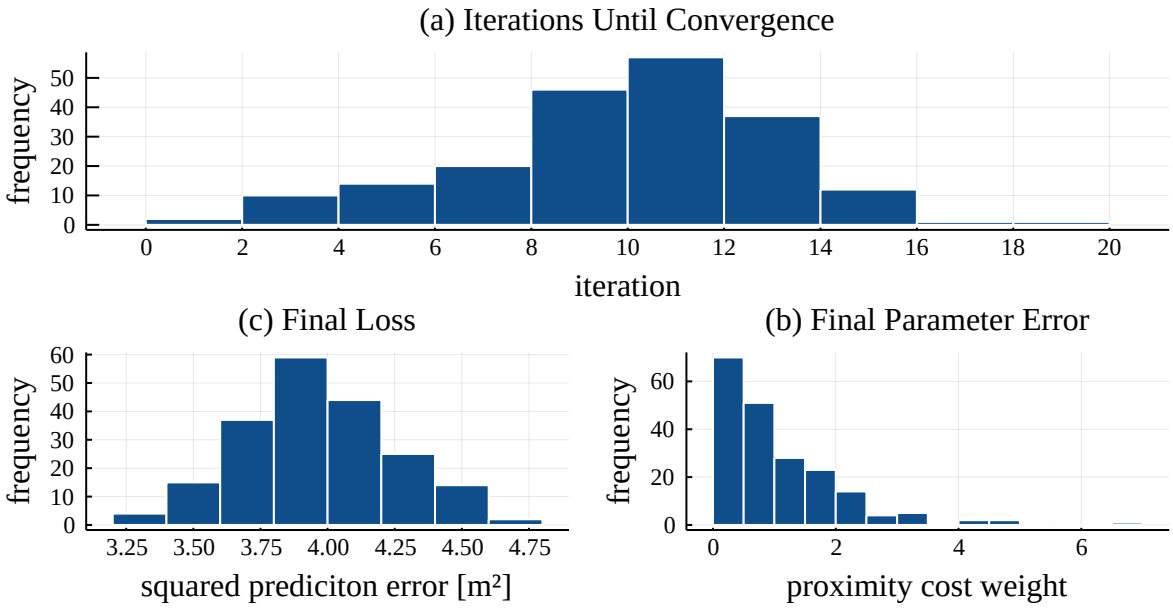


Figure 6.3.3: Convergence statistics for inverse ILQG.

6.3.3 Discussion and Conclusion

The presented results demonstrate that the proposed approach for numerical inversion of the ILQG algorithm is able to accurately recover unknown objectives from datasets of noise-corrupted partial state observations. The reliable convergence within a small number of iterations makes this approach particularly interesting for further investigation. However, this chapter provides only a first step towards future work and there are several interesting directions for further investigation.

Most importantly, the framework should be tested in scenarios with a larger number of unknown parameters and with more flexible model structures for the player cost. In particular, the use of artificial neural networks for the representation of player objectives poses an interesting direction. Similar approaches have shown promising results in the related field of deep inverse reinforcement learning [85]. Furthermore, the optimization problem may be extended to include not only parameters of the objective function but also other aspects of the game. For example, the approach may be extended to estimate the initial strategy profile to address the issue of equilibrium uncertainty.

Beyond that, the approach should be tested on datasets of real-world interactions, such as [10]. This step is particularly important to assess the capability of game-theoretic models to capture real human behavior.

Finally, the presented approach may be combined with some of the ideas of the previous chapters to create an online inference approach that scales well to high-dimensional latent objectives. A discussion of this more overarching topic of future work is deferred to Chapter 7.

Chapter 7

Summary and Future Work

To interact safely and efficiently with environments that are shared with humans, a robot needs to consider the effects of its actions on the decisions of others. These coupled interactions of multiple players are well-described by a general-sum differential game, in which players have differing objectives, the state evolves in continuous time, and optimal play may be characterized by one of many equilibrium concepts; e.g., a Nash equilibrium. Recent work has put forth efficient approximations to differential games that admit solution at real-time planning rates [1]. Despite these advances, an application of these game-theoretic models to human-robot interaction (HRI) is impeded by the fact that, in the current formulation, they do not admit to model an important aspect of such problems, namely, intention uncertainty. This type of uncertainty arises if there are multiple equilibrium strategies that humans may adapt to achieve their objective, or if the robot has incomplete knowledge of the human objectives. This work develops an approach for accommodating intention uncertainty in game-theoretic formulations of HRI problems and investigates the utility of using this approach in comparison to an existing game-theoretic planning model that ignores these sources of uncertainty. In addition to this main contribution, this work proposes an approach for fitting objective models to datasets of noise-corrupted partial state observations of multi-player interactions; i.e. the inverse differential game problem.

The remainder of this chapter reviews the specific contributions and results of this thesis and outlines potential future research directions.

7.1 Contributions and Summary

The first contribution (Chapter 3) is an open-source software framework, iLQGames.jl, that has been developed as part of this thesis to aid the design and solution of general-sum differential games. This framework is build around the ILQG algorithm presented in [1] and is written in the Julia programming language to enable both flexibility and performance. Benchmark results provided in this chapter demonstrate that iLQGames.jl can keep up with execution times of a comparable C++ implementation.

The second contribution (Chapter 4) is an investigation into game-theoretic planning under equilibrium uncertainty in HRI. This chapter is concerned with scenarios in which a robot has full knowledge of the objectives of other players but there exist multiple equilibria that may characterize the human behavior. To address this problem, a particle filtering technique is presented that estimates the likelihood of multiple equilibria from observations of the state. Based on this, a planning model is proposed that uses these likelihood estimates to align the robot's strategy to the most likely equilibrium. The approach is evaluated in simulations of a 3-player game between a robot and two humans that seek to navigate an intersection. A comparison of the equilibrium inference method with a game-theoretic baseline that ignores equilibrium uncertainty admits two main insights. First, by inferring the equilibrium that characterizes the human behavior, the robot is able to predict their trajectories more accurately. And second, by aligning its strategy to the inferred equilibrium, the robot is able to reduce the cost of all players. A case study of a scenario in which the robot invokes a misaligned strategy shows that misalignment can cause a coordination failure in which the strategies of two or more players are rendered highly inefficient. Finally, timing results show that the proposed equilibrium inference planner can be run at real-time planning rates for the 3-player example. When scaled to five players, real-time planning is not achieved but the runtime remains moderate.

The third contribution (Chapter 5) is an extension of the approach proposed in Chapter 4 to scenarios with incomplete knowledge of the objectives of human players. For this purpose, the inference approach presented in Chapter 4 is augmented to additionally reason about latent objective parameters of other players. The augmented approach is evaluated on two different versions of the 3-player intersection driving problem studied in Chapter 4. First, a problem in which the aggressiveness, i.e. the extend to which a players is willing to make room for others, of human players is unknown, and second, a problem where the goal location of human players is unknown. The objective inference approach is compared to an approach that uses only equilibrium inference and the baseline approach of Chapter 4. The advantage of objective inference over all other approaches is clear with respect to both prediction and planning performance. While the baseline approach is clearly dominated, equilibrium inference ignoring objective uncertainty still shows good performance in many cases. It is found that this good performance of equilibrium inference is enabled by a structural similarity between equilibria for different objectives in the studied intersection driving problems. Due to this structure, even if the robot assumes an incorrect objective for a human player, equilibrium inference is still able to predict whether this human will wait or whether she wants to pass first since both behaviors are also captured by the incorrect objective model. Nonetheless, objective inference still improves the performance of the robot, particularly improving the accuracy of long-term predictions and reducing the worst-case cost for the robot in closed-loop interaction with humans.

