DISTRIBUTED GAME-THEORETIC TRAJECTORY PLANNING FOR MULTI-AGENT INTERACTIONS

BY

ZACHARY JAMES WILLIAMS

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

Adviser:

Assistant Professor Negar Mehr

ABSTRACT

In this work, we develop a scalable, local trajectory optimization algorithm that enables robots to interact with other agents. It has been shown that the interactions of multiple agents can be successfully captured in game-theoretic formulations, where the interaction outcome can be best modeled via the equilibria of the underlying dynamic game. However, it is challenging to compute equilibria of dynamic games, as it involves simultaneously solving a set of coupled optimal control problems. Existing solvers operate in a centralized fashion and do not scale up tractably to multiple interacting agents. We enable scalable distributed game-theoretic planning by leveraging the structure inherent in multi-agent interactions, namely, interactions belonging to the class of dynamic potential games. Since equilibria of dynamic potential games can be found by minimizing a single potential function, we can apply distributed and decentralized control techniques to seek equilibria of multi-agent interactions in a scalable and distributed manner. We compare the performance of our algorithm with a centralized interactive planner in a number of simulation studies and demonstrate that our algorithm results in better efficiency and scalability. We further evaluate our method in hardware experiments involving multiple quadcopters.

To my dad, for his steadfast love and relentless support.

ACKNOWLEDGMENTS

I would like to sincerely thank my adviser, Dr. Mehr, for leading me through this process and forging me into the researcher I am today through the exciting discoveries and the disheartening failures. I am indebted to her unwavering dedication to her students' growth and to solving important robotics problems. Another individual who spent much time in the trenches with me in this effort is Randy Chen, without whom this would not have been possible.

Several faculty here at UIUC have significantly impacted my academic journey through their exceptional teaching, including Dr. Roy Dong, Dr. Saurabh Gupta, and Dr. Zhi-Pei Liang. I extend my sincere gratitude toward my colleagues at Raytheon BBN Technologies for always pushing me as an engineer and fostering the early phases of my career including Dr. Vince Radzicki, Justin Swartz, Dr. Collin Smith, Gigs Gregory, and Matt Rebholz. I am also immensely grateful for my company providing financial assistance for me to attend graduate school.

I cannot talk about my time here in Illinois without also thanking my roommates. For the late night talks by the fire, for the random existential debates, for the meals we shared together, and for y'all's constant friendship and community.

Lastly, I'd like to express my deepest thanks to my family - Beth, Lauren, Josh, and Dad, who are always present in my life and have collectively made me into who I am today.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ADMM	Alternating Direction Method of Multipliers
DDP	Differential Dynamic Programming
DP-iLQR	Distributed Potential iLQR
LQR	Linear Quadratic Regulator
iLQG	Iterative Linear Quadratic Gaussian
iLQR	Iterative LQR
MAS	Multi-Agent System
MPC	Model Predictive Control
NE	Nash Equilibrium
OC	Optimal Control
ToM	Theory of Mind
UAV	Unmanned Aerial Vehicle

1 INTRODUCTION

Automatically generating intuitive trajectories in interactive robotic applications involving multiple agents is an important and challenging problem [1]. In situations where scalability matters, we need algorithms that enable robots to safely and tractably interact with other agents. For example, crowd robot navigation may require a robot to navigate among multiple humans. Alternatively, drone delivery systems may require quadcopters to navigate by multiple drones in an aerial space. Autonomous driving among human drivers is another highly relevant problem where agents must reason about their impacts on others. A critical challenge in trajectory planning in such interactive settings is that each robot must account for the likely reactions of other agents to its actions, which results in a coupling among the agents, accounting for which quickly becomes intractable as the number of agents increases. In this work, we address this challenge and develop a scalable trajectory optimization algorithm that enables robots to interact efficiently with other navigating agents.

One concept in psychology that relates to this recursive problem in robotics relates to the theory of mind (ToM), which is a person's ability to associate mental states such as thoughts, emotions, and desires to others and reason about how those affect their behavior [2]. One test commonly conducted to quantify ToM is the Imposing Memory Task [3]. Participants are tasked with reading a series of complicated stories involving reasoning about various social situations. They then fill out a questionnaire regarding their understanding of the psychological dynamics behind the characters' actions. This asks them questions about first order beliefs - "Sam wanted to go to the Post Office to buy a stamp," to fifth order beliefs with increasing recursion across multiple characters - "John thought that Pamela thought that he, John, wanted that Pamela discovered what Sara wanted to do because John wanted to go out alone with Pamela." Similarly, multi-agent planning problems require agents to reason about other agents' intentions and how context

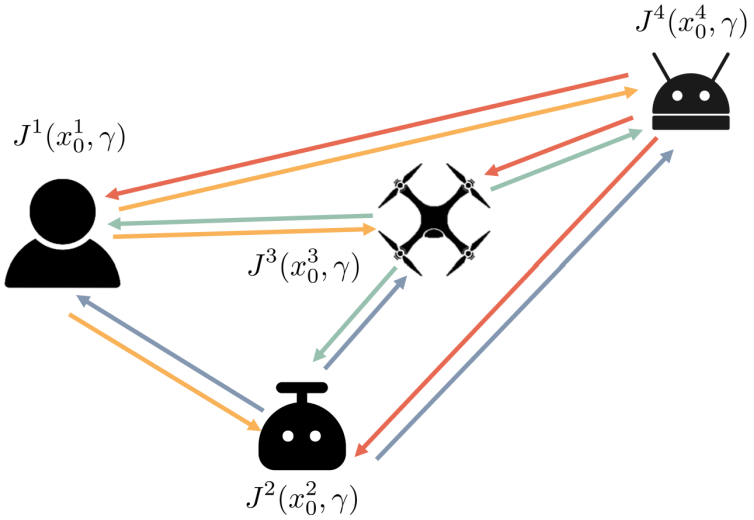

Figure 1.1: Heterogeneous multi-agent navigation problem featuring four agents, each with separate cost functions $J^i(\cdot)$ for agent i. Starting from some initial condition x_0^i, agent i would like to navigate around the other agents to a given target state.

shapes them.

Game-theoretic planning has proven to be a powerful framework for capturing interactions between independent agents [4, 5], where an agent seeks to maximize its utility. Since agents' utilities depend on the state and actions of all agents, the interaction outcome can be best represented via equilibria that account for the mutual influence of agents. While conceptually powerful for modeling a large variety of problems, solving for the equilibria is difficult, as it requires navigating a set of coupled optimal control problems. Several recent works have developed approximate trajectory optimization algorithms for such games [6–8]. However, all these works execute in a centralized fashion, i.e. they require the robot to account for interactions with all agents in the environment. As a result, they rarely extend beyond three agents, even for simple dynamics models. Section 2.1 will briefly introduce the reader to foundational concepts in optimal control, and Section 2.2 will provide greater context for game theoretic interactions and notions of equilibria.

1.1 Contributions

We consider the couplings that result from agents' interactions in a game-theoretical setup and draw from decentralized and distributed MPC literature to develop a distributed trajectory planner that can be run in a receding horizon fashion efficiently and scalably. We leverage the fact that certain multi-agent interactions are part of a more special class of games, namely dynamic potential games [9]. Potential games are a class of games for which equilibria can be found efficiently and tractably by solving a single optimization problem. Our key insight is that since equilibria of dynamic potential games can be found by minimizing a single potential function, we can apply techniques from distributed and decentralized model predictive control to seek Nash equilibria of game-theoretic interactions in a scalable and efficient manner. Section 2.3 introduces the notions of distributed and decentralized control architectures in more vivid detail.

Our algorithm is distributed because each agent independently computes its control input using the state information of its neighboring agents and requires no centralized coordinator to solve its local problem. This architectural modification allows the algorithm to scale to arbitrary numbers of agents. Additionally, realistic problems do not have access to any centralized source of information and are decentralized in structure [10, 11]. When driving on the highway, humans must make decisions based on what they can currently see. We compare the performance of our algorithm with a centralized interactive planner in a number of simulation studies and demonstrate that our algorithm can efficiently account for interactions with a larger number of agents. We further showcase the success of our method in hardware experiments involving multiple quadcopters.

We note that this thesis is based on the work of the authors in [12], which was accepted at the International Conference of Robotics and Automation 2023.

1.2 Thesis Organization

The organization of this thesis is as follows. In Chapter 2, we review relevant background material. We put this work into the context of the literature in

Chapter 3. In Chapter 4, we introduce the planning problem to be solved. In Chapter 5, we discuss the previous work that utilizes dynamic potential games. Our distributed trajectory planning algorithm is defined in Chapter 6. In Chapter 7, we highlight empirical results in simulations and on hardware. We conclude the thesis in Chapter 8.

2 BACKGROUND

This thesis builds on a variety of tools from the fields of optimal control (OC) to solve the optimization problem, game theory to model the outcomes of multi-agent interactions, and distributed architectures to enable scalability. This section highlights several of these foundational methods and concepts. Those comfortable with the aforementioned topics can skip ahead to Chapter 3 to explore the related works.

2.1 Optimal Control

The Linear Quadratic Regulator (LQR) is one of the most significant contributions to the field of OC to date, finding a multitude of applications in unmanned aerial vehicle (UAV) control [13–15], spacecraft control systems [16–18], power and energy system stabilization [19–22], and modeling chemical and biological phenomena [23–25], to name a few. For those coming from an estimation background, LQR is the analog to the Kalman filter [26] (see [27] for a friendlier introduction), another name for which is the Linear Quadratic Estimator (LQE), which refers to the fact that its construction relies on the system having linear dynamics and quadratic estimate covariance.

