Decentralized Multi-Agent Collision Avoidance and Reinforcement Learning

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Hao Li, B.E.

Graduate Program in
Department of Electrical and Computer Engineering

The Ohio State University

2021

Dissertation Committee:

Abhishek Gupta, Advisor

Wei Zhang, Advisor

Parinaz Naghizadeh

Levent Guvenc

Abstract

This dissertation studies decentralized multi-agent collision avoidance and reinforcement learning (RL) for Markov Decision Process (MDP) with state-dependent action constraints. The multi-agent collision avoidance problem is a fundamental problem in robotics, and it can be generally defined as multiple robots navigating in a shared environment while avoiding collisions with each other. It is well known in the literature that multi-agent collision avoidance is challenging to solve, mainly due to complex dynamics constraints, limited information for each agent, and strict safety constraints.

We first propose a decentralized collision avoidance algorithm for heterogeneous multi-agent systems by introducing the extended control obstacles (ECOs). Pairwise state-dependent action constraints from ECOs are introduced to avoid pairwise collisions, which provides strict safety guarantees for heterogeneous linear systems. The overall collision avoidance algorithm for each agent is formulated as a simple convex optimization, which can be solved in real-time. The proposed approach can handle complicated scenarios with uncontrolled agents, nonlinear agents, and obstacles.

In the second part of this dissertation, we propose a fast RL-based decentralized collision avoidance algorithm for general nonlinear agents with continuous action space. To reduce online computation, we first decompose the multi-agent scenario and solve a two agents collision avoidance problem via RL. When extending the trained

policy to a multi-agent problem, safety is enforced by introducing state-dependent action constraints from the optimal reciprocal collision avoidance (ORCA). The overall collision avoidance action could be found through a simple convex optimization in real-time.

Inspired by the collision avoidance algorithms that incorporate state-dependent action constraints, we study RL for continuous MDPs with and state-dependent action constraints. We establish the convergence of fitted value iteration and fitted Q-value iteration. We further extend the algorithms and the convergence result to the case of monotone MDPs, where a function approximating class for the monotone MDPs is identified.

This is dedicated to my grandparents.

Acknowledgments

During my Ph.D. study, I have been greatly impacted by many wonderful people. I cannot make it without their selfless help. First and foremost, I would like to express sincere gratitude to my advisors, Prof. Wei Zhang and Prof. Abhishek Gupta, for their patience, inspirations, and guidance. Both of them are brilliant and patient instructors.

I would also like to thank all of my committee members: Prof. Levent Guvenc and Prof. Parinaz Naghizadeh, for serving my Ph.D. committee and providing valuable advice. Prof. Levent Guvenc was also the committee member of my Qualifying Exam and Candidacy Exam. His expertise in Robotics has greatly inspired and enriched my work.

During my journey to the Ph.D. degree, I have been helped a lot by my enthusiastic labmates, Hua Chen, Huaqing Xiong, Shiping Shao, Bowen Weng, Yueyun Lu, and many more. I benefited a lot from discussions and collaborations with them.

I am extremely grateful to my parents, Hui Li and Xiaoai Wang, for their unwavering support and love through this experience. I would also like to thank my cat, Pepper, for her company. Finally, I would like to offer the greatest thanks to my girlfriend, Dr. Congcong Xue. She led me through the hardest stages of my Ph.D. study.

Vita

2016	B.E., Electrical Engineering, University of Science of Technology of China
2016-2019	Graduate Research Associate, Electrical and Computer Engineering, The Ohio State University.
2019-present	Graduate Teaching Associate, Electrical and Computer Engineering, The Ohio State University.

Fields of Study

Major Field: Electrical and Computer Engineering

Table of Contents

Page

Abstract ii

Dedication iv

Acknowledgments v

Vita vi

List of Tables x

List of Figures xi

1. Introduction 1

    1.1 Overview 1
    1.2 Literature Review 3
        1.2.1 Reciprocal Collision Avoidance 4
        1.2.2 Reinforcement Learning 7
        1.2.3 Reachability Based Approaches 8
        1.2.4 Barrier Functions 10
        1.2.5 Potential Functions 10
        1.2.6 Buffered Voronoi Cells 12
    1.3 A Preview of Main Results and Contributions 12
    1.4 Organization and Notation 15

2. Multi-Agent Collision Avoidance for Heterogeneous Systems via Extended Control Obstacles 17

    2.1 Introduction 17
    2.2 Problem Statement 19
        2.2.1 Problem Setup 19

2.2.2 Problem Statement 20
2.3 The Extended Control Obstacles for Heterogeneous Systems 21
2.3.1 Extended Control Obstacles 21
2.3.2 A Collisions Avoidance Framework for Heterogeneous Systems 22
2.4 Collision Avoidance for Heterogeneous Linear Systems 25
2.4.1 ECO Computations for Linear Systems 25
2.4.2 Linear Safety Constraints Computations 26
2.4.3 Collision Avoidance for Heterogeneous Linear Systems . . . 30
2.5 Uncontrolled Agents and Nonlinear Dynamics 30
2.5.1 Nonlinear Dynamics 31
2.5.2 Uncontrolled Agents and Obstacles 31
2.6 Experiments and Results 34
2.6.1 Simulation Setup 34
2.6.2 Simulation Results 37
2.7 Conclusions 41

3. Multi-agent Collision Avoidance for General Nonlinear Agents via Reinforcement Learning 43

3.1 Introduction 43
3.2 Problem Formulation 45
3.3 Approach 47
3.3.1 Two Agents Collision Avoidance via Reinforcement Learning 48
3.3.2 Multi-agent Collision Avoidance with Improved Safety . . . 50
3.4 Simulations and Results 54
3.4.1 Simulation Setup 55
3.4.2 Policy Training with RL 56
3.4.3 Simulation Results 56
3.5 Conclusions 59

4. Fitted Value Iteration in Continuous Markov Decision Processes with State-Dependent Action Sets 61

4.1 Introduction 61
4.2 Problem Formulation 63
4.2.1 Empirical Bellman Operators 65
4.2.2 Fitted Value Iteration and Q-Value Iteration 65
4.3 Main Results 67
4.3.1 Lipschitz MDP 68
4.3.2 Fitted Value Iteration and Q-Value Iteration 69
4.4 Proofs of the Main Results 71

4.4.1 Probabilistic Contraction Analysis of Iterated Random Operators . . . 71
4.4.2 Proof of Theorem 1 . . . 73
4.4.3 Proof of Theorem 2 and 3 . . . 73
4.5 Extensions to Monotone MDP . . . 74
4.6 Conclusions . . . 77

5. Conclusions and Future Work . . . 78

Appendices 81

A. Proofs . . . 81

A.1 Proof of Lemma 1 . . . 81
A.2 Proof of Theorem 5 . . . 82

Bibliography . . . 84

List of Tables

Table **Page**

1.1 A comparison of multi-agent collision avoidance algorithms. 5

List of Figures

Figure **Page**

2.1 ECO and a set $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ of form (2.7) that satisfies (2.8). The set $\mathcal{H}^{i,j}(\mathbf{s}^{i,j}) \cap \mathscr{A}^{i,j}$ is a set of feasible joint control for agent i and j to avoid collisions with each other. 23

2.2 Linear Safety Constraints Computations for ECO. Given the solution $\mathbf{x}_0$ to the nonconvex optimization (2.12), we could construct the $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ via (2.13). The corresponding $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ is tangent to $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$. 28

2.3 Linear Constraints Decomposition for PECO. Given the solution $\mathbf{x}_0$ to the nonconvex optimization (2.16), we construct the $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ via (2.17). The corresponding $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ is tangent to $\overline{\mathcal{PO}}_N^{i,j}(\mathbf{s}^{i,j})$. 33

2.4 Unicycle Model 35

2.5 Kinematic Bicycle Model 36

2.6 A simulation where one single integrator and one double integrator swapping positions via the proposed approach. The single integrator is denoted by a solid circle and the double integrator is denoted by a ring. Their goal positions are denoted by cross markers with the same colors as the agents themselves. The generated trajectories are marked by dashed lines with corresponding colors. 38

2.7 A simulation that contains two single integrators and two double integrators. 39

2.8 A simulation that contains four agents: a single integrator (solid circle), a double integrator (ring), a unicycle (circle with black arrow), and a bicycle (circle with white arrow). 39

2.9 A simulation that contains eight agents with four types of dynamics. 40

2.10 A simulation that contains an uncontrolled agent. The black circle represents an uncontrolled agent that moves toward its goal position directly. 41

3.1 The Algorithm Framework for RL-based Multi-agent Collision Avoidance Algorithm 48

3.2 Trajectories of three, four and eight agents 57

3.3 Trajectories of 42 agents. The left figure: the initial position of each agent, different agents are marked by dots with different color. The middle figure: the trajectory in the middle of the process. The right figure: complete trajectories. 57

3.4 Trajectories of 20 agents. The left figure: the initial position of each agent is denoted by solid circle with different colors. The middle figure: the trajectory in the middle of the process. The right figure: complete trajectories. 58

3.5 Illustration of the importance of ORCA constraints through a 24 agents example. The left column shows positions of all agents and trajectories of agent 1. The right column shows combined actions and selected actions of agent 1. 60

Chapter 1: Introduction

1.1 Overview

This dissertation studies the decentralized multi-agent collision avoidance and RL algorithms for Markov Decision Processes (MDPs) with state-dependent action constraints. Roughly speaking, the multi-agent collision avoidance problem can be defined as multiple robots navigating in a shared environment while avoiding collisions with each other and obstacles. The problem has recently gained much interest and has many practical applications, including service robots, logistic robotics, search and rescue, and self-driving vehicles. One of the most important criteria to evaluate collision avoidance algorithms are safety, and it can be formulated as state-dependent constraints in the action space. Inspired by this, we also study RL for MDPs with state-dependent action constraints, continuous state space, and continuous action space.