The final contribution (Chapter 6) is an approach for fitting objective models to datasets of noise-corrupted partial state observations of multi-player interactions; i.e. the inverse differential game problem. The proposed approach works by numerically inverting the solver presented in [1] via differentiable programming, allowing a gradient-based optimization of objective model parameters. A validation on synthetic datasets of observed multi-player

interactions demonstrates that the proposed approach can reliably fit a model that matches the observations and accurately predicts the objective parameters of the model that was used to generate the data.

7.2 Future Work

There are many promising directions for future work to proceed from the results presented in this thesis. The discussion at the end of Chapters 4 to 6 gives short-term specific extensions to the individual research efforts described therein. This section gives a broader overview.

The first line of future work is to test the performance of the proposed planning approach in interaction with real human behavior. In this work, human behavior is assumed to be approximately characterized by a local Nash equilibrium of the game and a Monte Carlo study performed in Chapters 4 and 5 indicates that the corresponding equilibrium strategies constitute plausible behavior. However, it remains to be quantified to which extend real human behavior deviates from this equilibrium concept and further experiments are necessary to test the robustness of the inference-based planning approach in closed-loop interaction with such behavior. In a first step, this could be realized in a human-in-the-loop simulation. Thereafter, validation in a hardware experiment or on a real-world dataset (e.g. [10]) must be performed. Another important investigation in this context is to test whether human equilibrium preferences are fixed, or whether they change over time in response to the decisions of other players. If the latter behavior is observed, the proposed inference method must be adapted to account for the dynamics of these preferences. This can be achieved by defining a transition model for the latent equilibrium state in the inference framework.

The second line is focused on scalability of the proposed planning approach. While this work demonstrates near real-time performance of the proposed planning approach in problems with up to three players and a small number of unknown objective parameters, scaling to more complex problems remains challenging. In particular, the number of samples required to cover high-dimensional latent spaces in problems with many possible objectives of other players is an important limiting factor for the proposed particle filtering approach. Chapter 6 shows that objective model parameters can be estimated using a gradient-based approach to solve the inverse differential game problem. While in this work, this inverse game solver has been discussed as a tool for *offline* estimation of objective models from previously recorded datasets of observed behavior, future work should focus on developing an *online* inference method that uses this approach to update the objective model parameters as new state observations are received. Here, a possible direction is to combine the approaches discussed in Chapters 4 to 6 to a particle filtering technique that uses weighted samples to represent equilibria and updates the objective model for each particle using the inverse game approach. Given the effectiveness of gradient-based methods for optimization of models with many parameters, such as artificial neural networks, research efforts in this direction are also important to improve the flexibility of the planning approach.

Finally, the proposed planning approach can be extended to use more information from the belief it maintains over possible human strategies. In the current formulation, the planning approach uses only the most likely hypothesis in this belief to make control decisions. While this approach is a good approximation in many cases, it may lead to unsafe behavior if there are multiple likely hypotheses in the belief. Since the approach maintains a particle belief over equilibria and corresponding feedback strategies, a wide range of approximate planning techniques for partially observable Markov decision processes may be used to make better decisions in the face of high uncertainty. A commonly used approximation that has been shown to provide good performance in other areas of decision making under uncertainty is generalized QMDP [4, 77]. In the context of the game-theoretic planning problems studied in this work, a QMDP approach requires to solve an optimization problem in which the robots strategy is evaluated against all possible human strategies in the belief. Further research efforts are required to test whether this optimization problem admits a tractable solution.

References

[1] 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", *ArXiv:1909.04694*, 2019.

[2] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games", *Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, 2005.

[3] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin, and A. D. Dragan, "Probabilistically safe robot planning with confidence-based human predictions", *ArXiv:1806.00109*, 2018.

[4] Z. N. Sunberg, C. J. Ho, and M. J. Kochenderfer, "The value of inferring the internal state of traffic participants for autonomous freeway driving", in *American Control Conference (ACC)*, 2017.

[5] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, "A belief state planner for interactive merge maneuvers in congested traffic", in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[6] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.", in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 8, 2008.

[7] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[8] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[9] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[10] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The inD dataset: A drone dataset of naturalistic road user trajectories at german intersections", *ArXiv:1911.07602*, 2019.

[11] P. Cardaliaguet, "Differential games with asymmetric information", *SIAM Journal on Control and Optimization*, vol. 46, no. 3, pp. 816–838, 2007.

[12] R. Buckdahn, P. Cardaliaguet, and M. Quincampoix, "Some recent aspects of differential game theory", *Dynamic Games and Applications*, vol. 1, no. 1, pp. 74–114, 2011.

[13] G. Hexner, "A differential game of incomplete information", *Journal of Optimization Theory and Applications*, vol. 28, no. 2, pp. 213–232, 1979.

[14] C. H. Fitzgerald, "The princess and monster differential game", *SIAM Journal on Control and Optimization*, vol. 17, no. 6, pp. 700–712, 1979.

[15] A. Rapaport and P. Bernhard, "On a planar pursuit game with imperfect knowledge of a coordinate", *RAIRO-APII-JESA-Journal Europeen des Systemes Automatises*, vol. 29, no. 6, pp. 575–602, 1995.

[16] C. Jimenez, M. Quincampoix, and Y. Xu, "Differential games with incomplete information on a continuum of initial positions and without Isaacs condition", *Dynamic Games and Applications*, vol. 6, no. 1, pp. 82–96, 2016.

[17] T. Başar and P. Bernhard, *$\mathcal{H}_\infty$-optimal control and related minimax design problems.* Birkhäuser Boston., 1995.

[18] M. R. James and J. Baras, "Partially observed differential games, infinite-dimensional Hamilton–Jacobi–Isaacs equations, and nonlinear $\mathcal{H}_\infty$ control", *SIAM Journal on Control and Optimization*, vol. 34, no. 4, pp. 1342–1364, 1996.

[19] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[20] M. Simaan and J. B. Cruz, "On the Stackelberg strategy in nonzero-sum games", *Journal of Optimization Theory and Applications*, vol. 11, no. 5, pp. 533–555, 1973.

[21] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions.", in *Robotics: Science and Systems*, 2016.

[22] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[23] J. H. Yoo and R. Langari, "Stackelberg game based model of highway driving", in *ASME Dynamic Systems and Control Conference joint with the JSME Motion and Vibration Conference*, 2012.

[24] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[25] E. Mazumdar and L. J. Ratliff, "On the convergence of gradient-based learning in continuous games", *ArXiv:1804.05464*, 2018.

[26] R. Spica, D. Falanga, E. Cristofalo, E. Montijano, D. Scaramuzza, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing", *ArXiv:1801.02302*, 2018.

[27] E. Ward, N. Evestedt, D. Axehill, and J. Folkesson, "Probabilistic model for interaction aware planning in merge scenarios", *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 2, pp. 133–146, 2017.

[28] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic", *ArXiv:1906.11021*, 2019.

[29] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations", *Physical Review E*, vol. 62, no. 2, p. 1805, 2000.

[30] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model MOBIL for car-following models", *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[31] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving", in *International Conference on Robotics and Automation (ICRA)*, 2015.

[32] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction.", in *Robotics: Science and Systems*, vol. 1, 2015.

[33] D. Mehta, G. Ferrer, and E. Olson, "Autonomous navigation in dynamic social environments using multi-policy decision making", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[34] D. Fridovich-Keil, A. Bajcsy, J. F. Fisac, S. L. Herbert, S. Wang, A. D. Dragan, and C. J. Tomlin, "Confidence-aware motion prediction for real-time collision avoidance", *The International Journal of Robotics Research*, p. 0 278 364 919 859 436, 2019.

[35] A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, J. F. Fisac, S. Deglurkar, A. D. Dragan, and C. J. Tomlin, "A scalable framework for real-time multi-robot, multi-human collision avoidance", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[36] C. L. Baker, J. B. Tenenbaum, and R. R. Saxe, "Goal inference as inverse planning", in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 29, 2007.

[37] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. SIAM, 1999.

[38] L. C. Evans and P. E. Souganidis, "Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations.", Wisconsin University Mathematics Research Center, Tech. Rep., 1983.