In this analysis, we consider the problem of steering a single agent toward some goal state using LQR over some finite horizon $T \in \mathbb{Z}$. More generally, we can think of such an agent or robot as a dynamical system. Let the state of a dynamical system at timestep k be denoted as $x_k \in \mathbb{R}^n$, where n is the dimension of the state space. The corresponding control input at time k is similarly written as $u_k \in \mathbb{R}^m$, with m serving as the control dimension and $u_{0:T-1} := (u_0, u_1, \ldots, u_{T-1})$ being the sequence of control inputs over the horizon T. The dynamics of a system $f(x, u) : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$ can be expressed as

$$x_{k+1} = f(x_k, u_k). \tag{2.1}$$

The dynamics uniquely define how a system traverses the state space using its controls. In order to formulate the optimal control problem, we require a cost or objective function $J(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^{m\times T} \mapsto \mathbb{R}$ which we would like to minimize.

Definition 1. *The optimal control problem for costs $J(\cdot,\cdot)$ and dynamics $f(\cdot,\cdot)$ for a single agent from initial condition x_0 is defined as follows:*

$$\begin{aligned} &\min_{u_{0:T-1}} J(x_0, u_{0:T-1}), \\ &s.t.\ x_{k+1} = f(x_k, u_k) \quad \forall k. \end{aligned} \tag{2.2}$$

2.1.1 Linear Dynamics and Quadratic Costs

For the purposes of this introduction to LQR, we focus on a subset of the more general nonlinear dynamics expressed in (2.1) to the following linear time-invariant model:

$$x_{k+1} = Ax_k + Bu_k, \tag{2.3}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times m}$ are constant matrices defining the linear transformation of the current state and control input to the next state. Additionally, we restrict the cost function $J(\cdot,\cdot)$ to be quadratic in both the states and controls.

Definition 2. *Let the Discrete-Time Finite Horizon LQR Problem for an initial condition of x_0 be defined below:*

$$\min_{u_{0:T-1}} x_T^\intercal Q_f x_T + \sum_{k=0}^{T-1} x_k^\intercal Q x_k + u_k^\intercal R u_k, \tag{2.4}$$

$$s.t.\ x_{k+1} = Ax_k + Bu_k \quad \forall k, \tag{2.5}$$

where $Q \in \mathbb{R}^{n\times n}$ weights the state costs and is symmetric and positive semi-definite, which we write as $Q = Q^\intercal \succeq 0$, $R \in \mathbb{R}^{m\times m}$ is some positive definite matrix that weights the various dimensions of the control effort, $R = R^\intercal \succ 0$, and $Q_f \in \mathbb{R}^{n\times n}$ forms the state cost at the terminal timestep, $Q_f = Q_f^\intercal \succeq 0$.

In Equation (2.4), the ultimate goal is to drive the system toward equilibrium where $x_T = 0$ with minimal control input. Thus, we balance trajectory

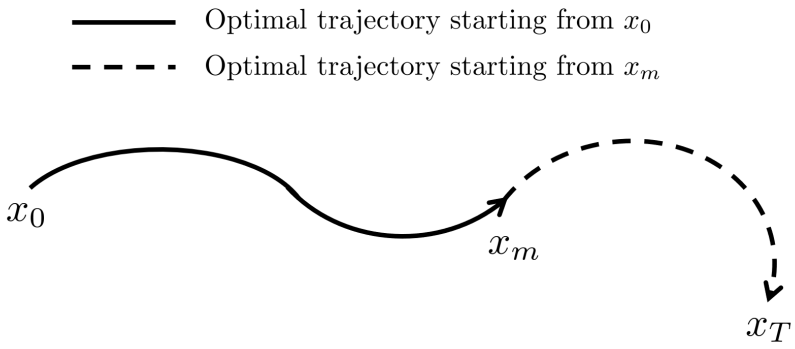

Figure 2.1: Visualization of Bellman's Principle of Optimality.

tracking error with what could be interpreted as fuel consumption or motor torque.

Let the value function $V_k^*(x)$ from a given state x at time k be written as:

$$V_k^*(x) = \min_{u_{k:T-1}} J(x, u_{k:T-1}). \tag{2.6}$$

Therefore, if we had a way to solve for this $V_k^*(x)$, the LQR problem would be solved, since by definition, this is a minimizer of $J(\cdot, \cdot)$ starting from the initial condition x at time k.

To carry on with the derivation, we invoke Bellman's Principle of Optimality, where the core idea is that if some optimal trajectory were to exist, all its subintervals would also be optimal. More precisely, we ultimately seek to find the optimal trajectory over the interval $t \in [T] := \{0, 1, \ldots, T-1\}$. Having acted optimally until some intermediate time m, $0 < m < T$, which passes through some state x_m, we know that the trajectory over the interval $[m, T]$ is also optimal, where $[m, T] := \{m, m+1, \ldots, T], T > m$. This concept is visualized in Figure 2.1. With this intuition, we can write a recursive expression of the value function. Starting from some state x at time t,

$$V_t^*(x) = x^\intercal Q x + u^{*\intercal} R u^* + V_{t+1}^*(Ax + Bu^*), \tag{2.7}$$

where u^* is the OC input at the given timestep t. To compute the optimal trajectory, we have to iterate backward in time using this recursive relationship, starting from $t = T$. Suppose that we can write the following closed-form

quadratic expression for the value function

$$V_t^*(x) = x^\intercal P_t x, \tag{2.8}$$

where $P_t \in \mathbb{R}^{n\times n}$, $P_t = P_t^\intercal \succeq 0$ is some symmetric positive definite matrix. From here, we would like to utilize induction to identify the structure of P_t for all t. At time $t = T$, we already know that $V_T^*(x) = x^\intercal Q_f x$, which shows that $P_T = Q_f$. Now, starting from some arbitrary time t assuming knowledge of P_{t+1}, we can apply (2.7) and (2.8) to compute the value function backward in time.

$$\begin{aligned} V_t^*(x) &= x^\intercal Q x + \min_u u^\intercal R u + V_{t+1}^*(Ax + Bu), \\ &= x^\intercal Q x + \min_u u^\intercal R u + (Ax + Bu)^\intercal P_{t+1}(Ax + Bu). \end{aligned}$$

Taking the gradient of the expression inside the minimization with respect to u and setting to zero, we solve for the minimum.

$$\begin{aligned} \nabla_u \left(u^\intercal R u + (Ax + Bu)^\intercal P_{t+1}(Ax + Bu)\right) = 2u^\intercal R + 2(Ax + Bu)^\intercal P_{t+1} B = 0, \\ \Rightarrow u^* = -(R + B^\intercal P_{t+1} B)^{-1} B^\intercal P_{t+1} A x = -K_t x. \end{aligned}$$

Hence, we get the following feedback law.

$$K_t = (R + B^\intercal P_{t+1} B)^{-1} B^\intercal P_{t+1} A. \tag{2.9}$$

However, we must plug Equation (2.9) back into the previous expression for $V_t^*(\cdot)$ to compute P_t.

$$\begin{aligned} V_t^*(x) &= x^\intercal \left(Q + K_t^\intercal R K_t + (A - BK_t)^\intercal P_{t+1}(A - BK_t)\right) x, \\ &= x^\intercal P_t x. \end{aligned}$$

Simplifying the previous expression yields the analytical expression.

$$\begin{aligned} P_t &= Q + A^\intercal P_{t+1} A + K_t^\intercal (R + B^\intercal P_{t+1} B) K_t - A^\intercal P_{t+1} B K_t - K_t^\intercal B^\intercal P_{t+1} A, \\ &= Q + A^\intercal P_{t+1} A - A^\intercal P_{t+1} B (R + B^\intercal P_{t+1} B)^{-1} B^\intercal P_{t+1} A. \end{aligned} \tag{2.10}$$

This shows that (2.9) and (2.10) solve Problem 2 for the given time t, assuming we already knew P_{t+1}. Hence, by induction, with the base case that $P_T = Q_f$, we have that these recursions hold for all time t within the time horizon $[T]$.

In summary, we were able to derive the expression for the value function $V_t^*(\cdot)$ starting from time $t = T$ with the guess that it obeys a quadratic relationship with the current state. This provides the OC sequence $u_{0:T-1}^*$ for some initial condition x_0, thereby solving Problem 2. See [28, 29] for additional details.

2.1.2 Nonlinear Dynamics and Non-quadratic Costs

The above analysis was valid for the special case of linear dynamics expressed in (2.3). However, how would we apply this to the more general case of Equation (2.1)? Additionally, there was an inherent assumption that the costs had a quadratic structure in Equation (2.4). How do we handle these limitations? To explore this route, we must introduce the concept of linearization.

Consider that rather than driving the system toward equilibrium at all timesteps, we would instead like to follow some nominal trajectory $X = \{x_0, \dots, x_T\}$ with corresponding controls $U = \{u_0, \dots, u_{T-1}\}$. Note that the controls only go to timestep $T-1$ because the system has already been driven to its terminal state by time T using u_{T-1}. We can approximate the dynamics (2.1) using a first-order Taylor-series expansion. To apply this in discrete time, we consider a small perturbation in the current state δx_k and controls δu_k.

$$\begin{aligned}
x_{k+1} + \delta x_{k+1} &= f(x_k + \delta x_k, u_k + \delta u_k), \\
&\approx f(x_k, u_k) + \left.\frac{\partial f}{\partial x}\right|_{x_k, u_k} (x - x_k) + \left.\frac{\partial f}{\partial u}\right|_{x_k, u_k} (u - u_k), \\
\delta x_{k+1} &= A_k \delta x_k + B_k \delta u_k, \\
A_k &= \left.\frac{\partial f}{\partial x}\right|_{x_k, u_k} (x - x_k), \\
B_k &= \left.\frac{\partial f}{\partial u}\right|_{x_k, u_k} (u - u_k).
\end{aligned}$$

While the derivation in Section 2.1.1 was shown for time-invariant A and B,

it can also be shown to hold for the time-variant case above for all time k.