Many multi-agent collision avoidance algorithms have been proposed to solve the aforementioned challenges. They could be categorized into centralized and decentralized approaches. For centralized ones, all agents and the environment are considered as a whole system, and one decision-maker is in charge of control for all agents [1, 2]. In practice, centralized approaches may encounter many issues. First, the centralized

ones rely on fast communications between agents. They are very sensitive to transmission delay. Second, if the decision-maker fails, the whole system would fail, and collision could occur, which makes the system not reliable. Third, the centralized approaches scale poorly to large-scale systems. The computation complexity usually grows exponentially with respect to the number of agents. On the contrary, decentralized approaches allow each agent to make decisions independently. They are less sensitive to time delay, more robust to system failures, and could be easily extended to large systems.

Though traditional collision avoidance problem for a single agent has been extensively studied in the literature, decentralized multi-agent collision avoidance still remains a challenging problem. First, unlike the single agent problem, the environment of each agent is dynamically changing. Each agent needs to coordinate with its nearby agents with limited communications or without communications. Second, the safety constraints introduced for inter-agent collision avoidance are non-convex, which makes the problem hard to solve from the perspective of optimizations. Third, the dynamics constraints introduced by each agent can be complex. Fourth, in practical applications, each agent only has limited observation of other agents and the environment. Usually, each agent is not able to predict the action of nearby agents accurately. Fifth, the real-world scenarios are complicated: the dynamics of all the agents may be different, some agents may not follow the same collision avoidance policy.

Like many other studies in the literature, we consider the realistic case without communications between agents. We first study the decentralized multi-agent collision avoidance problem. Two collision avoidance algorithms are proposed based on

different settings. The first approach handles the realistic applications where the multi-agent system is heterogeneous by introducing the ECO. This algorithm could also handle complicated environments with nonlinear, uncontrolled agents and obstacles. The second approach provides real-time collision avoidance for general nonlinear agents by introducing RL, where the safety constraints are transformed into state-dependent action constraints. Inspired by that, we also investigate the RL algorithms for MDPs with state-dependent action constraints, continuous state space, and continuous action space, where convergence is established.

1.2 Literature Review

In this dissertation, two decentralized collision avoidance algorithms and RL algorithms for MDPs with state-dependent action constraints are presented. A comprehensive literature review on related works is conducted in this section. A comparison of common decentralized multi-agent collision avoidance is given in Table 1.1. Some related centralized algorithms are also included for completeness. We compare these algorithms from different perspectives: whether they are centralized or decentralized, if they provide any safety guarantee, what model assumptions they have about each agent. Note that for the safety property, "safe if feasible" means safety is guaranteed when all safety constraints can be satisfied.

The goal of multi-agent collision avoidance is to find a control trajectory for each agent that avoids collisions with each other while driving it to its corresponding goal state. For decentralized algorithms, the control of each agent depends on its goal state, internal state, observations of other agents, and the environment.

Decentralized multi-agent collision avoidance algorithms can be generally divided into two categories. The first category adopts a hierarchical structure in design, where a high-level planning module is used to guide each agent to its goal state, and a low-level collision avoidance module is used to avoid collisions. The focus of this category is the design of low-level collision avoidance controllers, which would only regulate the control reference from the planning module when collision becomes impendent. This category of algorithms includes some reachability approaches [3], reciprocal collision avoidance [4, 5], barrier functions [6, 7, 8]. The second category considers planning together with collision avoidance, this category includes potential functions [9], RL algorithms [10], and geometric approach [11].

1.2.1 Reciprocal Collision Avoidance

The reciprocal collision avoidance (RCA) algorithms formulate the collision avoidance problem as a simple convex optimization with linear constraints. It usually provides a fast online solution to avoid collisions. A pair of linear control constraints are introduced to avoid collisions between two agents. The constraints in control space are constructed via velocity obstacles [36] or control obstacles [12], which are defined as a set of relative constant velocities or controls that would cause collisions within a given time horizon. Two agents would not collide with each other within the given time horizon if constant controls satisfying the constraints are applied. When extended to multiple agents, each agent selects the closest control to the control reference from a high-level planning module by solving a simple convex optimization with linear constraints induced by surrounding agents.

Algorithm	Centralized or Decentralized	Safety	Model Assumption
Reciprocal Collision Avoidance			
ORCA [5]	decentralized	Safe if feasible	Single integrators
Generalized RCA [12]	decentralized	Safe if feasible and homogeneous linear	Homogeneous linear or heterogeneous nonlinear
AVO [13]	decentralized	Safe if feasible	Double integrators
CCO [14]	decentralized	Safe if feasible	Dynamics with continuous constraints
LQR-obstacles [15]	decentralized	Safe if feasible	Homogeneous linear
ORCA with MPC [16]	decentralized	Safe if feasible	Double integrators
ORCA with flatness-based MPC [16]	decentralized	No	Quadrotors
NH-ORCA [17]	decentralized	Safe if feasible	Nonlinear dynamics with a velocity tracking controller
B-ORCA [18]	decentralized	Safe if feasible	Unicycle
Reinforcement Learning			
CADRL [19]	decentralized	No	Single integrator with kinematic constraints
SA-CADRL [10]	decentralized	No	Single integrator with kinematic constraints
GA3C-CADRL [20, 21]	decentralized	No	Single integrator with kinematic constraints
GA3C-CADRL-NSL [22]	decentralized	No	Single integrator with kinematic constraints
PPO with Multiple Robots [23, 24]	decentralized	No	Single integrator with kinematic constraints
DeepMNavigate [25]	centralized	No	Single integrator with kinematic constraints
Reachability Approach			
Evasion-evasion [3]	decentralized	Safe for up to 3 agents	Double integrators
Pursuit-evasion with MIP [26]	centralized	Safe for up to 3 agents	Arbitrary
Safe sequential planning [27, 28, 29]	centralized	Yes	Arbitrary model with disturbance
Barrier Functions			
CBF [30, 7, 6, 31]	decentralized	Safe if feasible	Double integrators
Potential Functions			
DNF [9, 32]	decentralized	Yes	Single integrators
Artificial Potential Functions [33]	decentralized	No	Double integrators
Buffered Voronoi Cells			
BVC [11]	decentralized	Yes	Single integrators
PBVC [34]	decentralized	No	Single integrators with noise
B-UVC [35]	decentralized	Safe with Probability	Single integrators with noise

Table 1.1: A comparison of multi-agent collision avoidance algorithms.

Among many velocity obstacle based approaches [37, 4, 5, 38], the optimal reciprocal collision avoidance (ORCA) [5] is the most influential one. It assumes that each agent could be modeled as a single integrator, and its velocity can directly be controlled. Each agent could also observe the position and velocity of nearby agents. Although ORCA provides fast online solutions for collision avoidance, the assumption that velocity can be directly controlled limits its real-world applications. There are many extensions of ORCA to other dynamics. They can be generally categorized into two types. The first type of algorithms extends the definitions of velocity obstacles to the control space. They usually assume that the control of nearby agents can be observed. [12] summarizes some previous extensions of this type [5, 13, 14, 15], and propose the generalized RCA algorithm to linear homogeneous dynamics. The generalized RCA algorithm is based on the concept of control obstacles, which is a natural extension of velocity obstacles to control space. The generalized RCA could also be extended to nonlinear, nonhomogeneous dynamics by linearization, which would introduce error and safety could no longer be guaranteed. Another type of ORCA extensions is to directly apply ORCA with fixed velocity tracking controllers [18, 39, 17]. The introduction of tracking controllers would cause the tracking error between the nominal single integrators and real dynamics, which is compensated via enlarging the agent radius by the tracking error in computation.

The RCA algorithms could provide fast solutions to multi-agent collision avoidance. However, most of their applications are limited to certain simple dynamics. Besides, it is generally hard to compute linear safety constraints for arbitrary linear systems. So far, most works are based on homogeneous linear dynamics, and the

extensions to heterogeneous or nonlinear would add linearization error or extra conservativeness. The optimization problem could become infeasible when the intersection of multiple linear constraints is empty.

1.2.2 Reinforcement Learning

The development of modern computation tools and deep neural networks have made reinforcement learning (RL) algorithms a popular tool to solve control problems with high dimensions and continuous control space in a model-free manner [40, 41, 42, 43]. The neural networks are usually used to approximate the value functions or policies. Compared to traditional model-based approaches, RL could handle complex systems with very high dimensions of state space and action spaces. The online computation is very fast, since RL offloads online computation to the training process. RL could also handle disturbance directly in their formulation. These are very desirable properties for multi-agent collision avoidance problems with complex dynamics. More and more researchers are using RL to solve multi-agent collision avoidance problems [23, 24, 19, 10, 20, 44].