[39] P.-L. Lions and P. E. Souganidis, "Differential games, optimal control and directional derivatives of viscosity solutions of Bellman's and Isaacs' equations", *Journal on Control and Optimization*, vol. 23, no. 4, pp. 566–583, 1985.

[40] R. Isaacs, *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. Courier Corporation, 1999.

[41] V. Conitzer and T. Sandholm, "Complexity results about Nash equilibria", in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2003.

[42] E. V. Mazumdar, M. I. Jordan, and S. S. Sastry, "On finding local Nash equilibria (and only local Nash equilibria) in zero-sum games", *ArXiv:1901.00838*, 2019.

[43] Z. Wang, R. Spica, and M. Schwager, "Game theoretic motion planning for multi-robot racing", in *Distributed Autonomous Robotic Systems*, Springer, 2019, pp. 225–238.

[44] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game theoretic planning for self-driving cars in competitive scenarios", in *Robotics: Science and Systems*, 2019.

[45] S. L. Cleac'h, M. Schwager, and Z. Manchester, "Algames: A fast solver for constrained dynamic games", *ArXiv:1910.09713*, 2019.

[46] M. Chen, J. F. Fisac, S. Sastry, and C. J. Tomlin, "Safe sequential path planning of multi-vehicle systems via double-obstacle Hamilton-Jacobi-Isaacs variational inequality", in *European Control Conference (ECC)*, 2015.

[47] D. Fridovich-Keil, V. Rubies-Royo, and C. J. Tomlin, "An iterative quadratic method for general-sum differential games with feedback linearizable dynamics", *ArXiv:1910.00681*, 2019.

[48] M. Green and D. J. Limebeer, *Linear Robust Control*. Courier Corporation, 2012.

[49] H. Mukai, A. Tanikawa, I. Tunay, I. Katz, H. Schättler, P. Rinaldi, I. Ozcan, G. Wang, L. Yang, and Y. Sawada, "Sequential linear quadratic method for differential games", in *In Proceedings of the DARPA-JFACC Symposium on Advances in Enterprise Control*, 2000.

[50] A. Tanikawa, H. Mukai, and M. Xu, "Local convergence of the sequential quadratic method for differential games", *Transactions of the Institute of Systems, Control and Information Engineers*, vol. 25, no. 12, pp. 349–357, 2012.

[51] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems", in *ICINCO*, 2004.

[52] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems", in *American Control Conference (ACC)*, 2005.

[53] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[54] M. Kochenderfer, *Decision Making Under Uncertainty: Theory and Application*, ser. MIT Lincoln Laboratory Series. MIT Press, 2015.

[55] D. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[56] M. H. DeGroot and M. J. Schervish, *Probability and Statistics*. Pearson Education, 2012.

[57] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems", *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[58] M. Simandl and O. Straka, "Sampling densities of particle filter: A survey and comparison", in *American Control Conference (ACC)*, 2007.

[59] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation", in *IEE Proceedings F*, vol. 140, 1993.

[60] N. M. Kwok, G. Fang, and W. Zhou, "Evolutionary particle filter: Re-sampling from the genetic algorithm perspective", in *International Conference on Intelligent Robots and Systems (IROS)*, 2005.

[61] J. D. Hol, T. B. Schön, and F. Gustafsson, "On resampling algorithms for particle filters", in *Proceedings of the IEEE Nonlinear Statistical Signal Processing Workshop*, 2006.

[62] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filtering: Classification, implementation, and strategies", *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 70–86, 2015.

[63] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters", *Journal of the American statistical association*, vol. 94, no. 446, pp. 590–599, 1999.

[64] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing", *ArXiv:1209.5145*, 2012.

[65] J. Revels, M. Lubin, and T. Papamarkou, "Forward-mode automatic differentiation in Julia", *ArXiv:1607.07892*, 2016.

[66] J. Regier, K. Fischer, K. Pamnany, A. Noack, J. Revels, M. Lam, S. Howard, R. Giordano, D. Schlegel, J. McAuliffe, *et al.*, "Cataloging the visible universe through Bayesian inference in Julia at petascale", *Journal of Parallel and Distributed Computing*, vol. 127, pp. 89–104, 2019.

[67] S. J. DeCanio and A. Fremstad, "Game theory and climate diplomacy", *Ecological Economics*, vol. 85, pp. 177–187, 2013.

[68] P. G. Straub, "Risk dominance and coordination failures in static games", *Quarterly Review of Economics and Finance*, vol. 35, no. 4, pp. 339–364, 1995.

[69] L. H. Summers, "International financial crises: Causes, prevention, and cures", *American Economic Review*, vol. 90, no. 2, pp. 1–16, 2000.

[70] D. Schmidt, R. Shupp, J. M. Walker, and E. Ostrom, "Playing safe in coordination games:: The roles of risk dominance, payoff dominance, and history of play", *Games and Economic Behavior*, vol. 42, no. 2, pp. 281–299, 2003.

[71] J. C. Harsanyi, R. Selten, *et al.*, "A general theory of equilibrium selection in games", *MIT Press Books*, vol. 1, 1988.

[72] J. C. Harsanyi, "The tracing procedure: A Bayesian approach to defining a solution for n-person noncooperative games", *International Journal of Game Theory*, vol. 4, no. 2, pp. 61–94, 1975.

[73] ——, "A new theory of equilibrium selection for games with complete information", *Games and Economic Behavior*, vol. 8, no. 1, pp. 91–122, 1995.

[74] J. B. Van Huyck, J. P. Cook, and R. C. Battalio, "Adaptive behavior and coordination failure", *Journal of Economic Behavior and Organization*, vol. 32, no. 4, pp. 483–503, 1997.

[75] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains", *Artificial Intelligence*, vol. 101, no. 1, pp. 99–134, 1998, ISSN: 0004-3702.

[76] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes", *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987.

[77] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up", in *International Conference on Machine Learning (ICML)*, 1995.

[78] L. Peters, "Partially observable Markov decision processes for planning in uncertain environments", Project Thesis, Aug. 2019.

[79] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.", in *International Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 96, 1996.

[80] A. K. Jain, "Data clustering: 50 years beyond K-means", *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[81] J. Hu, M. Prandini, and S. Sastry, "Optimal maneuver for multiple aircraft conflict resolution: A braid point of view", in *IEEE Conference on Decision and Control (CDC)*, vol. 4, 2000, pp. 4164–4169.

[82] S.-Y. Ko, J. Kim, W.-J. Lee, and D.-S. Kwon, "Surgery task model for intelligent interaction between surgeon and laparoscopic assistant robot", *International Journal of Assitive Robotics and Mechatronics*, vol. 8, no. 1, pp. 38–46, 2007.

[83] J. Shi, G. Jimmerson, T. Pearson, and R. Menassa, "Levels of human and robot collaboration for automotive manufacturing", in *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, 2012.

[84] G. A. Bekey, "System identification – an introduction and a survey", *Simulation*, vol. 15, pp. 151–166, 1970.

[85] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning", *ArXiv:1507.04888*, 2015.

[86] S. Rothfuß, J. Inga, F. Köpf, M. Flad, and S. Hohmann, "Inverse optimal control for identification in non-cooperative differential games", *IFAC World Congress*, vol. 50, no. 1, pp. 14 909–14 915, 2017.

[87] T. L. Molloy, G. S. Garden, T. Perez, I. Schiffner, D. Karmaker, and M. V. Srinivasan, "An inverse differential game approach to modelling bird mid-air collision avoidance behaviours", *IFAC World Congress*, vol. 51, no. 15, pp. 754–759, 2018.

[88] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes", in *European Conference on Computer Vision (ECCV)*, 2016.

[89] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method", *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, 2005.

[90] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization", in *IEEE swarm intelligence symposium*, 2007.

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

[92] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications", *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[93] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey", *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5595–5637, 2017.