A similar action can be performed on the costs with a second order Taylor series to obtain a quadratic approximation to the costs about the current operating point (x_k, u_k). However, because the approximation is only valid in regions near the operating point, computed feedback gains might not hold the same validity as they did for the quadratic case. With this severe limitation, researchers modeling complex biological movement in animals and humans came up with a new approach that iteratively optimized for the locally-optimal feedback controller [30] known in the control community as Iterative-Linear Quadratic Regulator (iLQR). While we will not introduce the algorithm for the sake of brevity, the core idea is that we can run LQR over the linear-quadratic approximation to some nominal trajectory until some convergence criterion is met. For the sake of completeness, another algorithm that is also commonly referred to in these settings is known as Differential Dynamic Programming (DDP) [31]. The slight difference between DDP and iLQR is that DDP computes the full hessian of the dynamics, which are approximated in iLQR by only considering the first order terms for the sake of efficiency, which is usually an acceptable compromise.

As will be emphasized later, we note that while iLQR is a core component of the algorithm in this work, it is merely the solver of the optimization problem. Any other black-box nonlinear optimization algorithm could be used as a surrogate for iLQR such as Sequential Quadratic Programming or Interior Point Methods [32] by directly treating the states and controls as decision variables.

2.2 Multi-Agent Decision Making

Thus far, we have considered the OC problem of driving some dynamical system to some desired state, exerting minimal control effort across a finite time horizon. However, how do we model the interactions between multiple agents, each with their unique dynamics and objective functions? Dynamic game theory provides a powerful theoretical framework for this purpose. Chapter 4 will bring this theory into better focus for the purposes of this thesis. In this section, we seek to provide a brief introduction to game theory and intuition on the various notions of equilibria to model the outcomes of static

multi-agent problems.

2.2.1 Game Theory Overview

The formalisms underlying game theory are broadly concerned with how rationally behaving agents act. Especially when agents have competing cost functions, who wins? Before understanding the outcomes, we must first appreciate the game's structure. Consider a game, where the participants are referred to as agents. Agents can take actions within some set and have a payoff function that depends on the simultaneously taken actions of other agents. For consistency with the literature, we use the notion of a reward or payoff to be maximized, as opposed to a cost to be minimized in the controls community. As part of this broad class of games, we typically consider these dimensions.

- **Cooperative vs. Noncooperative**: Can agents form alliances?
- **Symmetric vs. Asymmetric**: Are the payoffs for a given strategy the same across all agents?
- **Zero-sum vs. General-sum**: What are the possibilities for the net payoff of all agents? Are there finite resources?
- **Static vs. Dynamic**: Do agents move once or sequentially over some period of time? Is there state?

It is usually assumed that agents are rational, meaning they act optimally or seek to enact the best response given their potentially imperfect information. Our work falls into the category of noncooperative asymmetric general-sum dynamic games. There is an additional sector of dynamic game theory known as differential game theory, in which agents' states are governed by a differential equation and is closely related to optimal control. Since we consider dynamics that have already been discretized, we reside in the class of dynamic games.

2.2.2 Illustrative Examples

To add color to these abstract concepts, imagine the "chicken" game, with two drivers on a collision course (some texts also refer to this as the hawk-

dove game [33]). If both swerve, neither agent wins. If either one swerves and the other goes straight, the one that went straight wins. If neither agent yields, both crash into each other, incurring a large penalty. This game can be visualized by the payoff matrix in Table 2.1.

Table 2.1: Example payoff matrix for the chicken game, in which two drivers heading for collision must decide whether to swerve or go straight. The rows correspond to a given action of player A and the columns correspond to a given action of player B. The two elements within each entry contain the payoff for a given combination of actions, with the left number being the payoff for player A and the right number being for player B.

	Swerve	Straight
Swerve	0, 0	-1, 1
Straight	1, -1	-1000, -1000

The idea is that since both drivers are prideful, neither wants to be the "chicken" and swerve out. As shown in Table 2.1, the cost of winning is $+1$, the cost of losing is -1, and the cost of crashing is -1000. We see the chicken model play out in the current geopolitical crisis in the Russo-Ukrainian War. Putin claims that "This is not a bluff. And those who try to blackmail us with nuclear weapons should know that the weathervane can turn and point towards them" [34,35]. It could be argued that Russia's indiscriminate attacks on cities are his way of doubling down in a game of chicken, and that the West will not retaliate with nuclear force.

Table 2.2: Example payoff matrix for the volunteering dilemma from some ego agent's perspective.

	Others Volunteer	Nobody Volunteers
I Volunteer	1	0
I Defect	10	-10

When applied to more than just two agents, this idea can be recast into what is known as the Volunteer's dilemma [36]. In this problem, one person can sacrifice for the good of the group, or choose to defer, hoping that someone else will contribute. We also see this in the bystander effect, where people are less likely to help others in need in the presence of other people [37,38]. Table 2.2 provides one potential payoff matrix for this setting.

As depicted in Figure 2.2, chess is yet another intriguing example of how the game-theoretical toolbelt can be applied to new problems. While it does

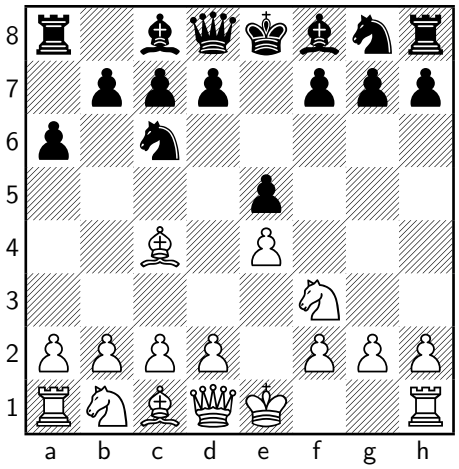

Figure 2.2: Chess is a canonical application of game-theory in which two agents engage in a perfect-knowledge non-cooperative zero-sum dynamic game. Strategies typically involve performing backward induction to simulate forward various possible outcomes given an opponent's best-response.

not have quite as clear equilibria as the matrix games, it is still a powerful game where agents iteratively execute their best-response strategies.

2.2.3 Nash Equilibrium in Static Games

While these examples allow us to model and simplify the mutual interactions of rational agents, we must introduce the concept of equilibrium to better understand the outcomes. Specifically, we highlight the notion of a static Nash Equilibrium for use in the dynamic case in Section 4.2.

Let N be the number of agents in the static game. The strategy profile for a given agent i is denoted by γ^i, where $\gamma^i \in \Gamma^i$ is some finite set of strategies for agent i. We use the notation γ^{-i} to refer to the strategies of all agents $j \in [N] = \{0, \dots, N-1\}, j \neq i$ excluding agent i. The payoff of agent i is written as $P^i(\cdot)$ and depends on the strategies of all agents.

Definition 3. *The set of strategies $\gamma^* = (\gamma^{i*}, \gamma^{-i*})$ constitutes a NE if for every agent $i \in [N]$ and every strategy $\gamma^i \in \Gamma^i$, we have the following relationship:*

$$P^i(\gamma^{i*}, \gamma^{-i*}) \geq P^i(\gamma^i, \gamma^{-i*}). \tag{2.11}$$

Informally, a set of strategies is considered a NE when no agent has an incentive to deviate from his or her strategy, given that he or she knows the

best response strategies of the other agents [39]. This can also be understood as all agents executing their respective best-response strategy to the possible strategies of the other agents.

We see this play out in a host of everyday interactions. Consider two friends who would both like to work out. Each must decide whether to go in the morning or at night. They do not get as good of a workout by themselves, so it benefits them to go together. They are also not morning people and hence would prefer to go at night. We see the payoff matrix for such a scenario depicted in Table 2.3. Here, the two pure-strategy NE, pure meaning agents use policies with deterministic actions, are to either both go in the morning or both go at night.

Table 2.3: One everyday life example of Nash Equilibrium, in which two friends who are both night owls are coordinating a time to go to the gym together.

	Morning	Night
Morning	2, 2	-1, 1
Night	1, -1	4, 4

Going back to the chicken game in Table 2.1, there are two pure-strategy Nash Equilibria — one swerves and the other goes straight or vice versa. Because agents are encouraged to choose opposite policies, this game is considered an anti-coordination game.

2.3 Multi-Agent Control Architectures

Now that we are equipped with a working understanding of how to drive a dynamical system to some goal and model the interactions of competing agents' objectives, we now consider the following question — what control architectures should we implement to facilitate desirable yet scalable outcomes in the multi-agent system (MAS)? This considers the structure of the communications between agents. It also specifies the hierarchy of control, i.e. do agents independently observe and process information? Or is some centralized node responsible for planning? These questions are paramount to the trade space of designing robotic systems in modern society that produce favorable outcomes using available information, yet are efficient and fault-tolerant. While there does not seem to be strong consensus in the robotics

community on the technical requirements for the various architectures, we utilize the definitions presented in [40, 41].