For RL-based decentralized multi-agent collision avoidance algorithms, one main challenge is how to represent the state for RL training. Since the number of agents in the environment could vary, it is not proper to directly concatenate the output of nearby agents. To address this problem, the authors in [19] proposed to train a value function with two agents, and then extend the obtained policy to the multi-agent case by selecting the action that maximizes the minimum of pairwise value functions. A symmetric neural network structure is adopted in [10], which makes all neighbors of one agent have the same impact on the policy. Long short-term memory networks

(LSTM) [45] are introduced in [20] to handle the varying number of agents. Raw sensor data is used for part of state in [23, 24].

One key issue of RL-based approaches is difficulty in providing a safety guarantee. Though different RL algorithms have been applied for collision avoidance, the collision avoidance behavior is achieved by constructing reward functions that award agents for reaching goals and penalize dangerous behaviors. Thus, there are no hard constraints in terms of safety. Another issue is that most of the aforementioned studies are restricted to discrete action space, and require rather complex neural networks with demanding training processes. Besides, they all assume that the velocities of robots could be directly controlled, which limits their applications to robots with more complex nonlinear dynamics. This issue originates from the inconsistency of MDP assumptions. Though RL algorithms assume the environment to be an MDP, this is not true in the training process of decentralized multi-agent collision avoidance. Since the behaviors of surrounding agents would evolve with the policy or value functions, the environment is no longer stationary.

1.2.3 Reachability Based Approaches

The most important aspect of evaluating a collision avoidance algorithm is safety, and reachability analysis can provide a strict guarantee for safety [46]. In reachability analysis, a backward reachable set (BRS) is usually computed, which can be defined informally as a set of initial states start from which the agent could reach a target set. If the target set is chosen to be the set of collisions, the complement of the BRS is a safe set and control-invariant. There always exists at least one safe policy to avoid collisions if the initial state is within the safe set.

One common way to compute the BRS and the corresponding safe policy is via Hamilton-Jacobi (HJ) reachability, where the collision avoidance problem is defined as an optimal control problem [47, 3]. The collision avoidance problem between two agents is formulated as an evasion-evasion game in [3], and the corresponding policy to avoid collisions is computed analytically for double integrators dynamics. The evasion-evasion game formulation provides the least conservative solution for two agents collision avoidance, but safety could not be guaranteed when extended to multiple agents. To alleviate the conflicts introduced by multiple agents, centralized extensions, such as mixed-integer programming [26] and sequential planning algorithm [27, 28, 29] are introduced. In [26, 48], the collision avoidance problem for two agents is formulated as a pursuit-evasion game. Though solutions from the pursuit-evasion game formulation are more conservative compared to the evasion-evasion formulation, they can handle more complex environments with uncontrolled moving obstacles.

Though reachability could guarantee safety in relatively simple scenarios, its computation burden always limits its applications to more complex dynamics or environments. The computation complexity usually grows exponentially with respect to the dimension of state space. Therefore, most of its applications are for very simple dynamics, such as single or double integrators. To alleviate the computation burden, it is common to decompose the collision avoidance into pairwise collision avoidance problems. Safety is usually sacrificed, and a central coordinator becomes necessary to handle conflicts when extended to multi-agent scenarios.

1.2.4 Barrier Functions

Similar to the reachability analysis, the barrier function is a powerful tool to guarantee the safety of a multi-agent system by enforcing a safe set control-invariant [49]. But unlike reachability analysis, the barrier function does not require the safe set or the corresponding policy to be calculated explicitly. Therefore, the barrier functions could provide a faster solution. For control-affine systems, the Lie derivative of the barrier functions introduces pairwise linear control constraints for two agents. Similar to RCA algorithms, the pairwise linear control constraints from different neighbors are considered, and the control problem is formulated as an optimization when extended to multiple agents [30, 7, 6, 31]. In the construction of barrier functions for two agents, only the states of two agents are considered. This is less conservative than RCA algorithms that usually requires both the state and control of an agent to be observable.

Though the barrier function provides a fast online solution to the multi-agent collision avoidance problem, its applications is limited to certain forms of control-affine systems, such as double integrators, unicycles. Similar to Lyapunov functions, there is no universal algorithm to construct barrier functions. The safe set could be too small, and the policy to avoid collisions might be too conservative if the barrier function is not properly constructed. Besides, the optimization problem could be infeasible when the intersection of multiple linear constraints becomes empty.

1.2.5 Potential Functions

Potential functions handle the collision avoidance problems from the perspective of optimization by controlling all agents move towards the opposite direction of their

gradients. They are first designed for single agent collision avoidance [50], known as the *navigation function*. Potential functions are usually constructed to drive all agents to move towards their goal states while avoiding collisions [51]. Potential functions are closely related to value functions [52] and Lyapunov functions [9] in the sense that the goal state is the only global minimum that makes functions vanish.

To extend the navigation function to multiple agents, [53] proposed a centralized algorithm for multi-agent formation control. It proposes a multi-agent navigation function with similar convergence and safety properties as [54]. [9] proposes another form of multi-agent navigation function for decentralized collision avoidance problem with proved convergence and safety properties [32]. The sum of multi-agent navigation functions is shown to be a global Lyapunov function. The extension of navigation functions to multi-agent systems provides an approach to handle collision avoidance via optimization and Lyapunov theory. It can provide safety and convergence in a decentralized manner. However, discussions have been mainly focused on single integrators. Though [52] proposes a potential function for multi-agent collision avoidance with general nonlinear dynamics, only equivalent conditions for safety and stability are provided. The construction of navigation functions is limited to certain forms of function.

Another trend of potential functions is to focus on practical applications instead of strictly analysis for safety [33, 55, 56]. These approaches could easily incorporate different types of obstacles, environments, and social norms. But there is no safety guarantee or analysis, and they usually require tedious parameter-tuning to get good performance.

1.2.6 Buffered Voronoi Cells

The Buffered Voronoi Cell (BVC) [11] handles the decentralized multi-agent collision avoidance by segmenting the workspace into polyhedra. Each agent computes its collision avoidance action by solving a receding horizon control problem with linear constraints. The BVC originates from the concept of Voronoi Cells [57], which has been widely applied to many robotics problems, such as path planning problems [58], coverage control problems [59]. The BVC extends the concept of Voronoi Cells by incorporating the size of each agent in the partitions in the workspace. It is has been extended to single integrators dynamics with disturbances in localization [35, 34]. Similar to RCA approaches and barrier functions, pairwise linear constraints to avoid pairwise collisions are introduced. Unlike those approaches, BVC only requires each agent to observe the positions of nearby agents. However, the main discussions of the BVC algorithms focus on single integrators dynamics, which limits their applications in the real world.

1.3 A Preview of Main Results and Contributions

The main focus of this dissertation is decentralized multi-agent collision avoidance and RL. Two decentralized multi-agent collision avoidance algorithm is proposed. The first algorithm handles collision avoidance problems for heterogeneous systems by introducing the extended control obstacle (ECO). The second one tackles collision avoidance for general nonlinear agents with continuous action space by combining ORCA and RL. The collision avoidance constraints are formulated as state-dependent action constraints in both collision avoidance algorithms. Inspired by this, we develop

RL algorithms for MDPs with state-dependent action constraints and establish convergence results. The algorithms and convergence are also extended to the case of monotone MDPs.

Though real-world applications of collision avoidance algorithms require complicated interactions between agents with different dynamics, the majority of decentralized algorithms are designed for homogeneous systems. Therefore, the first part of this dissertation focuses on collision avoidance of heterogeneous systems. Our work is inspired by [12], which extends the definition of velocity obstacles to control space and proposes control obstacles. We propose the extended control obstacle (ECO) and the corresponding collision avoidance algorithm for heterogeneous systems. The ECO is defined as a set of constant joint controls between two agents that could result in collisions within a given time horizon. For linear systems, it can be computed as a polyhedron or union of polyhedra. Given an ECO, the corresponding linear constraints can be computed independently by two agents, which can be achieved by solving convex optimizations. The proposed collision avoidance algorithm works for general heterogeneous systems. There is no limitation to the dimensions of action space like [12]. This algorithm can also handle complex environments with nonlinear agents, uncontrolled agents, and obstacles. The corresponding linear constraints provide a strict safety guarantee for heterogeneous linear systems. The main computation processes can be formulated as convex optimizations, and our algorithm can provide a real-time solution for collision avoidance.

In the second part of the dissertation, we propose a decentralized multi-agent collision avoidance algorithm that combines RL and ORCA. One common issue of most decentralized multi-agent collision avoidance algorithms is that they only work

for specific dynamics or very simple dynamics. Though RL-based approaches are applicable to general nonlinear dynamics in theory, in practice, they only work for very simple dynamics where the velocity can be directly controlled. The non-stationary environments make it hard for RL algorithms to converge if the dynamics are complex. Moreover, RL-based approaches do not provide any safety guarantee. On the contrary, the proposed RL-based algorithm solves the multi-agent collision avoidance problem with a systematic consideration of safety. Our approach is designed for robots with general nonlinear dynamics, where each agent can only observe the positions and velocities of nearby robots. We first decompose our problem into a two agents collision avoidance problem with continuous action spaces, and solve it using RL. Since we only train the RL policy for two agents collision avoidance, the learning process of our algorithm is much faster, and the neural network is much smaller. Then we handle the multi-agent collision avoidance problem by solving a simple convex optimization with safety constraints from the ORCA. The linear safety constraints from the ORCA algorithm acts as state-dependent action constraints in the optimization to avoid collisions. Unlike other RL-based approaches that assume velocities of robots could be directly controlled, or other ORCA-based approaches that need specific models and controllers, our approach works for general nonlinear systems with continuous action spaces. The performance of the proposed approach is demonstrated via complex and challenging tasks in simulations.