[94] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory", in *Numerical Analysis*, Springer, 1978, pp. 105–116.

[95] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[96] N. Qian, "On the momentum term in gradient descent learning algorithms", *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

Appendix A

Planning Performance with Increased Number of Particles

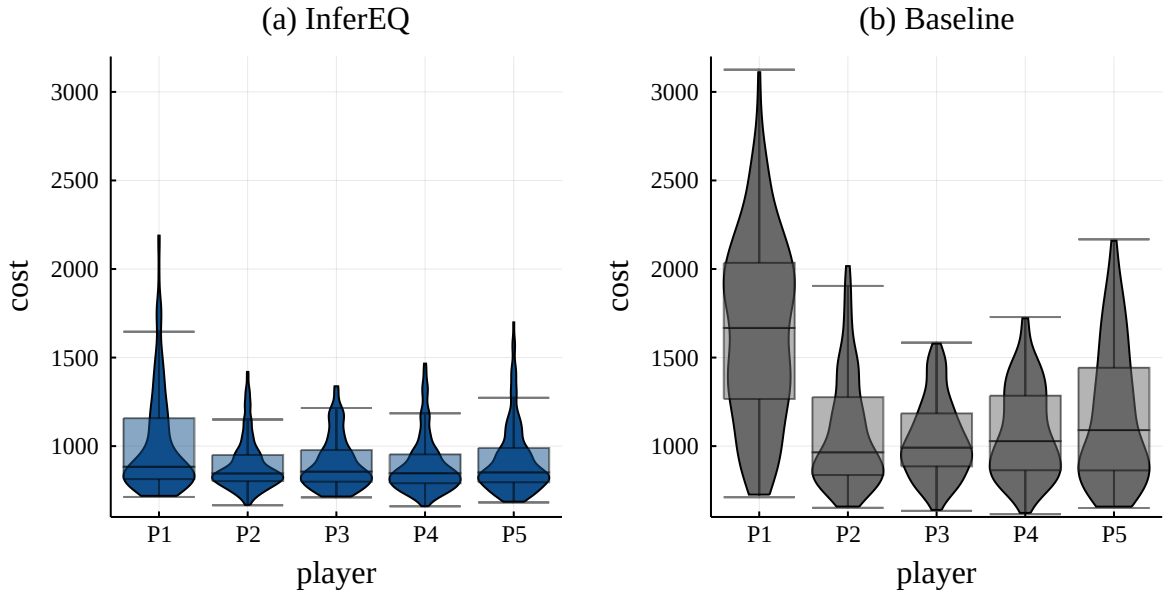


Figure A.1.1: Distribution of costs for strategy alignment in a 5-player navigation problem for different numbers of particles. (a) Player-1 uses strategy alignment with 150 particles. (b) Player-1 uses strategy alignment with 300 particles.

Appendix B

Complete Monte-Carlo Study for Chapter 5

Below, the complete Monte Carlo study for two evaluation problems of Section 5.3 are presented. Each figure is created from 300 random initial strategy profile and equilibria are grouped via spatial clustering of the trajectories. Within each figure, clusters are sorted from left to right in ascending order of cost incurred by Player-1. The initial position of each player as well as their position at half the simulation horizon is highlighted, respectively, with a rectangular and a circular marks.

B.1 Uncertain Aggressiveness

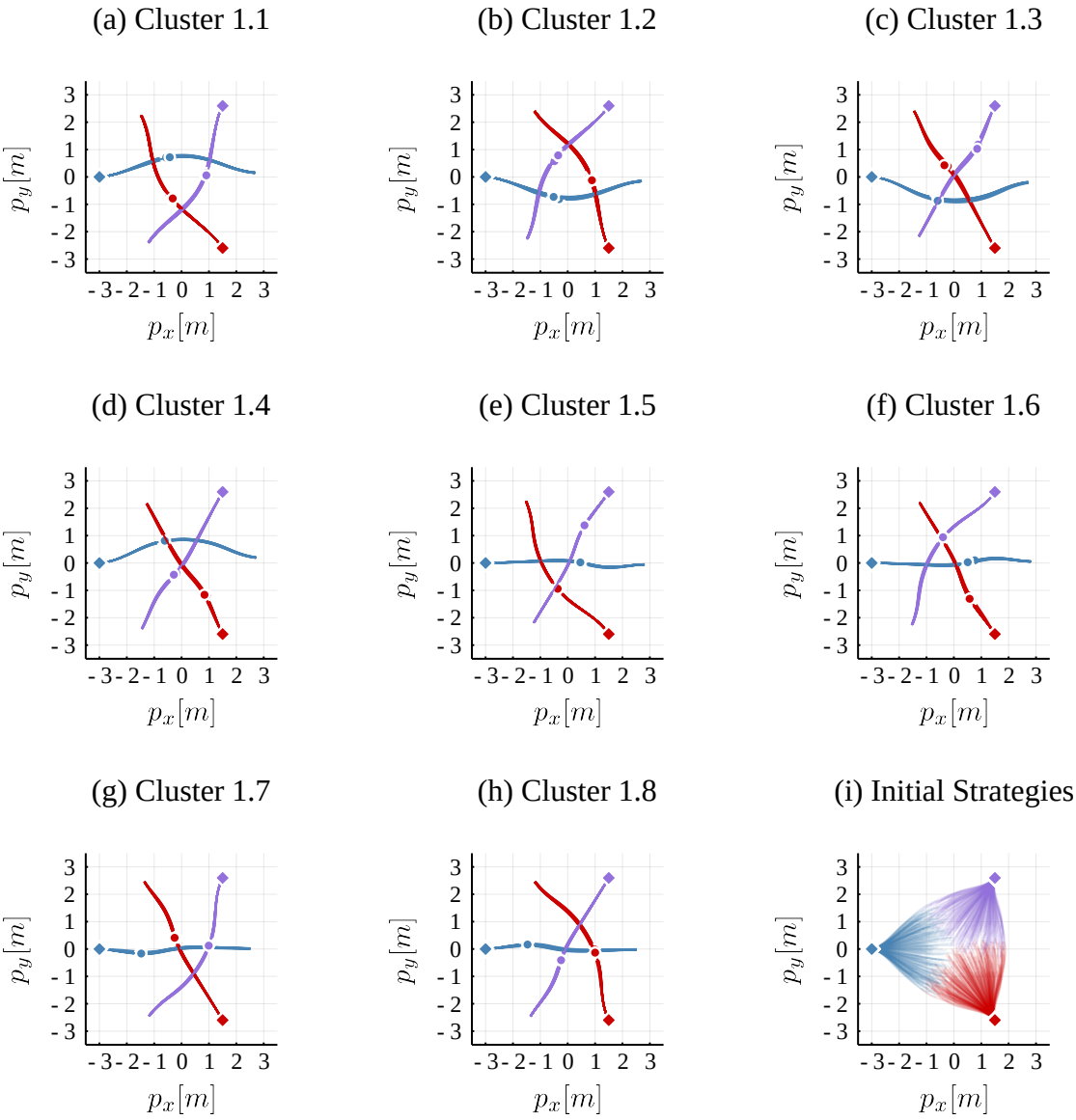


Figure B.1.1: Clustered local equilibria if both human players have a *high* proximity cost weight (non-aggressive). $\Theta = [c_{\text{prox},2}; c_{\text{prox},3}] = [\bar{c}_{\text{prox}}; \bar{c}_{\text{prox}}]$.