2.3.1 Centralized Architectures

This dimension of the MAS concerns where the information is processed and who makes decisions for each of the systems. More conventional systems were created to be centralized, where a single computational node processes information for all the agents [42]. A MAS where sensors are located in each of the subsystems, but the measurements are communicated to the central processor would still be considered centralized since the subsystems are relying on the central entity to carry out their function. Consider a team of workers tasked with cleaning a building. A centralized group would take directions from a leader that coordinates the actions and decides how it should be done.

2.3.2 Decentralized Architectures

In contrast, decentralized MAS's have the property that there is not a central node where resources reside, either computational or information related. Each node decides independently based on local information. Regarding the cleaning analogy, the decentralized team would instead operate more independently to carry out the task, with each person choosing to clean a given area based on their local observations.

Decentralized MAS's have been developed in more recent years to overcome the various limitations of the centralized architecture [43]. In the centralized case of the analogy, one might imagine how the leader might get flustered both deciding and communicating to a large team one-by-one what to do. Therefore, centralized architectures are not scalable in the number of agents. Whereas in the decentralized team, people are mostly self-sufficient in knowing how to do the job.

3 RELATED WORK

In this section, we review several recent developments to put this work into the context of the broader literature. Specifically, we focus on various aspects of the work, including game-theoretic navigation planning in Section 3.1, advantages of potential games in Section 3.2, and various recent approaches to distributed architectures in Section 3.3 that distinguish it from the state of the art.

3.1 Interactive Trajectory Planning

Reactive methods that utilize multi-modal probabilistic prediction models of agents are one of the popular approaches to interactive trajectory planning [44–46]. The downside with these approaches is that agents are not able to sufficiently influence each other, resulting in conservative interactions. Consequently, several recent methods have considered game-theoretic planning for interactive domains, which enables agents to influence one another and achieve joint prediction and planning. Several approaches that rely on DDP have been proposed [6,47,48] for finding NE of general dynamic games, with [6] demonstrated in real-time. The hierarchical method shown in [49] decomposes the problem into a strategic global planner and a tactical local planner, but it requires discretizing the state and action spaces. Dynamic programming approaches were further utilized in [8] and [50] to approximately find equilibria of interactions under uncertainty. In [51], equilibria of interactive dynamic games were sought under nonlinear state and input constraints.

Our work also correlates with various game-theoretic racing scenarios such as those studied in [5, 52–55]. Due to the proximal nature of these scenarios, however, problems typically only have two agents interacting at a time and, hence, do not explore the influence of the control architecture on the planning problem. To handle stochasticity, authors in [56] used a method resembling

Iterative Linear Quadratic Gaussian (iLQG) control to do planning in belief space. Implicit methods utilize some form of inverse reinforcement learning over trajectory datasets to achieve collision avoidance without directly imposing constraints on the structure of interactions [57–59], whereas we explicitly take advantage of the structure of certain multi-agent interactions to simplify the problem.

There has been a myriad of approaches exploring the various forms of game-theoretic equilibria among agents. Stackelberg Equilibria were initially used to model the outcomes [4, 52, 60]. However, it proved insufficient for general forms of interactions because it assumes a leader-follower structure, which does not apply to non-cooperative games with more than two agents. In navigating through a crowd of people, for example, there is not necessarily a designated leader. Some people might take more initiative than others in walking on a certain side or walking more quickly to get others to move, but this is not a general solution nor one to be deployed in robots. Due to these challenges, others considered NE to capture the interactions among more than two agents.

3.2 Potential Games

When it comes to computing equilibria, potential games are a class of games for which equilibria can be found efficiently and reliably. While much of the literature on potential games is oriented toward the static case, there have been several recent advancements in dynamic potential games. Following the pioneering works in [61] and [62], there were initially two primary methods of solving dynamic games: Euler-Lagrange and Pontryagin's Maximum methods. More recently, a Hamiltonian potential function was explored in [63] for open-loop games with continuous time models. While [64] was primarily focused on communications applications, they demonstrated successful utilization of the simplified problem structure offered by potential games. Similar to our work, [65] posed the idea of connecting the open-loop NE of a dynamical game with the solutions to an OC problem under certain conditions. Recently, [9] and [66] demonstrated that dynamic potential games can be leveraged for trajectory planning in interactive robotics, but each used a centralized construction.

3.3 Decentralized Architectures in Multi-Robot Teams

The control structures used in a given multi-agent planning problem have strong implications for its applicability in the real world. We examine problems in which distributed and decentralized techniques have been applied to various dynamic games in robotics. Yang et al. [67] explored the problem of distributed role assignment cast as a dynamic game. In the vein of multi-agent sensor fusion over a finite communication bandwidth, [68] proposes an approach to enable a fleet of UAV's to upload sensor data into the cloud using a potential game formulation. However, this potential game is done over a common convex function and is not relevant for problems with disparate objectives. Formation control is another potential problem domain for decentralized architectures, in which a set of leaders each guide a group of followers in some formation, typically for coverage control or tracking purposes. Hu et al. [69] introduces a hierarchical cluster formation technique that utilizes a game-theoretic rule to split up the leaders into a Nash-stable partition based on local information. The approach in [70] employs a fully decentralized communication-free method, but makes assumptions about the cost structures of the problem, limiting its applicability to formation control.

4 PROBLEM FORMULATION

4.1 Planning Problem Notation

Consider an interactive trajectory planning problem with N agents. Let $[N] \equiv \{1, \ldots, N\}$ be the set of agents' indices. We refer to the state of the ith agent at time k as $x_k^i \in \mathbb{R}^{n_i}$, where n_i is the dimension of the state vector of agent $i \in [N]$. Agent i's corresponding control input is denoted as $u_k^i \in \mathbb{R}^{m_i}$, where m_i is the dimension of the control space of agent $i \in [N]$. Concatenating the states and inputs of all agents, the full state vector of the system at time k is given by $x_k = (x_k^1, x_k^2, \ldots, x_k^N) \in \mathbb{R}^n$, where $\sum_{i=1}^N n_i \equiv n$. The concatenated set of control inputs of all agents at time k is similarly denoted by $u_k = (u_k^1, u_k^2, \ldots, u_k^N) \in \mathbb{R}^m$, where $\sum_{i=1}^N m_i \equiv m$. Generally, subscripts are used to denote the time index, whereas superscripts denote the agent index. The states for agent i across an entire horizon T is notated as $x^i = \{x_0^i, \ldots, x_T^i\} \in \mathbb{R}^{n_i \times T}$. Similarly, for the controls, let $u^i = \{u_0^i, \ldots, u_T^i\} \in \mathbb{R}^{m_i \times T}$ be agent i's control inputs across the horizon. We drop both subscript and superscript to refer to all agents over an entire horizon for states $x \in \mathbb{R}^{n \times T}$ and controls $u \in \mathbb{R}^{m \times T}$. Lastly, we use capitalization to differentiate the predicted states X from the actual states x and the same for the controls U from u, respectively.

We consider separable agents' dynamics defined by $f^i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \mapsto \mathbb{R}^{n_i}$, i.e. we assume that for every agent $i \in [N]$, we have:

$$x_{k+1}^i = f^i(x_k^i, u_k^i). \tag{4.1}$$

We denote the strategy space of each agent $i \in [N]$ as Γ^i. Let the strategy $\gamma^i \in \Gamma^i$ of agent i be given by $\gamma^i : \mathbb{R}^{n_i} \times \{0, 1, \ldots, T-1\} \mapsto \mathbb{R}^{m_i}$ which determines the actions of agent i at all time instants. We consider open-loop strategies that are only a function of the system's initial state and the time step. Therefore, we have $\gamma^i(x_0, k) := u_k^i$. Hence, for simplicity, we use

strategies and actions interchangeably here out.

4.2 Nash Equilibria in Dynamic Games

While Section 2.2.3 covered NE for the static case, we cover the dynamic case in this section. We assume that each agent i is minimizing some cost $J^i(\cdot)$ over the time horizon T:

$$J^i(x_0, \gamma) = S^i(x_T, T) + \sum_{k=0}^{T-1} L^i(x_k, \gamma(x_0, k), k), \tag{4.2}$$

where $S^i(\cdot)$ is the terminal cost of agent i and $L^i(\cdot)$ is the running cost of agent i. Note that the cost perceived by an agent i may depend on the states and actions of all the other agents. As a result, generally, it is not possible for all agents to optimize their costs simultaneously, and we need to model the outcome of the interactions between the agents as equilibria of the underlying dynamic game. We denote our dynamic games by a compact notation, $G^T_{x_0} := (T, \{\gamma^i\}_{i=1}^N, \{J^i\}_{i=1}^N, \{f^i\}_{i=1}^N)$, which denotes the dynamic game that arises from interactions of the agents over a horizon T starting from the initial condition x_0.