In the third part of the dissertation, we study RL algorithms for MDPs with state-dependent action constraints. This work is inspired by our previous works on collision avoidance, where the safety constraints are converted into state-dependent

action constraints. We establish the convergence of fitted value iteration and fitted Q-value iteration for continuous MDPs with state-dependent action constraints. Unlike most fitted value iteration problems [60, 61, 62, 63], we assume the admissible action set are state-dependent, which is more realistic in many applications. We establish the Lipschitz continuity of value functions, and relax the absolute continuity assumptions on the transition kernel of the MDPs [60, 63, 64]. We also extend the convergence result to a sufficiently general class of monotone MDPs.

1.4 Organization and Notation

In Chapter 2, we study the collision avoidance algorithm for heterogeneous systems by introducing the extended control obstacle. In Chapter 3, we propose an efficient RL-based algorithm to solve multi-agent collision avoidance problems with a systematic consideration of safety. Our approach is designed for robots with general nonlinear dynamics, where each agent can only observe the positions and velocities of nearby robots. In Chapter 4, we establish the convergence of fitted value iteration and fitted Q-value iteration for continuous-state continuous-action Markov decision problems (MDPs) with state-dependent action sets. We further extend the algorithm and the convergence result to the case of monotone MDPs. Chapter 5 concludes the dissertation and discusses potential future research directions.

We use the following notational conventions throughout this dissertation. The set of natural numbers and the set of non-negative integers are denoted by $\mathbb{N}$ and $\mathbb{N}_0$, respectively. Cartesian product between sets $\mathscr{A}$ and $\mathscr{B}$ are denoted by $\mathscr{A} \times \mathscr{B}$. For a set $\mathscr{A}$, we use $\partial(\mathscr{A})$ to represent its boundary. Given a set $\mathscr{S}$, the set of bounded measurable functions mapping from $\mathscr{S}$ to $\mathbb{R}$ is denoted by $\mathcal{B}(\mathscr{S})$, and the set of

continuous bounded functions endowed with supremum norm mapping from $\mathscr{S}$ to $\mathbb{R}$ is denoted by $\mathcal{C}_b(\mathscr{S})$.

Chapter 2: Multi-Agent Collision Avoidance for Heterogeneous Systems via Extended Control Obstacles

2.1 Introduction

In this chapter, we study the decentralized multi-agent collision avoidance problem for heterogeneous systems. Unlike the homogeneous multi-agent systems, the existence of different dynamics makes the collision avoidance problem combinatorial in nature. Therefore, it is very challenging to apply existing decentralized multi-agent collision avoidance algorithms to heterogeneous systems. Though [6] claims barrier functions can be applied to heterogeneous systems, the dynamics in that work is actually homogeneous with different action constraints. The reciprocal collision avoidance algorithm proposed in [12] can also be applied to heterogeneous systems, but it comes with many limitations and disadvantages. First, its theoretic development is based on homogeneous linear systems. Linearization is necessary when the algorithm is extended to heterogeneous systems, which would introduce linearization errors. Second, the dimension of control must be the same as the dimension of the workspace, which may not be true for many applications. Third, approximations are necessary to compute linear safety constraints for linear systems, which would introduce extra errors.

To address the aforementioned issues, we propose a decentralized collision avoidance algorithm for heterogeneous multi-agent systems by introducing the *extended control obstacle* (ECO). The ECO can be generally defined as a set of constant joint controls for two agents that may cause collisions within a time horizon. Unlike velocity obstacles or control obstacles that are defined in a relative control space, ECOs are defined in a joint control space between two agents, which makes it suitable for heterogeneous systems in theory. Pairwise linear constraints are introduced for each pair of heterogeneous agents to avoid collisions, which can be computed via convex optimizations. The overall collision avoidance algorithm for each agent is formulated as an independent convex optimization that can be solved in real-time. The proposed approach is applicable to general heterogeneous systems, and can also handle scenarios with uncontrolled agents that do not follow the proposed collision avoidance framework. The resulting collision avoidance strategy is verified via challenging simulations over heterogeneous systems, which include single integrators, double integrators, unicycles, and kinematic bicycles. The main contributions are as follows.

- **Heterogeneous Systems:** We propose a collision avoidance algorithm for general heterogeneous systems. Unlike [12], there are no limitations to the dimension of control spaces.

- **Unified Framework:** The proposed collision avoidance algorithm can handle complex scenarios with uncontrolled agents, nonlinear agents, and obstacles.

- **Safety:** The introduction of ECO and the corresponding linear constraints provide strict safety guarantees for heterogeneous linear systems.

- **Real-Time Computation:** Our algorithm can provide real-time control. The main computation processes can be formulated as convex optimizations.

2.2 Problem Statement

2.2.1 Problem Setup

We consider a discrete time heterogeneous multi-agent system consisting of M agents, indexed by $\mathcal{M} \triangleq \{1, \ldots, M\}$. Each agent $i \in \mathcal{M}$ is modeled by the following nonlinear dynamics

$$\mathbf{s}_{t+1}^i = f^i(\mathbf{s}_t^i, \mathbf{a}_t^i), \tag{2.1a}$$

$$\mathbf{p}_t^i = g^i(\mathbf{s}_t^i). \tag{2.1b}$$

Here $\mathbf{s}_t^i \in \mathscr{S}^i \subseteq \mathbb{R}^{n_i}$, $\mathbf{a}_t^i \in \mathscr{A}^i \subseteq \mathbb{R}^{m_i}$ and $\mathbf{p}_t^i \in \mathbb{R}^d$ represent the state, control and position of agent i at time $t \geq 0$, respectively. The control space $\mathscr{A}^i$ is assumed to be a bounded convex set. The function $f^i(\cdot, \cdot) : \mathscr{S}^i \times \mathscr{A}^i \to \mathscr{S}^i$ is the state transition function and $g^i(\cdot) : \mathscr{S}^i \to \mathbb{R}^d$ is the position projection function from state space to physical workspace for agent $i \in \mathcal{M}$. All agents share the same physical workspace, and $d = 2$ or 3 typically. Each agent $i \in \mathcal{M}$ is modeled as a disc or a sphere with radius $r_i > 0$.

For a two agents system with agents i and j, let the joint state, control and position at time $t \geq 0$ be written as

$$\mathbf{s}_t^{i,j} \triangleq (\mathbf{s}_t^i, \mathbf{s}_t^j) \in \mathscr{S}^{i,j} \triangleq \mathscr{S}^i \times \mathscr{S}^j,$$

$$\mathbf{a}_t^{i,j} \triangleq (\mathbf{a}_t^i, \mathbf{a}_t^j) \in \mathscr{A}^{i,j} \triangleq \mathscr{A}^i \times \mathscr{A}^j,$$

$$\mathbf{p}_t^{i,j} \triangleq (\mathbf{p}_t^i, \mathbf{p}_t^j) \triangleq g^{i,j}(\mathbf{s}_t^{i,j}) \in \mathbb{R}^{2d}.$$

Here $\mathscr{S}^{i,j}$, $\mathscr{A}^{i,j}$ represent the joint state space and control space respectively, and $g^{i,j}(\cdot) : \mathscr{S}^{i,j} \to \mathbb{R}^{2d}$ is the position projection function from the joint state space to the joint physical workspace $\mathbb{R}^{2d}$. Given an initial state $\mathbf{s}^{i,j} \in \mathscr{S}^{i,j}$ and an open loop control $\mathbf{a}^{i,j}_{(\cdot)} : \mathbb{N}_0 \to \mathscr{A}^{i,j}$, the corresponding state trajectory is denoted by $\xi^{i,j}(\cdot; \mathbf{s}^{i,j}, \mathbf{a}^{i,j}_{(\cdot)}) : \mathbb{N}_0 \to \mathscr{S}^{i,j}$.

Two agents collide with each other if the distance between them is not greater than the sum of their radii. Equivalently, we say two agents collide at time t if $\mathbf{s}^{i,j}_t \in \mathscr{O}^{i,j}$. Here $\mathscr{O}^{i,j}$ is called the *collision set*, which is defined as

$$\mathscr{O}^{i,j} \triangleq \left\{ \mathbf{s}^{i,j} \in \mathscr{S}^{i,j} \middle| \left\| [I_d \ -I_d] \, g^{i,j}(\mathbf{s}^{i,j}) \right\|_2 \leq r_i + r_j \right\}.$$

Given the collision set $\mathscr{O}^{i,j}$, we define the *dangerous set* $\widetilde{\mathscr{O}}^{i,j}$ as an polyhedron that outer-approximate the collision set $\mathscr{O}^{i,j}$, i.e., $\mathscr{O}^{i,j} \subseteq \widetilde{\mathscr{O}}^{i,j}$.