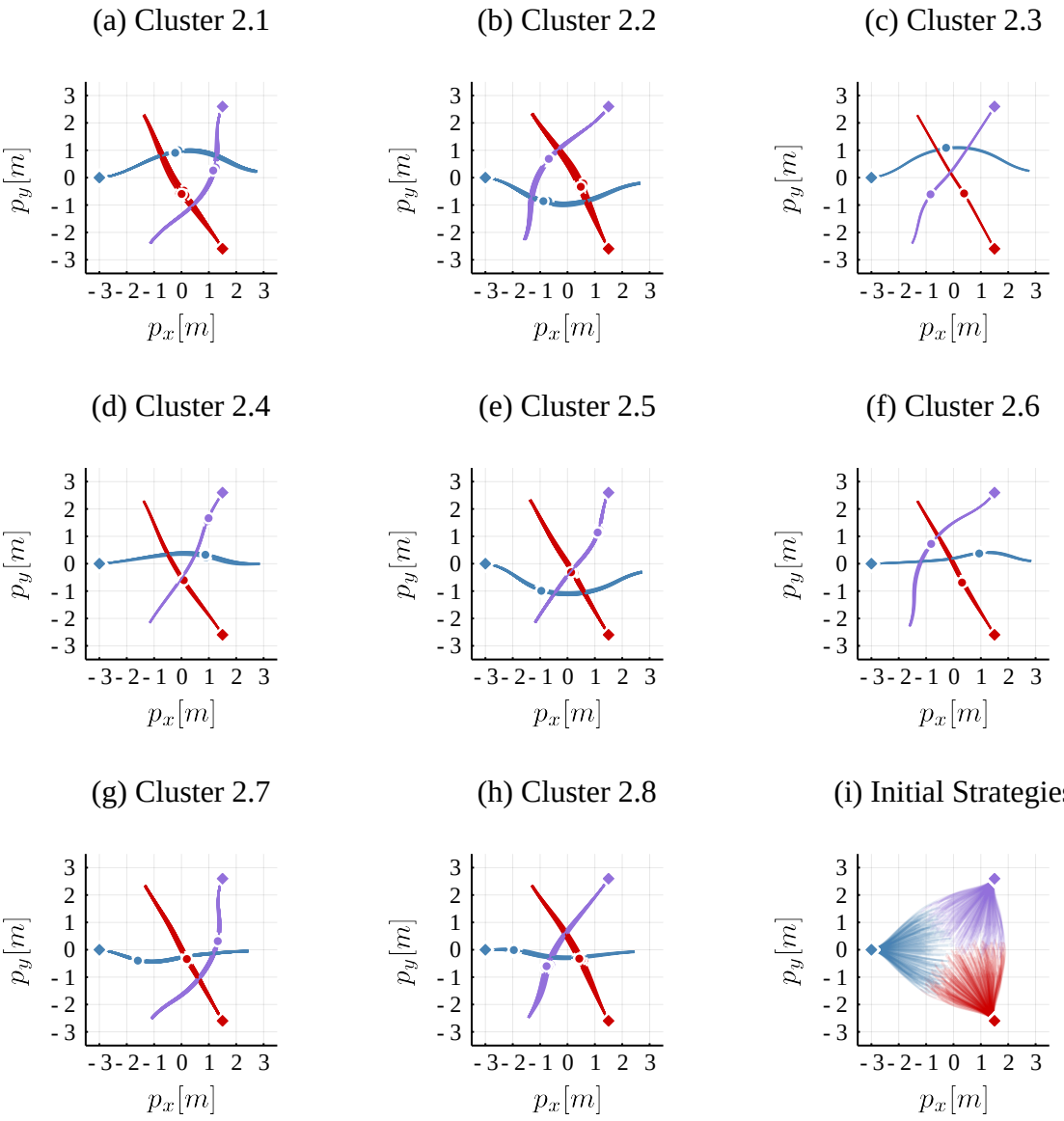


Figure B.1.2: Clustered local equilibria if Player-2 (red) has a *low* proximity cost weight (aggressive) and Player-3 (purple) has a *high* proximity cost weight (non-aggressive). $\Theta = [c_{\text{prox},2}; c_{\text{prox},3}] = [\underline{c}_{\text{prox}}; \bar{c}_{\text{prox}}]$.

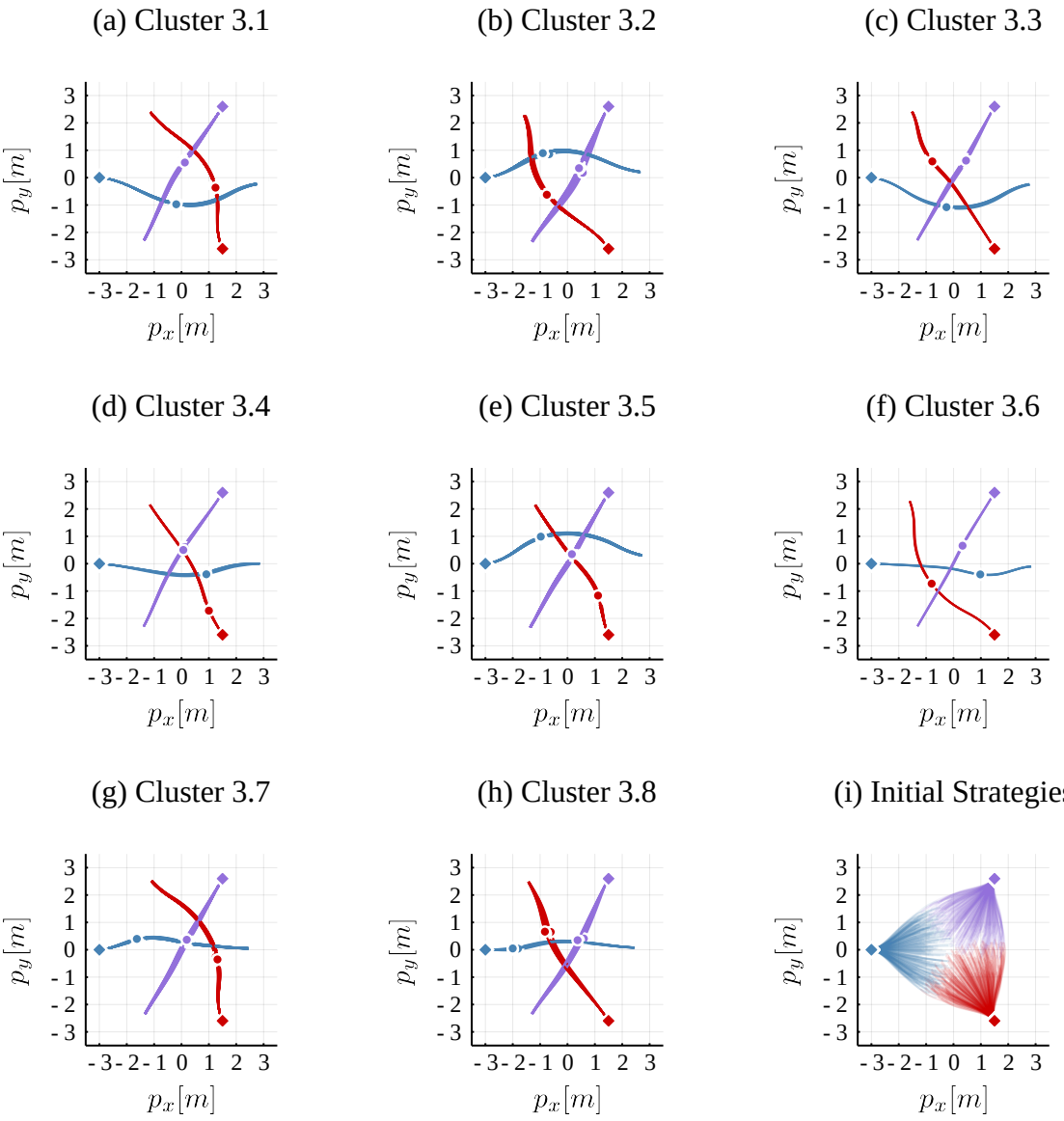


Figure B.1.3: Clustered local equilibria if Player-2 (red) has a *high* proximity cost weight (non-aggressive) and Player-3 (purple) has a *low* proximity cost weight (aggressive). $\Theta = [c_{\text{prox},2}; c_{\text{prox},3}] = [\bar{c}_{\text{prox}}; \underline{c}_{\text{prox}}]$.