Let $\gamma^{-i} = (\gamma^1, \ldots, \gamma^{i-1}, \gamma^{i+1}, \ldots, \gamma^N)$ denote the strategies of all other agents except i. We use similar notation to express the states and controls of all other agents except i for a given time step k as $x_k^{-i} \in \mathbb{R}^{n-n_i}$ and $u_k^{-i} \in \mathbb{R}^{m-m_i}$, respectively. Then, the Nash equilibria of our dynamic game are defined as [71]:

Definition 4. *For a given game, $G^T_{x_0} := (T, \{\gamma^i\}_{i=1}^N, \{J^i\}_{i=1}^N, \{f^i\}_{i=1}^N)$, a set of strategies γ^* are open-loop Nash equilibrium strategies if for every agent $i \in [N]$ and every strategy γ^i, we have*

$$J^i(x_0, \gamma^*) \leq J^i(x_0, \gamma^i, \gamma^{-i*}). \tag{4.3}$$

Definition 4 implies that at equilibrium, no agent has any incentive for changing its strategy and actions once it fixes the strategies and actions of all the other agents to be their equilibrium strategies γ^{-i*}. However, finding Nash equilibria is challenging, as solving (4.3) requires solving a set of coupled optimal control problems. Finding equilibria has proven difficult

even in two-player settings for discrete state and action spaces [72–74]. Most recent methods for finding Nash equilibria of dynamic games that arise in robotics [6, 7, 9] solve the game in a centralized fashion, i.e. each agent must maintain a full copy of all the other agents. As a result, the existing methods and solvers are not scalable and become intractable beyond three agents with simple dynamics. In this work, we seek to remedy this and develop a distributed trajectory optimization algorithm for finding Nash equilibria of dynamic games underlying multi-agent interactions.

5 PRIOR RESULTS

In this chapter, we review some of the results from our previous work that we will utilize in the current work. Specifically, we review our previous results from [9] that allow one to bypass solving (4.3) for interactions that are dynamic potential games [75].

5.1 Reduction to Single-Objective Optimization

We have the following result from [9].

Theorem 1. *For a given dynamic game* $G_{x_0}^T = (T, \{\gamma^i\}_{i=1}^N, \{J^i\}_{i=1}^N, \{f^i\}_{i=1}^N)$, *if for each agent* $i \in [N]$, *the running and terminal costs have the following structure*

$$L^i(x_k, u_k) = p(x_k, u_k) + c^i(x_k^{-i}, u_k^{-i}), \ \forall k \tag{5.1}$$

and

$$S^i(x_T) = \bar{s}(x_T) + s^i(x_T^{-i}), \tag{5.2}$$

then, the dynamic game $G_{x_0}^T$ *is a dynamic potential game. and open-loop Nash equilibria* $u^* = (u^{1*}, \dots, u^{N*})$ *can be found by solving the following optimal control problem*

$$\begin{aligned} &\min_u \sum_{k=0}^{T-1} p(x_k, u_k) + \bar{s}(x_T), \\ &s.t. \ x_{k+1}^i = f^i(x_k^i, u_k^i). \end{aligned} \tag{5.3}$$

Conditions (5.1) and (5.2) imply that one can decompose both the running costs $L^i(\cdot)$ and terminal costs $S^i(\cdot)$ into potential functions $p(\cdot)$ and $\bar{s}(\cdot)$ which can depend on the full state and control vector of the agents, and the cost terms $c^i(\cdot)$ and $s^i(\cdot)$ that have no dependence on the state and control input of agent i.

For the remainder of the section, we make this result more concrete via a navigation example, which serves as our running example throughout the paper.

5.2 Example Cost Decomposition for Multi-Agent Navigation

Consider a multi-agent navigation setup where each agent $i \in [N]$ must reach a goal state $\bar{x}^i$. We assume that each agent's running cost function is composed of a tracking cost and a control penalty term defined as:

$$\begin{aligned} C^i_{tr}(x^i_k, u^i_k) &= (x^i_k - \bar{x}^i)^\intercal Q^i (x^i_k - \bar{x}^i) \\ &\quad + (u^i_k - \bar{u}^i)^\intercal R^i (u^i_k - \bar{u}^i), \end{aligned} \tag{5.4}$$

$$C^i_{tr,T}(x_T) = (x^i_T - \bar{x}^i)^\intercal Q^i_f (x^i_T - \bar{x}^i), \tag{5.5}$$

where $Q^i, Q^i_f \in \mathbb{R}^{n_i \times n_i}$, $Q^i, Q^i_f \succeq 0$ and $R^i \in \mathbb{R}^{m_i \times m_i}$, $R^i \succ 0$ are all symmetric matrices, and $\bar{u}^i$ is a reference control input. Moreover, assume that agents' decisions are coupled through cost terms such as the collision avoidance terms between any pair of agents $i \neq j, i, j \in [N]$ defined as:

$$C^{ij}_{ca}(x^i_k, x^j_k) = \alpha^{ij}(d(x^i_k, x^j_k)), \tag{5.6}$$

where $d(x^i_k, x^j_k) \equiv d^{ij}_k$ is the distance between agents i and j at time k.

For each agent $i \in [N]$, the instantaneous running cost $L^i(\cdot)$ can then be defined as:

$$L^i(x_k, u^i_k) = C^i_{tr}(x^i_k, u^i_k) + \sum_{i \neq j}^{N} C^{ij}_{ca}(x^i_k, x^j_k). \tag{5.7}$$

It was shown in [9] that under the assumption that agents induce coupling costs between each other symmetrically, i.e. $C^{ij}_{ca}(x^i_k, x^j_k) = C^{ji}_{ca}(x^j_k, x^i_k), \forall i \neq j \in [N]$, the dynamic game is a dynamic potential game with the potential function:

$$p(x_k, u_k) = \sum_{i=1}^{N} C^i_{tr}(x^i_k, u^i_k) + \sum_{i=1, i<j}^{N} C^{ij}_{ca}(x^i_k, x^j_k). \tag{5.8}$$

This implies that Nash Equilibria (4.3) can be found by solving (5.3), which is a single optimal control problem. Prior work [9] used iLQR [30, 76] for

minimizing (5.8) in a MPC setting. Explicitly, it solves (5.3) for all agents iteratively as covered in Algorithm 1. We refer the reader to the original work [9] for additional details.

Algorithm 1 Potential-iLQR

Inputs
dynamics $\{f^i\}_{i=1}^N$, potentials p and $\bar{s}$, initial state x_0
Outputs
control inputs u
Initialization
$u \leftarrow 0^{N \times m}$

1. **while** not converged **do**
2. Rollout from x_0 with u s.t. (4.1) to compute X
3. Linearize dynamics, quadraticize costs about X, u
4. Solve the Riccati recursion to update controls u
5. **end while**

6 DISTRIBUTED TRAJECTORY OPTIMIZATION

Prior work has proposed to solve (5.8) by requiring every agent to maintain a copy of all the other agents, which restricts the scalability of the method. Our key insight is that we can solve the optimal control problem (5.8) more efficiently using ideas from distributed MPC [77–80]. Specifically, we can break up (5.8) into smaller subproblems more relevant to each agent.

6.1 Interaction Graph

To formalize distributed interactive trajectory planning, we borrow ideas from [79] and [81] and introduce the concept of an interaction graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ comprising nodes $\mathcal{V} = [N]$ for each agent i and edges $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$, where each edge $(i, j) \in \mathcal{V}$, indicates a coupling between two agents in their costs. For our purposes, these edges are connected if agents ever get within a proximity distance d_{prox} of each other throughout the predicted trajectories. Formally, let $\{d_k^{ij}\}_{k=0}^{T-1} \in \mathbb{R}^T$ be the predicted distances between agents i and j over the horizon T. We create a bidirectional edge (i, j) if:

$$\exists k, 0 \leq k < T, \text{ s.t. } d_k^{ij} < \alpha d_{\text{prox}}, \tag{6.1}$$

where $\alpha \in \mathbb{R}, \alpha \geq 1$ is some aggressiveness parameter on how finely to split up the graph. For sufficiently large α, the interaction graph is complete, i.e., all agents are connected with one another. Whereas for sufficiently small α, $\mathcal{E} = \emptyset$. Additionally, for each agent $i \in [N]$, its set of neighbors is defined as $\mathcal{N}^i$, such that $\mathcal{N}^i = \{j : (i, j) \in \mathcal{E}\}$ (see Figure 6.1).

The introduction of the interaction graph is motivated by human swarm motion. As referenced in [82], the key to collective motion is a pedestrian's zone of influence. Intuitively, humans need not worry about others outside their vicinity. Similarly, we argue that in multi-agent interactions, agents need only pay attention to their zone of influence in the navigation problem.

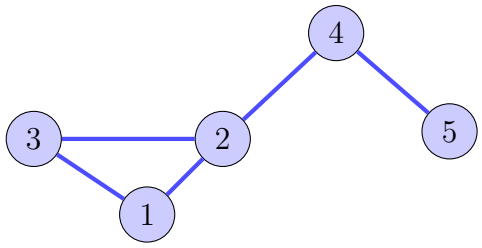

Figure 6.1: Schematic Example of an interaction graph for one-time step, where $\mathcal{N}^1 = \{2, 3\}$, $\mathcal{N}^2 = \{1, 3, 4\}$, $\mathcal{N}^3 = \{1, 2\}$, $\mathcal{N}^4 = \{2, 5\}$, and $\mathcal{N}^5 = \{4\}$.