2.2.2 Problem Statement

The problem of multi-agent collision avoidance now could be defined as having each agent $i \in \mathcal{M}$ compute a control $\mathbf{a}^i \in \mathscr{A}^i$ given the current state $\mathbf{s}^i$, such that it would not collide with any other agent j:

$$\xi^{i,j}(1; \mathbf{s}^{i,j}, \mathbf{a}^{i,j}) \notin \mathscr{O}^{i,j} \quad \forall i, j \in \mathcal{M} \text{ and } i \neq j, \tag{2.2}$$

here $\mathbf{s}^{i,j} = (\mathbf{s}^i, \mathbf{s}^j)$and $\mathbf{a}^{i,j} = (\mathbf{a}^i, \mathbf{a}^j)$ represents the current joint state and joint control of agents i and j.

The multi-agent collision avoidance problem (2.2) is challenging because each agent i does not know the intentions of nearby agents $j \neq i$. Therefore, we develop our algorithm based on the following assumptions:

Assumption 1. *The following holds*

(i) At each time $t \geq 0$, every agent $i \in \mathcal{M}$ would receive a control reference $\tilde{\mathbf{a}}_t^i$ from a high-level planning module.

(ii) Every agent i can identify the dynamics (2.1) of nearby agents $j \neq i$, as well as the radius r_j.

(iii) Every agent i can observe states $\mathbf{s}_t^j$ and controls $\mathbf{a}_t^j$ of each other agent $j \in \mathcal{M}$ at each time $t \geq 0$.

Note that Assumption 1 (i) implies that there exists a planning module that provides the control reference, which may lead to collisions. Assumption 1 (ii) and (iii) implies that each agent can observe the dynamics, state and control of nearby agents, which are very common assumptions for reciprocal collision avoidance algorithms [5, 12].

2.3 The Extended Control Obstacles for Heterogeneous Systems

In this section, we propose a collision avoidance framework for heterogeneous systems by introducing the concept of the *Extended Control Obstacles (ECO)*.

2.3.1 Extended Control Obstacles

For a two agents system consisting of agents i and j, given its initial state $\mathbf{s}^{i,j} \in \mathscr{S}^{i,j}$ at time 0, the *Extended Control Obstacle (ECO) at time t* is defined as the set of constant joint control that would drive the system into the *dangerous set* $\widetilde{\mathscr{O}}^{i,j}$ at time $t \in \mathbb{N}$, i.e.,

$$\mathcal{CO}^{i,j}(t, \mathbf{s}^{i,j}) \triangleq \left\{ \mathbf{a}^{i,j} \in \mathbb{R}^{m_i+m_j} \middle| \xi^{i,j}(t; \mathbf{s}^{i,j}, \mathbf{a}^{i,j}) \in \widetilde{\mathscr{O}}^{i,j} \right\}. \tag{2.3}$$

Then we define the *Extended Control Obstacle (ECO) with Horizon* N as the union of $\mathcal{CO}^{i,j}(t, \mathbf{s}^{i,j})$ over all time $t = 1, 2, \ldots, N$ for some horizon $N \in \mathbb{N}$

$$\begin{aligned} \mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j}) &\triangleq \bigcup_{t=1,2,\ldots,N} \mathcal{CO}^{i,j}(t, \mathbf{s}^{i,j}) \\ &= \left\{ \mathbf{a}^{i,j} \in \mathbb{R}^{m_i+m_j} \middle| \exists t = 1, 2, \ldots, N, \text{ s.t. } \xi^{i,j}(t; \mathbf{s}^{i,j}, \mathbf{a}^{i,j}) \in \widetilde{\mathscr{O}}^{i,j} \right\}. \end{aligned} \tag{2.4}$$

By the definition of $\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j})$, collisions between agents i and j would not occur in the following N steps if they apply a constant joint control $\mathbf{a}^{i,j} \in \mathscr{A}^{i,j}$ for the following N steps such that

$$\mathbf{a}^{i,j} \notin \mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j}). \tag{2.5}$$

2.3.2 A Collisions Avoidance Framework for Heterogeneous Systems

For decentralization, we need to decompose the constraint (2.5) into two independent linear control constraints

$$\mathbf{a}^i \in \mathcal{H}_i^{i,j}(\mathbf{s}^{i,j}), \tag{2.6a}$$

$$\mathbf{a}^j \in \mathcal{H}_j^{i,j}(\mathbf{s}^{i,j}), \tag{2.6b}$$

where $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j}) \subseteq \mathbb{R}^{m_i}$ and $\mathcal{H}_j^{i,j}(\mathbf{s}^{i,j}) \subseteq \mathbb{R}^{m_j}$ are two polyhedra such that for set $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$, which is defined as

$$\mathcal{H}^{i,j}(\mathbf{s}^{i,j}) \triangleq \mathcal{H}_i^{i,j}(\mathbf{s}^{i,j}) \times \mathcal{H}_j^{i,j}(\mathbf{s}^{i,j}) \subseteq \mathbb{R}^{m_i+m_j}, \tag{2.7}$$

we have

$$\mathcal{H}^{i,j}(\mathbf{s}^{i,j}) \cap \mathscr{A}^{i,j} \neq \emptyset, \tag{2.8a}$$

$$\mathcal{H}^{i,j}(\mathbf{s}^{i,j}) \cap \mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j}) = \emptyset. \tag{2.8b}$$

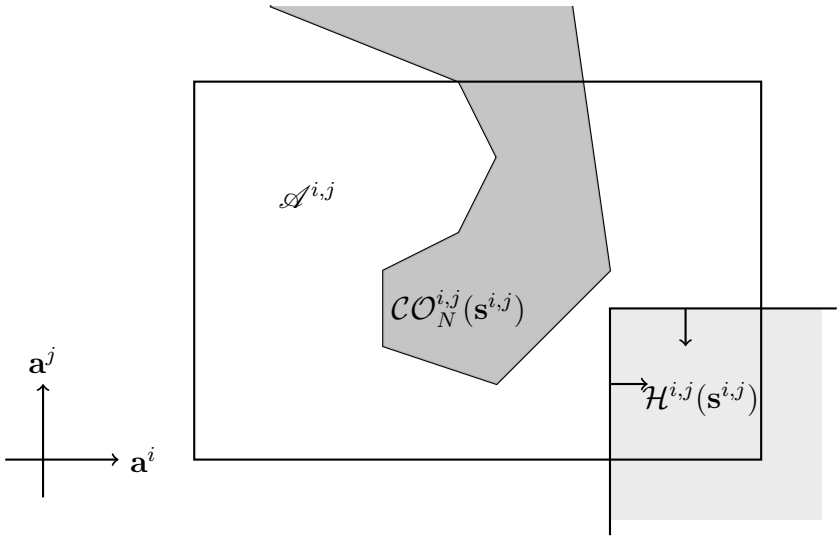


Figure 2.1: ECO and a set $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ of form (2.7) that satisfies (2.8). The set $\mathcal{H}^{i,j}(\mathbf{s}^{i,j}) \cap \mathscr{A}^{i,j}$ is a set of feasible joint control for agent i and j to avoid collisions with each other.

Here (2.8a) implies that there exists feasible control in the set $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$, and (2.8b) implies the control from $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ would not cause collisions. As illustrated in Fig. 2.1, the set $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ satisfying (2.8) is a set of safe joint control. It provides a pair of linear constraints for agent i and j to avoid collisions. Such a $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ always exists if $\left(\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j})\right)^c \cap \mathscr{A}^{i,j} \neq \emptyset$.

To avoid collisions with all other agents $j \neq i$, each agent i would select a control satisfying all linear constraints (2.6a) introduced by all other agent j, while trying to follow the reference control $\tilde{\mathbf{a}}_t^i$. Given the current joint state $\mathbf{s}_t^{i,j} = \mathbf{s}^{i,j}$, each agent i compute its control $\mathbf{a}_t^i = \mathbf{a}^i$ every time step by solving the following convex

Algorithm 1 A Collision Avoidance Framework for Heterogeneous Systems via ECO

Input: Initial time $t = 0$, the initial state $\mathbf{s}_0^i = \mathbf{s}_{initial}^i$, a control reference trajectory $\tilde{\mathbf{a}}_{(\cdot)}^i : \mathbb{N}_0 \to \mathscr{A}^i$ for each agent $i \in \mathcal{M}$, time horizon $N \geq 0$

while True **do**
  for $i \in \mathcal{M}$ **do**
    for $j \in \mathcal{M}$ and $j \neq i$ **do**
      Observe current state $\mathbf{s}_t^{i,j} = \mathbf{s}^{i,j}$
      Compute the ECO $\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j})$
      Compute the polyhedron $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ such that (2.8) holds
    end for
    Compute the current control $\mathbf{a}^i$ via (2.9)
    Apply $\mathbf{a}^i$ to agent i: $\mathbf{a}_t^i = \mathbf{a}^i$
  end for
  $t \leftarrow t + 1$
end while

optimization problem at each time step

$$\mathbf{a}^i = \underset{\mathbf{a} \in \mathbb{R}^{m_i}}{\arg\min} \left\| \mathbf{a} - \tilde{\mathbf{a}}_t^i \right\|_2 \tag{2.9a}$$

$$\text{s.t. } \mathbf{a} \in \mathscr{A}^i \cap \bigcap_{j \in \mathcal{M},\ j \neq i} \mathcal{H}_i^{i,j}(\mathbf{s}^{i,j}). \tag{2.9b}$$

The optimization (2.9) is a quadratic programming with convex constraints. The constraint (2.9b) represents a state-dependent action constraints. If $\mathscr{A}^i$ is a polyhedron, the constraint (2.9b) becomes linear inequality constraints.