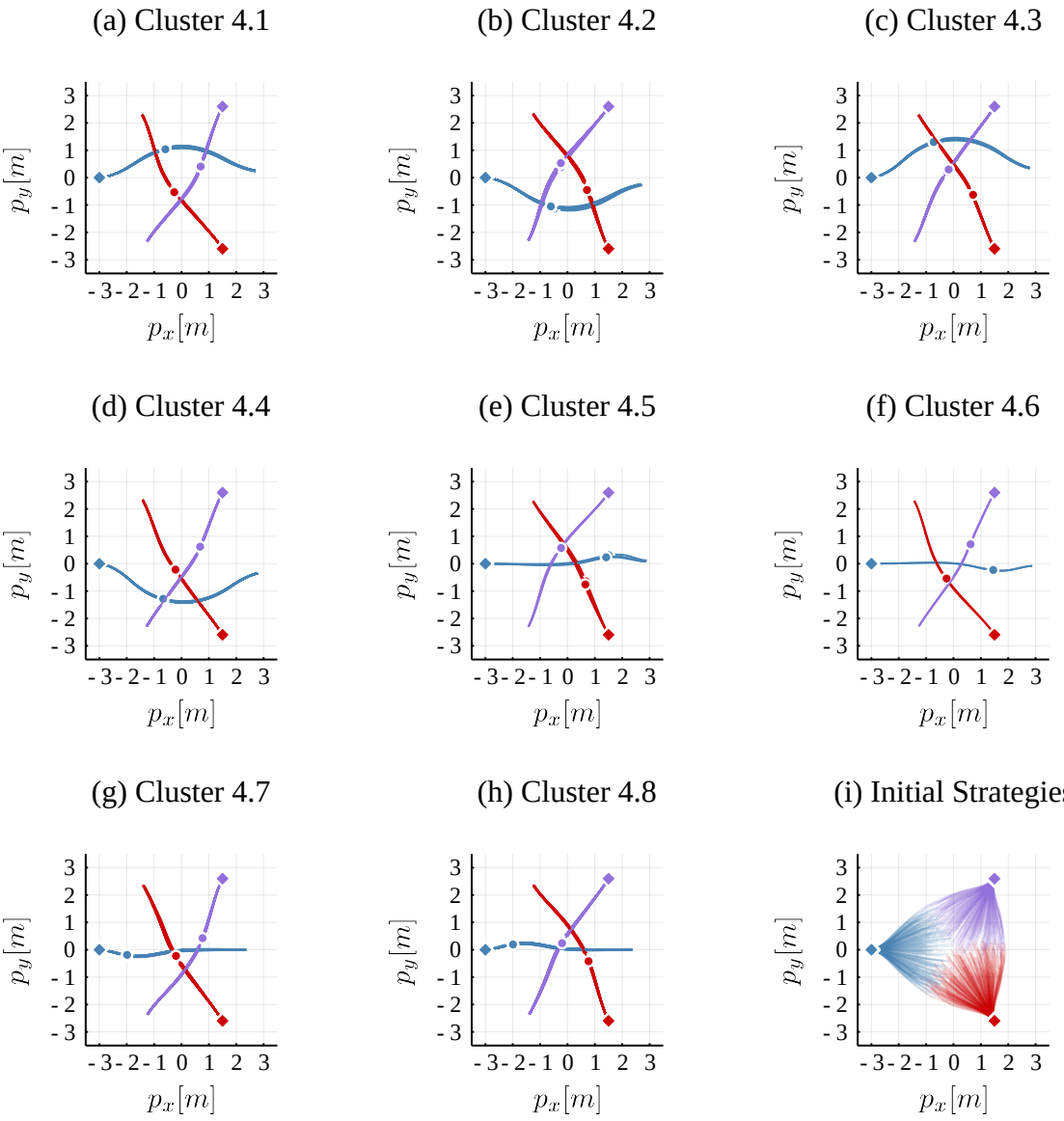


Figure B.1.4: Clustered local equilibria if both players have a *low* proximity cost weight (aggressive). $\Theta = [c_{\text{prox},2}; c_{\text{prox},3}] = [\underline{c}_{\text{prox}}; \underline{c}_{\text{prox}}]$.

B.2 Uncertain Goal Positions

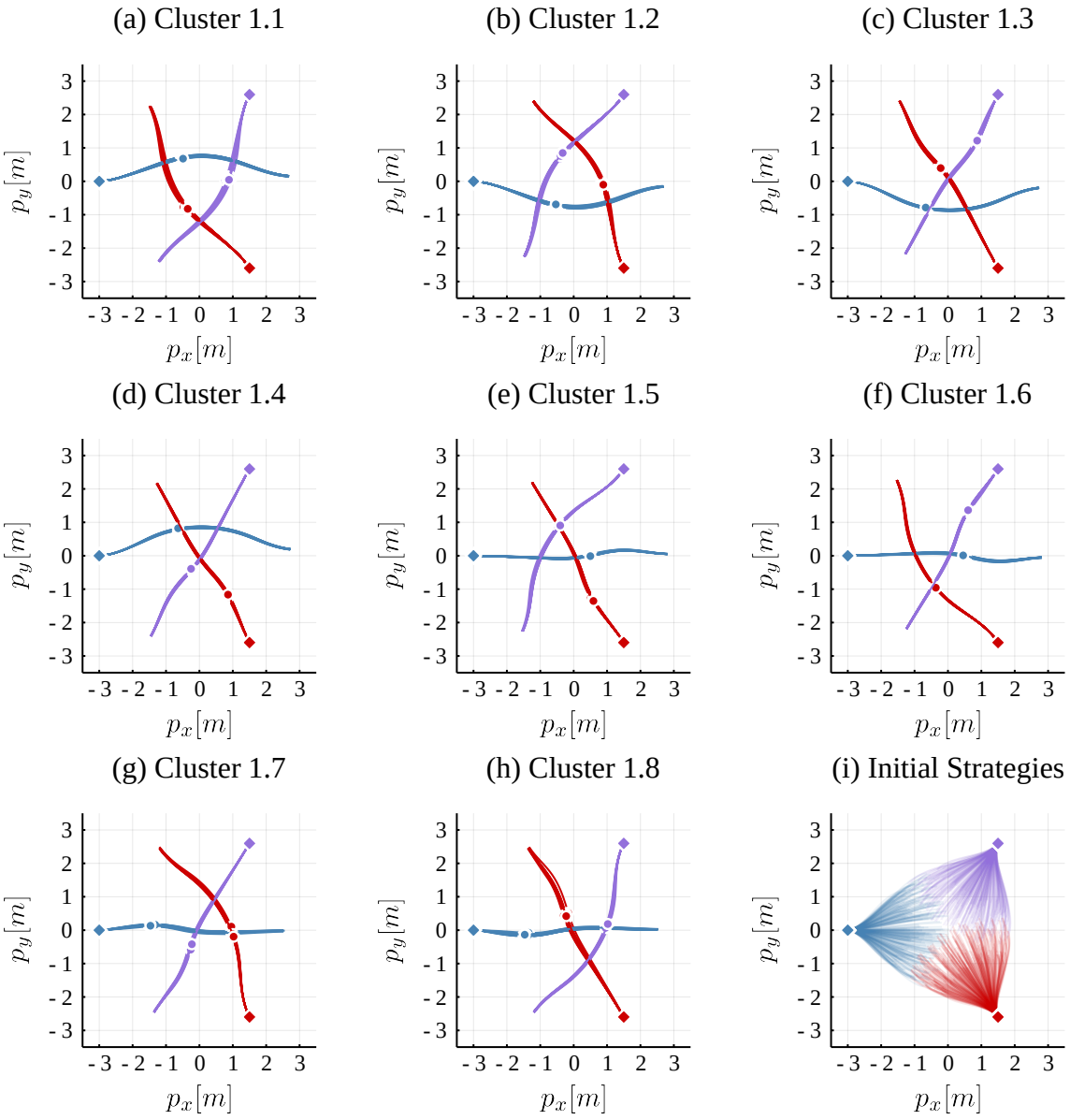


Figure B.2.1: Clustered local equilibria if both human players want to go straight.