6.2 Partitioning the Centralized Problem

We require each agent i to solve its own subproblem, which we denote by $\mathcal{P}^i$, associated with minimizing the potential function that comprises only itself and its neighbors $\tilde{p}^i$, i.e., the subset of the graph that it shares edges with $\mathcal{N}^i$. Let the full subproblem consist of agents in the set $\widetilde{\mathcal{N}}^i = \mathcal{N}^i \cup i$. The state vector of this subproblem at time k is then denoted by $\tilde{x}_k^i = \{x_k^j\}_{j \in \widetilde{\mathcal{N}}^i}$ with the corresponding control input $\tilde{u}_k^i = \{u_k^j\}_{j \in \widetilde{\mathcal{N}}^i}$. We propose to divide the centralized problem (5.3) into a set of local subproblems, where each agent i solves its subproblem $\mathcal{P}^i$ defined as

$$\begin{aligned} \min_{\tilde{u}^i} \; & \sum_{k=0}^{T-1} \tilde{p}^i(\tilde{x}_k^i, \tilde{u}_k^i) + \tilde{\tilde{s}}^i(\tilde{x}_T^i) \\ \text{s.t. } & x_{k+1}^i = f^i(x_k^i, u_k^i), \end{aligned} \tag{6.2}$$

where $\tilde{\tilde{s}}^i(\cdot) : \mathbb{R}^{\tilde{n}_i} \mapsto \mathbb{R}$ for $\tilde{n}_i = \sum_{i \in \tilde{N}_i} n_i$, is a terminal cost for the local problem. Therefore, (6.2) is the local analog to (5.3), with local potential functions $\tilde{p}^i$ and $\tilde{\tilde{s}}^i$ that comprise local costs. In the specific case of multi-agent navigation, one such potential function could take the following form:

$$\tilde{p}^i(x_k, u_k) = \sum_{j \in \widetilde{\mathcal{N}}^i} C_{tr}^j(x_k^j, u_k^j) + \sum_{j \in \mathcal{N}^i} C_{ca}^{ij}(x_k^i, x_k^j). \tag{6.3}$$

Hence, (6.3) is then a subset of (5.8) that takes advantage of the sparsity of the interaction graph.

6.3 Distributed Potential-iLQR

Let the trajectory of the full system predicted by agent i at time k be X_k^i, where $X_0^i \equiv x_0$, such that over the full horizon:

$$X^i = \{x_0, X_1^i, \ldots, X_{T-1}^i\}, \tag{6.4}$$

where $X^i \in \mathbb{R}^{n \times T}$ is the predicted trajectory according to agent i across the horizon.

We are now ready to define our distributed trajectory planner in Algorithm 2 — Distributed Potential-iLQR (DP-iLQR).

Algorithm 2 DP-iLQR

Inputs
predicted trajectories X^i (6.4), system dynamics $\{f^j\}_{j=1}^N$ (4.1), costs (5.4)–(5.6)
Outputs
control inputs u^i for agent i

1. $\mathcal{N}^i, \tilde{x}_0^i \leftarrow$ define Interaction Graph(X^i) (6.1)
2. $\tilde{p}^i, \tilde{\bar{s}}^i \leftarrow$ define local potential functions$(\mathcal{N}^i)$
3. $U \leftarrow$ Potential-iLQR$(\{f^i\}_{i \in \tilde{\mathcal{N}}_i}, \tilde{p}^i, \tilde{\bar{s}}^i, \tilde{x}_0^i)$ (Alg. 1)
4. $u^i \leftarrow$ extract Agent(U, i)

In applying Algorithm 2 in a receding horizon fashion, we continually compute the interaction graph at each step in line 1 and compose local potential functions in line 2. Agents then solve their local subproblem (6.2) in line 3. Agent i's solution u^i is then pulled out from U in line 4 and executed. This is then communicated with the rest of the system to define the subsequent graph at the next step.

We can utilize any single-agent trajectory optimization algorithm to solve each local subproblem by each agent i. We choose iLQR for solving each subproblem $\mathcal{P}^i$ due to its widespread success across many robotics domains. The broader question of how these sub-problems interact with each other remains an open question, which we will explore empirically in Chapter 7.

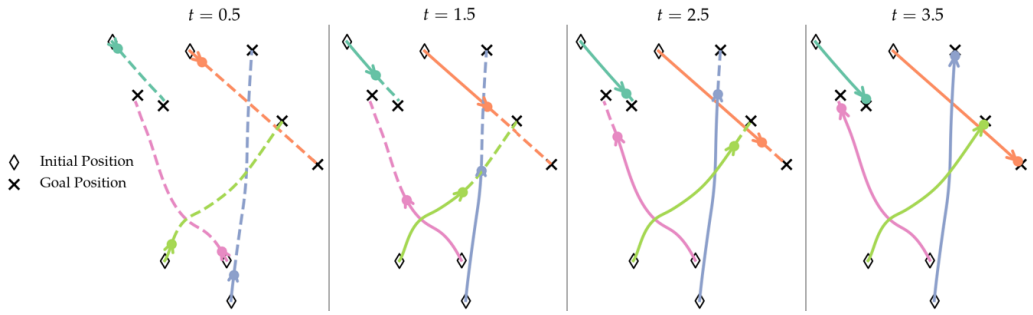

Figure 6.2: Example trajectories of DP-iLQR (Algorithm 2) for unicycle dynamics for varying numbers of agents from random initial conditions and goal positions. As time progresses from the left to right, the algorithm is able to guide agents through complex interactions.

7 RESULTS

We first evaluate the performance of DP-iLQR in a custom-built simulation environment across several dynamical models in a Monte-Carlo fashion in Section 7.1. We subsequently examine performance in the real-world on multiple quadcopters navigating around each other in Section 7.2.

7.1 Simulation Studies

We would like to show that DP-iLQR can handle larger-sized problems than Potential-iLQR. To accomplish this, we vary the number of agents and compare the centralized planner with the distributed planner at a given initial condition. We considered a multi-agent navigation setup with several different dynamics models, including 2D double integrator, 2D unicycle, and 3D quadcopter dynamics. We model our quadcopter as a six-dimensional model, specifically:

$$\begin{aligned} \dot{p}_x &= v_x, \quad \dot{v}_x = g\tan(\theta), \\ \dot{p}_y &= v_y, \quad \dot{v}_y = -g\tan(\phi), \\ \dot{p}_z &= v_z, \quad \dot{v}_z = \tau - g, \end{aligned} \tag{7.1}$$

where θ is the pitch, ϕ is the roll, τ is the combined force of the motors in the z-direction, p_x, p_y, and p_z are the 3D position, v_x, v_y, and v_z are the 3D velocities, and g is the acceleration due to gravity.

The specific form of collision avoidance cost that we utilized is the following:

$$\alpha^{ij}(d_k^{ij}) = \begin{cases} \beta(d_k^{ij} - d_{\text{prox}})^2 & d_k^{ij} < d_{\text{prox}} \\ 0 & \text{otherwise}, \end{cases} \tag{7.2}$$

where $\beta \in \mathbb{R}$ is some weighting parameter and d_{prox} is a threshold distance where agents begin incurring penalties for being too close to each other.

We implemented Algorithm 2 in a combination of Python and C++[1]. We generated 30 random initial conditions. For a given initial condition, we fixed the number of agents and computed the interactive trajectories of the agents using both Potential-iLQR and DP-iLQR. We implemented our algorithm in a receding horizon controller to fully exercise Algorithm 2. We ran the simulations with a horizon length of 40, a time interval of 0.1 seconds, and a collision radius d_{prox} of 0.5 meters. As shown in Figure 6.2, DP-iLQR generates intuitive collision-free trajectories under feasible initial conditions.

We conducted two primary analyses to quantify the differences in the performance between the centralized and distributed implementations: limited and unlimited solve time. The solve time refers to the time it takes at a given operating point to entirely execute the control algorithm, which corresponds to the time the algorithm would need to provide a new control input in real-time use cases.

7.1.1 Analysis 1 - Unlimited Solve Time

For the case of unlimited solve time, we enforce no computational time limit on the solver for converging to a solution at each receding horizon. This is an unrealistic assumption for practical implementations, but it enables us to compare the solve time of the two methods. In Figure 7.1, we compare the individual subproblem solve times throughout the simulated trajectories. As the number of agents increases, the problem gets increasingly congested; hence, we see each agent spending more time computing its subsequent control input. This is to be expected as the size of the state space grows and the cost surface becomes more challenging to navigate. The advantage of DP-iLQR in terms of the solve time is most prominent for simpler dynamical models like the double integrator. Regardless, it demonstrates a consistent decrease in solve time that widens with more agents.

7.1.2 Analysis 2 - Real-Time Constraint

When evaluating performance with more realistic timing constraints, we only permit the solver to iterate until it exceeds some time-based threshold. This

[1] `dp-ilqr` GitHub Repository: https://github.com/labicon/dp-ilqr

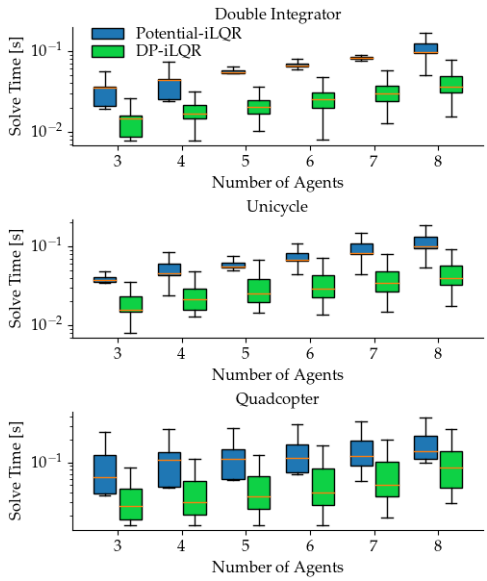

Figure 7.1: Average solve times without real-time constraints. DP-iLQR yields a consistently lower solve time than the centralized solver across various dynamics models and numbers of agents.

serves as a 'best-effort' solution that will be more applicable to a hardware implementation where the MPC planner requires a solution by the end of each time step. In this case, we compare the quality of the trajectories under real-time constraints. In particular, we measure the distance to goal at the end of the time horizon as a measure of solution quality. Figure 7.2 demonstrates the distance to the goal position under the two methods for various numbers of agents.