The complete collision avoidance framework for heterogeneous systems is summarized in Algorithm 1. For each agent i at each time step t, it needs to compute the $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ with respect to each other agent $j \neq i$, and solve the optimization (2.9) with state-dependent constraints to avoid collisions. In the following sections, we discuss how to compute the ECO $\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j})$ and construct the corresponding polyhedron $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$.

2.4 Collision Avoidance for Heterogeneous Linear Systems

In this section, we propose a collision avoidance algorithm for heterogeneous linear systems based on ECOs. For linear systems ,the dynamics (2.1) becomes the following:

$$\mathbf{s}_{t+1}^i = A^i \mathbf{s}_t^i + B^i \mathbf{a}_t^i + \mathbf{c}^i, \tag{2.10a}$$

$$\mathbf{p}_t^i = D^i \mathbf{s}_t^i + \mathbf{e}^i, \tag{2.10b}$$

where $A^i \in \mathbb{R}^{n_i \times n_i}$, $B^i \in \mathbb{R}^{n_i \times m_i}$, $\mathbf{c}^i \in \mathbb{R}^{n_i}$, $D^i \in \mathbb{R}^{d \times n_i}$, and $\mathbf{e}^i \in \mathbb{R}^d$, for each agent $i \in \mathcal{M}$.

For a system consisting of two agents i and j, given an initial joint state $\mathbf{s}^{i,j} \in \mathscr{S}^{i,j}$, and a constant control $\mathbf{a}^{i,j} \in \mathscr{A}^{i,j}$, the state trajectory could be written as the following

$$\xi^{i,j}(t; \mathbf{s}^{i,j}, \mathbf{a}^{i,j}) = A_t^{i,j} \mathbf{s}^{i,j} + B_t^{i,j} \mathbf{a}^{i,j} + \mathbf{c}_t^{i,j},$$

where $A_t^{i,j}$, $B_t^{i,j}$ and $\mathbf{c}_t^{i,j}$ is given as

$$\begin{aligned}
A_t^{i,j} &= \begin{bmatrix} A^i & \\ & A^j \end{bmatrix}^t \in \mathbb{R}^{(n_i+n_j)\times(n_i+n_j)}, \\
B_t^{i,j} &= \left(\sum_{k=0}^{t-1} A_k^{i,j} \right) \begin{bmatrix} B^i & \\ & B^j \end{bmatrix} \in \mathbb{R}^{(n_i+n_j)\times(m_i+m_j)}, \\
\mathbf{c}_t^{i,j} &= \left(\sum_{k=0}^{t-1} A_k^{i,j} \right) \begin{bmatrix} \mathbf{c}^i \\ \mathbf{c}^j \end{bmatrix} \in \mathbb{R}^{(n_i+n_j)}.
\end{aligned}$$

2.4.1 ECO Computations for Linear Systems

Recall that we approximate the collision set $\mathscr{O}^{i,j}$ by a dangerous set $\widetilde{\mathscr{O}}^{i,j}$, which is a polyhedron. We assume it could be written in the following form

$$\widetilde{\mathscr{O}}^{i,j} = \left\{ \mathbf{s}^{i,j} \in \mathbb{R}^{n_i+n_j} \middle| H_o \mathbf{s}^{i,j} \leq \mathbf{h}_o \right\},$$

where $H_o \in \mathbb{R}^{n_o \times (n_i+n_j)}$, and $\mathbf{h}_o \in \mathbb{R}^{n_o}$ for some $n_o \in \mathbb{N}$. Fix $t \in \mathbb{N}$, the ECO at time t, $\mathcal{CO}^{i,j}(t, \mathbf{s}^{i,j})$, can be represented as a polyhedron with the following form

$$\begin{aligned}\mathcal{CO}^{i,j}(t, \mathbf{s}^{i,j}) =& \left\{ \mathbf{a}^{i,j} \in \mathbb{R}^{m_i+m_j} \middle| A_t^{i,j}\mathbf{s}^{i,j} + B_t^{i,j}\mathbf{a}^{i,j} + \mathbf{c}_t^{i,j} \in \widetilde{\mathscr{O}}^{i,j} \right\} \\ =& \left\{ \mathbf{a}^{i,j} \in \mathbb{R}^{m_i+m_j} \middle| H_o B_t^{i,j}\mathbf{a}^{i,j} \leq \mathbf{h}_o - H_o\left(A_t^{i,j}\mathbf{s}^{i,j} + \mathbf{c}_t^{i,j}\right) \right\}.\end{aligned}$$

Given the ECO at time t, we could represent the ECO with horizon N, $\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j})$, as a union of N polyhedra

$$\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j}) = \bigcup_{t=1,2,\ldots,N} \mathcal{CO}^{i,j}(t, \mathbf{s}^{i,j}).$$

2.4.2 Linear Safety Constraints Computations

Given the current joint state $\mathbf{s}^{i,j}$ and the corresponding ECO $\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j})$ of a two agent linear system, we would like to compute the polyhedron $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ of the form (2.7) that makes (2.8) hold. Both agents i and j should be able to compute the corresponding $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ and $\mathcal{H}_j^{i,j}(\mathbf{s}^{i,j})$ respectively, without communications with each other. In this section, we propose an approach to compute the polyhedron $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ via convex optimizations. The returned $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ is a half-space, and the corresponding control constraint for each agent i is one single linear constraint.

Recall that $\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j})$ is a union of multiple polyhedra. In order to get $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ that makes (2.8b) hold, we introduce convex hulls. Let $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ be defined as the convex hull of $\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j}) \cap \mathscr{A}^{i,j}$,

$$\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j}) \triangleq \operatorname{conv}\left(\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j}) \cap \mathscr{A}^{i,j}\right),$$

which can be computed by both agents i and j. If both agents i and j can find $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ and $\mathcal{H}_j^{i,j}(\mathbf{s}^{i,j})$ respectively, such that $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ of form (2.7) satisfies

$$\mathcal{H}^{i,j}(\mathbf{s}^{i,j}) \cap \overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j}) = \emptyset, \tag{2.11}$$

then (2.8b) holds. In order to make (2.8a) also hold, we try to make $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ and $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ as close as possible while (2.11) holds. We achieve this by making $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ tangent to $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ via the following nonconvex optimization:

$$\mathbf{x}_0 = \underset{\mathbf{x}\in\mathbb{R}^{m_i+m_j}}{\arg\inf} \ \|\mathbf{x}\|_2 \tag{2.12a}$$

$$\text{s.t. } \bar{\mathbf{a}}^{i,j} + \mathbf{x} \in \partial\left(\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})\right) \tag{2.12b}$$

$$\bar{\mathbf{a}}^{i,j} + \mathbf{x} \notin \partial\left(\mathscr{A}^{i,j}\right), \tag{2.12c}$$

where $\bar{\mathbf{a}}^{i,j} = (\bar{\mathbf{a}}^i, \bar{\mathbf{a}}^j) \in \mathscr{A}^{i,j}$ is a joint control point that is known to both agents, and $\bar{\mathbf{a}}^i \in \mathscr{A}^i$, $\bar{\mathbf{a}}^j \in \mathscr{A}^j$. For smoothness of the generated trajectories, we make $\bar{\mathbf{a}}^{i,j} = \mathbf{a}_{t-1}^{i,j}$ at each time t, i.e., the last joint control.

The optimization (2.12) seeks the shortest vector $\mathbf{x}_0$ from the joint control point $\bar{\mathbf{a}}^{i,j}$ to the boundary of $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$. The constraint (2.12b) means that $\bar{\mathbf{a}}^{i,j} + \mathbf{x}_0$ should lie on the boundary of $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ and the constraint (2.12c) implies that $\bar{\mathbf{a}}^{i,j} + \mathbf{x}_0$ should not lie on the boundary of $\mathscr{A}^{i,j}$. Without constraint (2.12c), we may get $\mathbf{x}_0$ such that $\bar{\mathbf{a}}^{i,j} + \mathbf{x}_0 \in \partial\left(\mathscr{A}^{i,j}\right)$ and constraint (2.8a) would be violated.

Given $\mathbf{x}_0 = (\mathbf{x}_0^i, \mathbf{x}_0^j)$ with $\mathbf{x}_0^i \in \mathbb{R}^{m_i}$ and $\mathbf{x}_0^j \in \mathbb{R}^{m_j}$. The set $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ for agent i is defined as a half-space

$$\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j}) \triangleq \begin{cases} \left\{\mathbf{a}^i \in \mathbb{R}^{m_i} \middle| (\mathbf{a}^i - \bar{\mathbf{a}}^i - \mathbf{x}_0^i)^T \mathbf{x}_0^i \geq 0\right\} & \text{if } \bar{\mathbf{a}}^{i,j} \in \overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j}) \\ \left\{\mathbf{a}^i \in \mathbb{R}^{m_i} \middle| (\mathbf{a}^i - \bar{\mathbf{a}}^i - \mathbf{x}_0^i)^T \mathbf{x}_0^i \leq 0\right\} & \text{else} \end{cases}. \tag{2.13}$$

Similarly, agent j calculates the half-space $\mathcal{H}_j^{i,j}(\mathbf{s}^{i,j})$ via solving the same optimization (2.12). As illustrated in Fig. 2.2, the corresponding $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ is tangent to $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$.