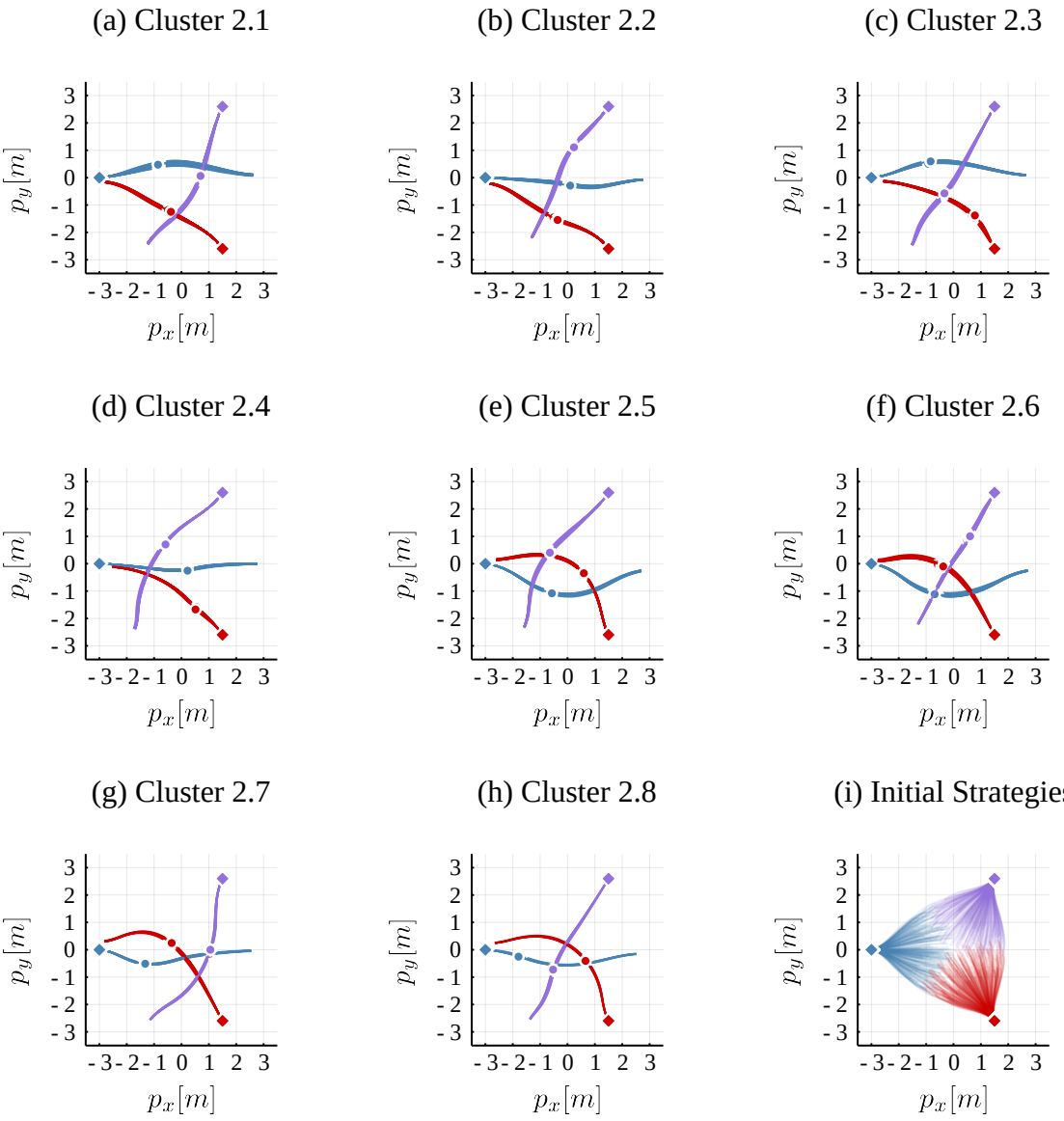


Figure B.2.2: Clustered local equilibria if Player-2 (red) wants to turn left and Player-3 (purple) wants to go straight.

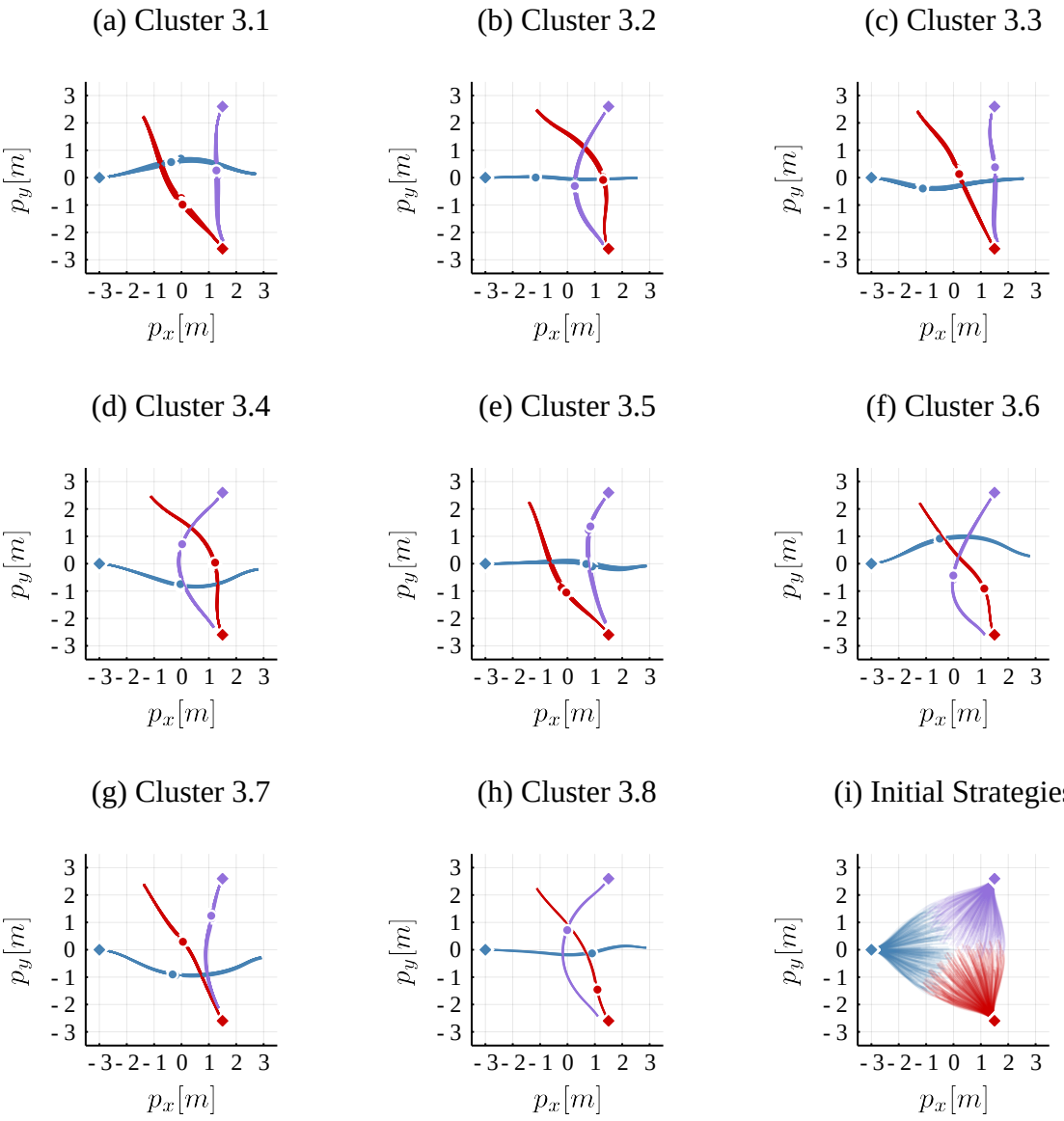


Figure B.2.3: Clustered local equilibria if Player-2 (red) wants to go straight and Player-3 (purple) wants to turn left.