As the number of agents exceeds six, we see in Figure 7.2 that the variance in the centralized solver increases significantly as it starts getting overwhelmed handling the multitude of agent interactions. In contrast, the distributed solver shows much more consistent convergence statistics and even outperforms the centralized solver in most cases.

7.2 Hardware Experiments

To evaluate the real-time capabilities of our algorithm, we conducted navigation experiments involving multiple quadcopters. We ran the DP-iLQR algorithm on five Crazyflie 2.1 quadcopters, where each quadcopter navigates to its designated final position. VICON motion capture system was used to provide position and velocity feedback to all the quadcopters. The position

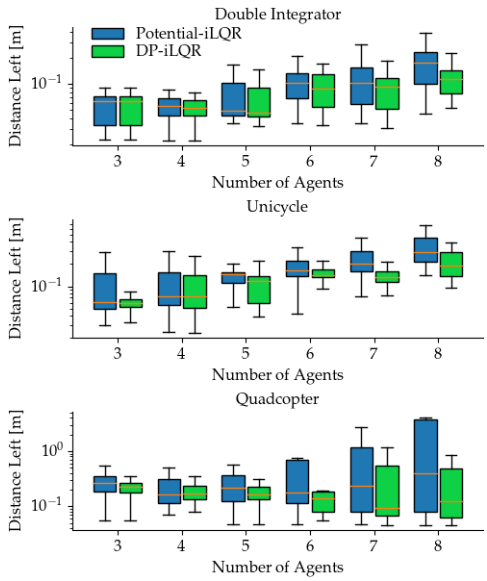

Figure 7.2: Average distance left to the goal position with a real-time constraint. While the difference between the two solvers is smaller at lower scales, Potential-iLQR is unable to maintain the quality of trajectories at the scales of six or more agents.

update computed by DP-iLQR was sent to each quadcopter online via the Crazyswarm API [83]. The algorithm was executed offboard on a laptop with an 8-core AMD processor with 32 GB of RAM. We demonstrated that DP-iLQR provides a more stable and intuitive trajectory than the Potential iLQR [9]. A visualization demonstrating the near-real-time trajectories generated by DP-iLQR is shown in Figure 7.3.

While the experiments performed served as a proof-of-concept for demonstrating the real-time capabilities of the algorithm, the experiments were limited in the links of the radio channels used to send instructions to the drones. With stronger link quality, we would be able to simulate with quantities of drones greater than five. Even more pressing to the processing speed of the algorithm, having on-board computations or at least dedicated hardware for each agent could have the potential to greatly speed up processing for sufficiently strong processors. For the sake of time and simplicity, we solved all individual agent subproblems on one laptop sequentially. This is still a distributed manner of solving the problem considering each subproblem was solved independently, with communication between agents happening only between solving from a given condition.

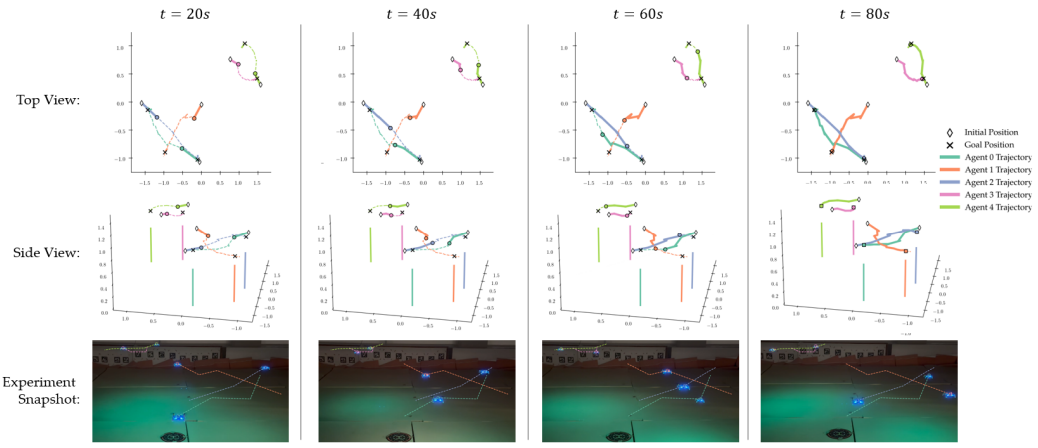

Figure 7.3: Hardware experiment demonstrated on Crazyflie 2.1 using the Crazyswarm library [83]. In this scenario, five drones navigate around each other in two separate intersections. As time evolves from left to right, the drones are able to safely navigate around each other. The first row depicts the birds-eye view, the middle row depicts the 3D rendered side view, and the lower row shows live pictures of the experiment. The vertical lines correspond to each of the $\times$ markers and denote the (x, y) of the intended goal position colored by agent.

8 CONCLUSION

Summary. We introduced a scalable and distributed algorithm for interactive trajectory planning in multi-agent interactions. We considered interactions in a game-theoretic setting and utilized the connection between multi-agent interactions and potential games to reduce the problem of finding equilibria of the game to that of minimizing a single potential function. Then, we used distributed trajectory optimization algorithms to minimize the resulting potential function. This results in a scalable and efficient trajectory planner for finding equilibria of interactive dynamic games. We compared our method with the state-of-the-art in several simulation studies and hardware experiments involving multiple quadcopters.

Limitations and Future Work. In our simulations and experiments, we found that the consistency of the trajectory that results from DP-iLQR is sensitive to the choice of cost parameters. More work would be required to investigate the feasibility of designing a self-tuning algorithm to determine optimal weights automatically. We would like to further examine both the theoretical guarantees and optimality gap of our proposed trajectory planner, as well as the application and impacts of distributed optimization techniques such as ADMM on distributed interactive trajectory planning.

REFERENCES

[1] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, and R. Wood, "The grand challenges of science robotics," *Science Robotics*, vol. 3, no. 14, p. eaar7650, 2018. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.aar7650

[2] M. Sabbagh and L. C. Bowman, "Theory of mind," *Stevens' handbook of experimental psychology and cognitive neuroscience*, vol. 4, pp. 249–288, 2018.

[3] P. Kinderman, R. Dunbar, and R. P. Bentall, "Theory-of-mind deficits and causal attributions," *British journal of Psychology*, vol. 89, no. 2, pp. 191–204, 1998.

[4] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions." in *Robotics: Science and systems*, vol. 2. Ann Arbor, MI, USA, 2016, pp. 1–9.

[5] 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.

[6] 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," 2019. [Online]. Available: https://arxiv.org/abs/1909.04694

[7] F. Laine, D. Fridovich-Keil, C.-Y. Chiu, and C. Tomlin, "The computation of approximate generalized feedback nash equilibria," *arXiv preprint arXiv:2101.02900*, 2021.

[8] M. Wang, N. Mehr, A. Gaidon, and M. Schwager, "Game-theoretic planning for risk-aware interactive agents," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6998–7005.

[9] T. Kavuncu, A. Yaraneri, and N. Mehr, "Potential ilqr: A potential-minimizing controller for planning multi-agent interactive trajectories," 2021. [Online]. Available: https://arxiv.org/abs/2107.04926

[10] M. Smyrnakis and S. M. Veres, "Coordination of control in robot teams using game-theoretic learning," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1194–1202, 2014.

[11] J. Banfi, N. Basilico, and S. Carpin, "Optimal redeployment of multi-robot teams for communication maintenance," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3757–3764.

[12] Z. Williams, J. Chen, and N. Mehr, "Distributed potential ilqr: Scalable game-theoretic trajectory planning for multi-agent interactions," in *2023 International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, awaiting Publication.

[13] H. Purnawan, Mardlijah, and E. B. Purwanto, "Design of linear quadratic regulator (lqr) control system for flight stability of lsu-05," *Journal of Physics: Conference Series*, vol. 890, no. 1, p. 012056, sep 2017. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/890/1/012056

[14] J. Willis, J. Johnson, and R. W. Beard, "State-dependent lqr control for a tilt-rotor uav," in *2020 American Control Conference (ACC)*, 2020, pp. 4175–4181.

[15] C. Hajiyev and S. Y. Vural, "Lqr controller with kalman estimator applied to uav longitudinal dynamics," *Positioning*, 2013.

[16] Y. Yang, "Analytic lqr design for spacecraft control system based on quaternion model," *Journal of aerospace engineering*, vol. 25, no. 3, pp. 448–453, 2012.

[17] M. Tillerson, G. Inalhan, and J. P. How, "Co-ordination and control of distributed spacecraft systems using convex optimization techniques," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 12, no. 2-3, pp. 207–242, 2002.

[18] Q. Hu, "Robust adaptive sliding mode attitude maneuvering and vibration damping of three-axis-stabilized flexible spacecraft with actuator saturation limits," *Nonlinear Dynamics*, vol. 55, pp. 301–321, 2009.

[19] P. Rao, M. Crow, and Z. Yang, "Statcom control for power system voltage control applications," *IEEE Transactions on Power Delivery*, vol. 15, no. 4, pp. 1311–1317, 2000.

[20] H. Ko, K. Lee, and H. Kim, "An intelligent based lqr controller design to power system stabilization," *Electric Power Systems Research*, vol. 71, no. 1, pp. 1–9, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378779603003134

[21] U. Markovic, Z. Chu, P. Aristidou, and G. Hug, "Lqr-based adaptive virtual synchronous machine for power systems with high inverter penetration," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 3, pp. 1501–1512, 2018.