Though (2.12) is a non-convex optimization in general, it can be converted into one convex optimization problem or a series of convex optimization problems. When $\bar{\mathbf{a}}^{i,j} \notin \overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$, the optimization (2.12) is equivalently to find the shortest vector

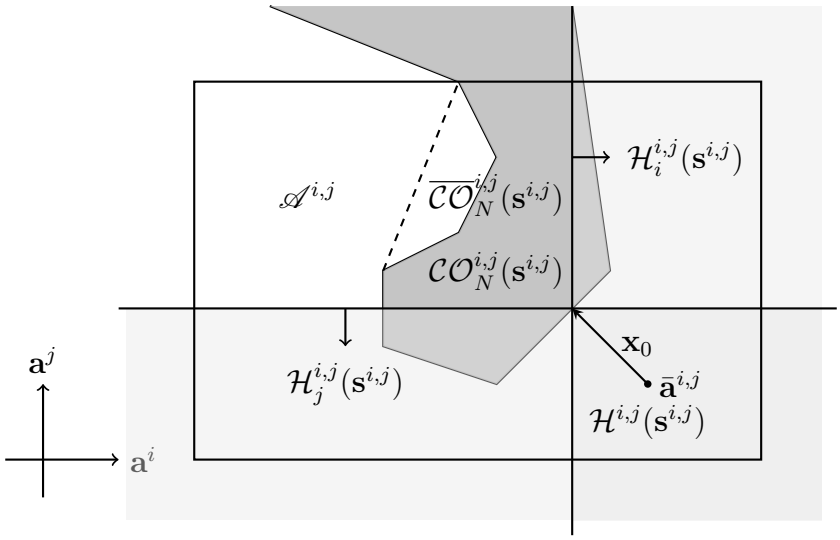


Figure 2.2: Linear Safety Constraints Computations for ECO. Given the solution $\mathbf{x}_0$ to the nonconvex optimization (2.12), we could construct the $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ via (2.13). The corresponding $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ is tangent to $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$.

from $\bar{\mathbf{a}}^{i,j}$ to $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$, which is equivalent to the following quadratic programming problem:

$$\mathbf{x}_0 = \underset{\mathbf{x}\in\mathbb{R}^{m_i+m_j}}{\arg\inf} \ \|\mathbf{x}\|_2 \tag{2.14a}$$

$$\text{s.t. } \bar{\mathbf{a}}^{i,j} + \mathbf{x} \in \overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j}). \tag{2.14b}$$

When $\bar{\mathbf{a}}^{i,j} \in \overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$, the optimization (2.12) can be converted into a series of convex optimization problem. We assume the set $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ could be written in the following form:

$$\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j}) = \left\{\mathbf{a}^{i,j} \in \mathbb{R}^{m_i+m_j} \middle| A^{eco}\mathbf{a}^{i,j} \leq \mathbf{b}^{eco}\right\} \cap \mathscr{A}^{i,j},$$

where $A^{eco} \in \mathbb{R}^{n_c\times(m_i+m_j)}$ and $\mathbf{b}^{eco} \in \mathbb{R}^{n_c}$ for some $n_c \in \mathbb{N}$. Let $A^{eco} = \left[\mathbf{a}_1^{eco}, \ldots, \mathbf{a}_{n_c}^{eco}\right]^T$ and $\mathbf{b}^{eco} = \left[b_1^{eco}, \ldots, b_{n_c}^{eco}\right]^T$, with $\mathbf{a}_p^{eco} \in \mathbb{R}^{m_i+m_j}$ and $b_p^{eco} \in \mathbb{R}$ for $p = 1, 2, \ldots, n_c$. The

p-th face of $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ is defined as

$$\left\{\mathbf{a}^{i,j} \in \mathbb{R}^{m_i+m_j} \middle| \left(\mathbf{a}_p^{eco}\right)^T \mathbf{a}^{i,j} = b_p^{eco}\right\} \cap \overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$$

for $p = 1, 2, \ldots, n_c$. Then the optimization (2.12) could be formulated as to find the shortest one among all shortest vectors from $\bar{\mathbf{a}}^{i,j}$ to each face of $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ that is not a subset of $\partial\left(\mathscr{A}^{i,j}\right)$. The shortest vector from $\bar{\mathbf{a}}^{i,j}$ to the p-th face of $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ could be found via the following convex optimization problem.

$$\mathbf{x}_p = \underset{w \in \mathbb{R}^{m_i+m_j}}{\arg\inf} \|\mathbf{x}\|_2 \tag{2.15a}$$

$$\text{s.t. } \bar{\mathbf{a}}^{i,j} + \mathbf{x} \in \overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j}) \tag{2.15b}$$

$$\left(\mathbf{a}_p^{eco}\right)^T \left(\bar{\mathbf{a}}^{i,j} + \mathbf{x}\right) = b_p^{eco}. \tag{2.15c}$$

The complete algorithm to compute the half-space $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ given ECO is summarized in Algorithm 2.

Algorithm 2 Linear Safety Constraints Computations

Input: The set $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$, the joint control space $\mathscr{A}^{i,j}$, a joint control point $\bar{\mathbf{a}}^{i,j} \in \mathscr{A}^{i,j}$

Output: The half-space $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$

Initialize $\mathbf{x}_0 \leftarrow$ None
if $\bar{\mathbf{a}}^{i,j} \notin \overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ **then**
    $\mathbf{x}_0 \leftarrow$ Solving optimization (2.14)
else
    for $p = 1, 2, 3, \ldots, n_c$ **do**
        if The p-th face of $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ is not a subset of $\partial\left(\mathscr{A}^{i,j}\right)$ **then**
            $\mathbf{x}_p \leftarrow$ Solving optimization (2.15)
            if $\mathbf{x}_0$ is None **or** $\|\mathbf{x}_0\|_2 > \|\mathbf{x}_p\|_2$ **then**
                $\mathbf{x}_0 \leftarrow \mathbf{x}_p$
            end if
        end if
    end for
end if
Compute $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ via (2.13)

Algorithm 3 Collision Avoidance Algorithm for Heterogeneous Linear Systems

Input: Initial time $t = 0$, the initial state $\mathbf{s}_0^i = \mathbf{s}_{initial}^i$, a control reference trajectory $\tilde{\mathbf{a}}_{(\cdot)}^i : \mathbb{N}_0 \to \mathscr{A}^i$ for each agent $i \in \mathcal{M}$, and the ECO time horizon $N \in \mathbb{N}$

while True **do**
  for $i \in \mathcal{M}$ **do**
    for $j \in \mathcal{M}$ and $j \neq i$ **do**
      Observe $\mathbf{s}_t^{i,j} = \mathbf{s}^{i,j}$
      Compute the set $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$
      Compute $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ via Algorithm 2
    end for
    Compute control $\mathbf{a}^i$ via the optimization (2.9)
    Apply control $\mathbf{a}^i$ to agent i: $\mathbf{a}_t^i = \mathbf{a}^i$
  end for
  $t \leftarrow t + 1$
end while

2.4.3 Collision Avoidance for Heterogeneous Linear Systems

For each agent $i \in \mathcal{M}$ with current state $\mathbf{s}^{i,j}$ at each time t, it first compute the half-space $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ with respect to each other agent $j \in \mathcal{M}$ and $j \neq i$. The linear control constraints induced by $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ provides a strict safety guarantee for each agent i. Then it computes its action $\mathbf{a}^i$ to avoid collisions with all other agents while trying to follow the reference control by solving the convex optimization (2.9) with state-dependent constraints. The complete collision avoidance algorithm for heterogeneous linear systems is summarized in Algorithm 3.

2.5 Uncontrolled Agents and Nonlinear Dynamics

Our previous discussions of the collision avoidance algorithm are based on the assumption that all agents in the workspace have linear dynamics and follow the same collision avoidance algorithm. However, this may not be true for some real-world applications. Linear models may not be accurate enough to describe the dynamics of

real robots, and there may exist uncontrolled agents in the environment that do not follow the proposed framework. In this section, we extend the proposed algorithm to heterogeneous nonlinear systems and multi-agent systems with uncontrolled agents.