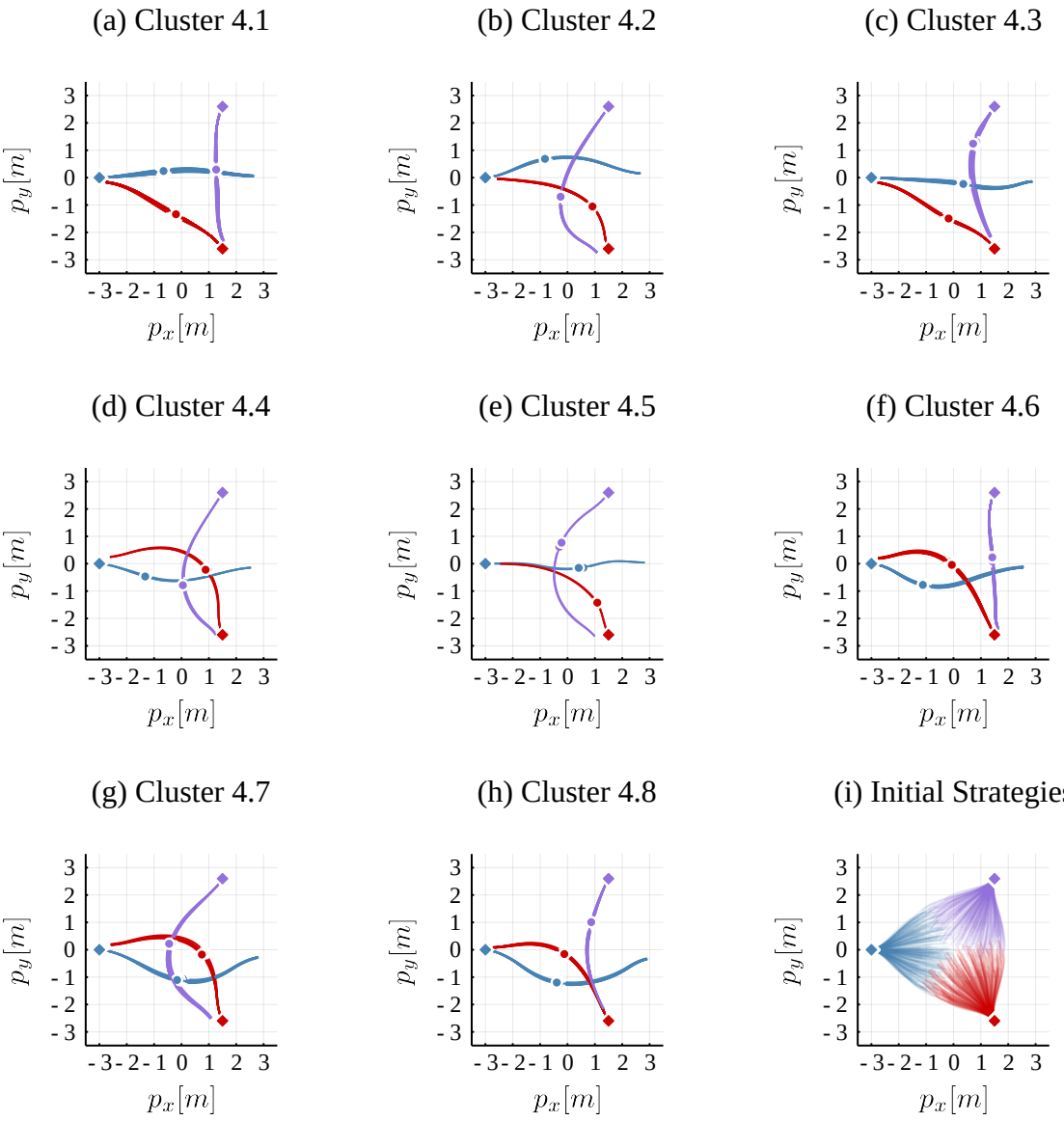


Figure B.2.4: Clustered local equilibria if both human players want to turn left.

Appendix C

Supplementary Example for Chapter 6

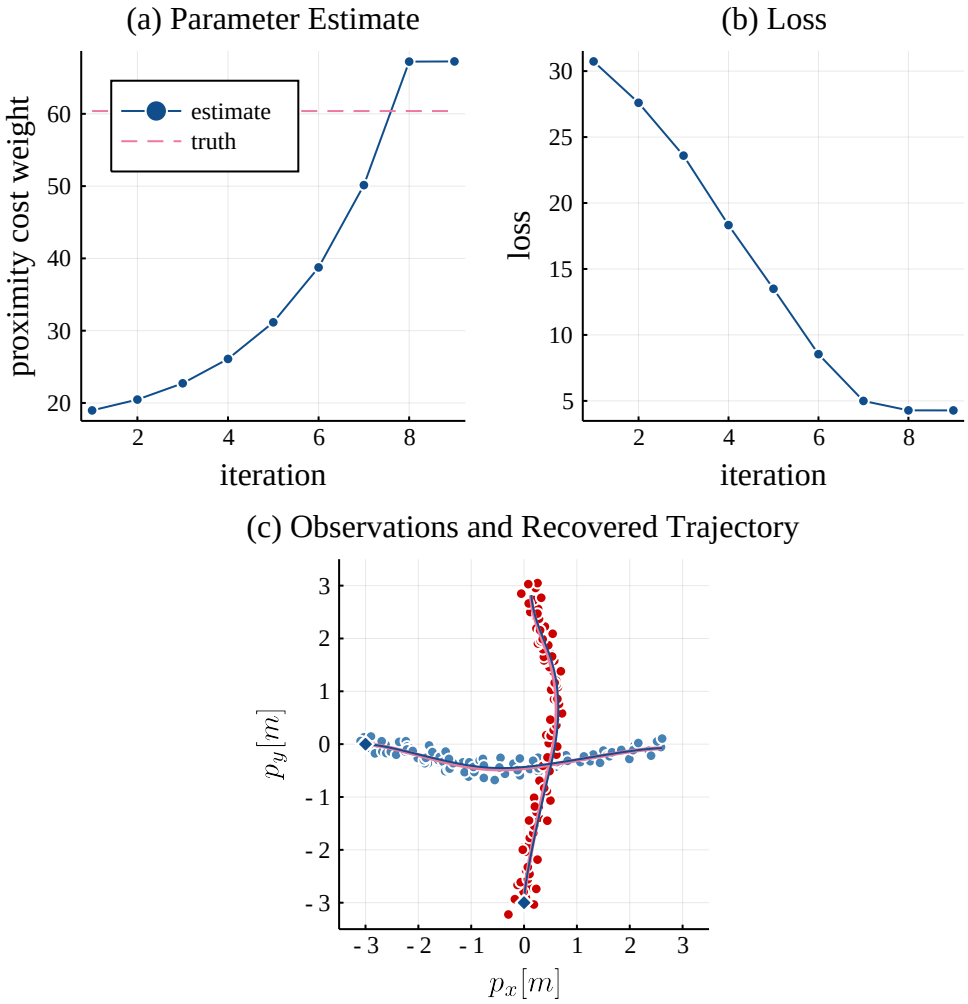


Figure C.1.1: Example of inverse ILQG with highest final error among all simluations. In (c), scattered points indicate observed position measurements, solid violet lines indicate the true game solution, and solid blue lines correspond to the recovered game solution for the final parameter estimate.