[22] M. Aldeen and F. Crusca, "Multimachine power system stabiliser design based on new lqr approach," *IEE Proceedings-Generation, Transmission and Distribution*, vol. 142, no. 5, pp. 494–502, 1995.

[23] M. C. Priess, R. Conway, J. Choi, J. M. Popovich, and C. Radcliffe, "Solutions to the inverse lqr problem with application to biological systems analysis," *IEEE Transactions on control systems technology*, vol. 23, no. 2, pp. 770–777, 2014.

[24] J. He, M. G. Maltenfort, Q. Wang, and T. M. Hamm, "Learning from biological systems: Modeling neural control," *IEEE Control Systems Magazine*, vol. 21, no. 4, pp. 55–69, 2001.

[25] K. Uygun, H. W. Matthew, and Y. Huang, "Dfba-lqr: An optimal control approach to flux balance analysis," *Industrial & engineering chemistry research*, vol. 45, no. 25, pp. 8554–8564, 2006.

[26] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[27] N. Thacker and A. Lacey, "Tutorial: The kalman filter," *Imaging Science and Biomedical Engineering Division, Medical School, University of Manchester*, vol. 61, 1998.

[28] R. Tedrake, *Underactuated Robotics*, 2023. [Online]. Available: https://underactuated.csail.mit.edu

[29] T. Basar, S. Meyn, and W. R. Perkins, "Lecture notes on control system theory and design," *arXiv preprint arXiv:2007.01367*, 2020.

[30] W. Li and E. Todorov, in *Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems.*, vol. 1, 01 2004, pp. 222–229.

[31] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[32] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[33] R. Sugden et al., *The economics of rights, co-operation and welfare*. Springer, 2004.

[34] G. Faulconbridge and A. Richardson, "Factbox: Has putin threatened to use nuclear weapons?" *Europe*, Oct. 2022. [Online]. Available: https://www.reuters.com/world/europe/has-putin-threatened-use-nuclear-weapons-2022-10-27/

[35] O. of the Kremlin, "Message from the president of the russian federation," *President of Russia*, Sep. 2022. [Online]. Available: http://www.kremlin.ru/events/president/news/69390

[36] "Volunteer's Dilemma," may 4 2022, [Online; accessed 2023-07-06].

[37] A. Diekmann, "Volunteer's dilemma," *The Journal of Conflict Resolution*, vol. 29, no. 4, pp. 605–610, 1985. [Online]. Available: http://www.jstor.org/stable/174243

[38] W. Przepiorka and A. Diekmann, "Heterogeneous groups overcome the diffusion of responsibility problem in social norm enforcement," *PloS one*, vol. 13, no. 11, p. e0208129, 2018.

[39] R. Gibbons, "A primer in game theory," 1992.

[40] A. C. Jiménez, V. García-Díaz, and S. Bolaños, "A decentralized framework for multi-agent robotic systems," *Sensors*, vol. 18, no. 2, p. 417, 2018.

[41] A. Jamshidpey, M. Wahby, M. Heinrich, M. Allwright, W. Zhu, and M. Dorigo, "Centralization vs. decentralization in multi-robot coverage: Ground robots under uav supervision," 2021.

[42] B. Khoshnevis and G. Bekey, "Centralized sensing and control of multiple mobile robots," *Computers & industrial engineering*, vol. 35, no. 3-4, pp. 503–506, 1998.

[43] H. Asama, "Operation of cooperative multiple robots using communication in a decentralized robotic system," in *Proceedings of PerAc'94. From Perception to Action.* IEEE, 1994, pp. 36–46.

[44] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," *CoRR*, vol. abs/1710.09483, 2017. [Online]. Available: http://arxiv.org/abs/1710.09483

[45] H. Nishimura, B. Ivanovic, A. Gaidon, M. Pavone, and M. Schwager, "Risk-sensitive sequential action control with multi-modal human trajectory forecasting for safe crowd-robot interaction," *CoRR*, vol. abs/2009.05702, 2020. [Online]. Available: https://arxiv.org/abs/2009.05702

[46] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 454–460.

[47] W. Sun, E. A. Theodorou, and P. Tsiotras, "Game theoretic continuous time differential dynamic programming," in *2015 American control conference (ACC)*. IEEE, 2015, pp. 5593–5598.

[48] B. Di and A. Lamperski, "Differential dynamic programming for nonlinear dynamic games," 2018. [Online]. Available: https://arxiv.org/abs/1809.08302

[49] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9590–9596.

[50] N. Mehr, M. Wang, M. Bhatt, and M. Schwager, "Maximum-entropy multi-agent dynamic games: Forward and inverse solutions," *IEEE Transactions on Robotics*, 2023.

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

[52] A. Liniger and J. Lygeros, "A noncooperative game approach to autonomous racing," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 884–897, 2019.

[53] R. Spica, E. Cristofalo, Z. Wang, E. Montijano, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1389–1403, 2020.

[54] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game-theoretic planning for self-driving cars in multivehicle competitive scenarios," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1313–1325, 2021.

[55] Y. Jia, M. Bhatt, and N. Mehr, "Rapid: Autonomous multi-agent racing using constrained potential dynamic games," *arXiv preprint arXiv:2305.00579*, 2023.

[56] W. Schwarting, A. Pierson, S. Karaman, and D. Rus, "Stochastic dynamic games in belief space," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2157–2172, 2021.

[57] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3931–3936.

[58] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 981–986.

[59] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016. [Online]. Available: https://doi.org/10.1177/0278364915619772

[60] J. H. Yoo and R. Langari, "Stackelberg game based model of highway driving," in *Dynamic Systems and Control Conference*, vol. 45295. American Society of Mechanical Engineers, 2012, pp. 499–508.

[61] D. Dechert, "Optimal control problems from second-order difference equations," *Journal of Economic Theory*, vol. 19, no. 1, pp. 50–63, 1978.

[62] W. D. Dechert and S. O'Donnell, "The stochastic lake game: A numerical solution," *Journal of Economic Dynamics and Control*, vol. 30, no. 9-10, pp. 1569–1587, 2006.

[63] D. Dragone, L. Lambertini, G. Leitmann, and A. Palestini, "Hamiltonian potential functions for differential games," *Automatica*, vol. 62, pp. 134–138, 2015.

[64] S. Zazo, S. Valcarcel Macua, M. Sánchez-Fernández, and J. Zazo, "Dynamic potential games with constraints: Fundamentals and applications in communications," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3806–3821, 2016.

[65] W. D. Dechert, "Non cooperative dynamic games: a control theoretic approach," *Unpublished. Available on request to the author*, 1997.

[66] M. Bhatt, A. Yaraneri, and N. Mehr, "Efficient constrained multi-agent interactive planning using constrained dynamic potential games," *arXiv preprint arXiv:2206.08963*, 2022.

[67] F. Yang, N. Mehr, and M. Schwager, "Decentralized role assignment in multi-agent teams via empirical game-theoretic analysis," *arXiv preprint arXiv:2109.14755*, 2021.

[68] O. Akcin, P.-h. Li, S. Agarwal, and S. P. Chinchali, "Decentralized data collection for robotic fleet learning: A game-theoretic approach," in *Conference on Robot Learning*. PMLR, 2023, pp. 978–988.

[69] J. Hu, P. Bhowmick, I. Jang, F. Arvin, and A. Lanzon, "A decentralized cluster formation containment framework for multirobot systems," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1936–1955, 2021.

[70] B. Reily, T. Mott, and H. Zhang, "Decentralized and communication-free multi-robot navigation through distributed games," *arXiv preprint arXiv:2012.09335*, 2020.

[71] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory, 2nd Edition*. Society for Industrial and Applied Mathematics, 1998. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611971132

[72] A. Fabrikant, C. Papadimitriou, and K. Talwar, "The complexity of pure nash equilibria," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 2004. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1007352.1007445 pp. 604–612.

[73] M. Ummels, "The complexity of nash equilibria in infinite multiplayer games," in *International Conference on Foundations of Software Science and Computational Structures*. Springer, 2008. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-78499-9_3 pp. 20–34.

[74] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a nash equilibrium," *SIAM Journal on Computing*, vol. 39, no. 1, pp. 195–259, 2009. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/070699652

[75] A. Fonseca-Morales and O. Hernández-Lerma, "Potential differential games," *Dynamic Games and Applications*, vol. 8, no. 2, pp. 254–279, 2018. [Online]. Available: https://link.springer.com/article/10.1007/s13235-017-0218-6

[76] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4906–4913.

[77] E. Camponogara, D. Jia, B. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems Magazine*, vol. 22, no. 1, pp. 44–52, 2002.

[78] Y. Zheng, S. E. Li, K. Li, F. Borrelli, and J. K. Hedrick, "Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 899–910, 2017.

[79] T. Keviczky, F. Borrelli, and G. J. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, no. 12, pp. 2105–2115, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109806003049

[80] A. Richards and J. How, "Decentralized model predictive control of cooperating uavs," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 4, 2004, pp. 4286–4291 Vol.4.

[81] G. Ferrari-Trecate, L. Galbusera, M. P. E. Marciandi, and R. Scattolini, "Model predictive control schemes for consensus in multi-agent systems with single- and double-integrator dynamics," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, pp. 2560–2572, 2009.

[82] W. H. Warren, "Collective motion in human crowds," *Current Directions in Psychological Science*, vol. 27, no. 4, pp. 232–240, 2018. [Online]. Available: https://doi.org/10.1177/0963721417746743

[83] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3299–3304.