2.5.1 Nonlinear Dynamics

To apply the proposed algorithm to an nonlinear agent, we approximate nonlinear models of the form (2.1) with linear ones of the form (2.10). We linearize nonlinear models via the first order Taylor expansion around the current state $\mathbf{s}_t^i$ and the last control $\mathbf{a}_{t-1}^i$ at each time t. We let A^i, B^i, $\mathbf{c}^i$, D^i and $\mathbf{e}^i$ in (2.10) be given as

$$\begin{aligned}
A^i &\triangleq \left.\frac{\partial f^i(\mathbf{s}^i, \mathbf{a}^i)}{\partial \mathbf{s}^i}\right|_{\mathbf{s}^i=\mathbf{s}_t^i, \mathbf{a}^i=\mathbf{a}_{t-1}^i}, \\
B^i &\triangleq \left.\frac{\partial f^i(\mathbf{s}^i, \mathbf{a}^i)}{\partial \mathbf{a}^i}\right|_{\mathbf{s}^i=\mathbf{s}_t^i, \mathbf{a}^i=\mathbf{a}_{t-1}^i}, \\
\mathbf{c}^i &\triangleq f^i(\mathbf{s}_t^i, \mathbf{a}_{t-1}^i) - A^i\mathbf{s}_t^i - B^i\mathbf{a}_{t-1}^i, \\
D^i &\triangleq \left.\frac{\partial g^i(\mathbf{s}^i)}{\partial \mathbf{s}^i}\right|_{\mathbf{s}^i=\mathbf{s}_t^i, \mathbf{a}^i=\mathbf{a}_{t-1}^i}, \\
\mathbf{e}^i &\triangleq g^i(\mathbf{s}_t^i) - D^i\mathbf{s}_t^i
\end{aligned}$$

Given the linearized models of form (2.10), we could directly apply Algorithm 3 to handle scenarios with nonlinear dynamics. To compensate for the linearization error, we slightly enlarge the radius of each nonlinear agent during the construction of ECO.

2.5.2 Uncontrolled Agents and Obstacles

To apply the collision avoidance framework proposed in Section 2.3.2 to scenarios with uncontrolled agents and obstacles, we would like to construct state-dependent control constraints to avoid collisions for each controlled agent. First we consider the collision avoidance problem between a controlled agent i and an uncontrolled agent j

with current joint state $\mathbf{s}^{i,j}$. We would like to find a half-space $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j}) \subseteq \mathbb{R}^{m_i}$, such that no collision would occur if agent i apply any constant control $\mathbf{a}^i \in \mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ while agent j apply arbitrary constant control $\mathbf{a}^j \in \mathscr{A}^j$ in the following N steps. Though the uncontrolled agent j may not take constant control, the half-space $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ at least provides a safety guarantee for one time step. When extended to multi-agent systems with both controlled and uncontrolled agents, each controlled agent $i \in \mathcal{M}$ could still find a safe control via the convex optimization (2.9) with state-dependent action constraints.

To construct such a half-space $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$, we define the *Projected Extended Control Obstacle (PECO) with horizon* $N \geq 0$ as

$$\mathcal{PO}_N^{i,j}(\mathbf{s}^{i,j}) \triangleq \left\{ \mathbf{a}^i \in \mathbb{R}^{m_i} \middle| \exists t = 1, 2, \ldots, N, \exists \mathbf{a}^j \in \mathscr{A}^j, \text{s.t. } \xi^{i,j}\left(t; \mathbf{s}^{i,j}, (\mathbf{a}^i, \mathbf{a}^j)\right) \in \widetilde{\mathscr{O}}^{i,j} \right\},$$

which is the set of constant control $\mathbf{a}^i \in \mathbb{R}^{m_i}$ that could result in collisions if agent j adopt some constant control $\mathbf{a}^j \in \mathscr{A}^j$. The set $\mathcal{PO}_N^{i,j}(\mathbf{s}^{i,j})$ is called PECO because it is the projection of the set $\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j}) \cap (R^{m_i} \times \mathscr{A}^j)$ from $\mathbb{R}^{m_i+m_j}$ to $\mathbb{R}^{m_i}$. We further assume the control space $\mathscr{A}^j$ for agent j is a polyhedron. Like $\mathcal{CO}_N^{i,j}(\mathbf{s}^{i,j})$, for agents i and j with linear dynamics, $\mathcal{PO}_N^{i,j}(\mathbf{s}^{i,j})$ is a union of multiple polyhedra.

We denote convex hull of $\mathcal{PO}_N^{i,j}(\mathbf{s}^{i,j}) \cap \mathscr{A}^i$ by $\overline{\mathcal{PO}}_N^{i,j}(\mathbf{s}^{i,j})$. Given $\overline{\mathcal{PO}}_N^{i,j}(\mathbf{s}^{i,j})$, we can find a half-space $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ such that $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j}) \cap \mathcal{PO}_N^{i,j}(\mathbf{s}^{i,j}) = \emptyset$ via the following non-convex optimization:

$$\mathbf{x}_0 = \arg\inf_{\mathbf{x} \in \mathbb{R}^{m_i}} \|\mathbf{x}\|_2 \tag{2.16a}$$

$$\text{s.t. } \bar{\mathbf{a}}^i + \mathbf{x} \in \partial\left(\overline{\mathcal{PO}}_N^{i,j}(\mathbf{s}^{i,j})\right) \tag{2.16b}$$

$$\bar{\mathbf{a}}^i + \mathbf{x} \notin \partial\left(\mathscr{A}^i\right). \tag{2.16c}$$

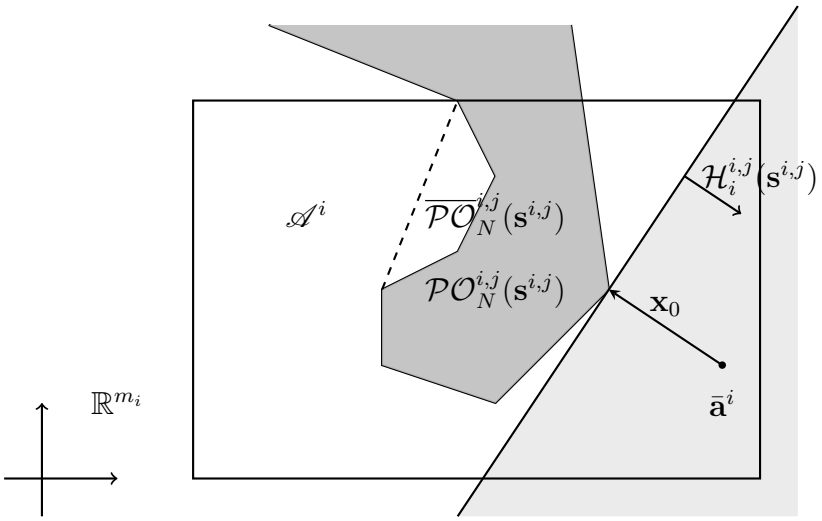


Figure 2.3: Linear Constraints Decomposition for PECO. Given the solution $\mathbf{x}_0$ to the nonconvex optimization (2.16), we construct the $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ via (2.17). The corresponding $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ is tangent to $\overline{\mathcal{PO}}_N^{i,j}(\mathbf{s}^{i,j})$.

Here $\bar{\mathbf{a}}^i = \mathbf{a}_{t-1}^i$ is the last control of agent i. We seeks $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ that are the closet half-space to the last control $\mathbf{a}_{t-1}^i$ to make the generated trajectory smooth, while making sure it provides strict safety guarantee. The optimization (2.16) is almost the same as (2.12), and we could adopt Algorithm 2 to solve it by replacing $\overline{\mathcal{CO}}_N^{i,j}(\mathbf{s}^{i,j})$ with $\overline{\mathcal{PO}}_N^{i,j}(\mathbf{s}^{i,j})$, and $\mathscr{A}^{i,j}$ with $\mathscr{A}^i$.

Given $\mathbf{x}_0$, the half-space $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ is given as

$$\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j}) \triangleq \begin{cases} \{\mathbf{a}^i \in \mathbb{R}^{m_i} | (\mathbf{a}^i - \bar{\mathbf{a}}^i - \mathbf{x}_0) \cdot \mathbf{x}_0 \geq 0\} & \text{if } \bar{\mathbf{a}}^i \in \overline{\mathcal{PO}}_N^{i,j}(\mathbf{s}^{i,j}) \\ \{\mathbf{a}^i \in \mathbb{R}^{m_i} | (\mathbf{a}^i - \bar{\mathbf{a}}^i - \mathbf{x}_0) \cdot \mathbf{x}_0 \leq 0\} & \text{else} \end{cases}. \tag{2.17}$$

As illustrated in Fig. 2.3, the half-space $\mathcal{H}_i^{i,j}(\mathbf{s}^{i,j})$ is tangent to $\mathcal{PO}_N^{i,j}(\mathbf{s}^{i,j})$ and provides a set of safe control for agent i to avoid collisions with uncontrolled agent j.

For a static or moving obstacle with constant velocity, we could approximate it by one or several uncontrolled agents with linear dynamics and empty control space $\mathscr{A}^j$. For multi-agent scenarios consisting of controlled and uncontrolled agents as well

as static or moving obstacles, each controlled agent i could still adopt the proposed algorithm to avoid collisions. The half-space $\mathcal{H}^{i,j}(\mathbf{s}^{i,j})$ with respect to each other agent j is a state-dependent action constraints that guarantee safety.

2.6 Experiments and Results

We illustrate the performance of our algorithm via several challenging multi-agent interactive scenarios with different robot dynamics.

2.6.1 Simulation Setup

The proposed collision avoidance algorithm is validated via multi-agent systems with up to four different dynamics: single integrators, double integrators, unicycles, and kinematic bicycles.

The dynamics of the single integrators is given as

$$\dot{p}_x = v_x \tag{2.18a}$$

$$\dot{p}_y = v_y \tag{2.18b}$$

where state $\mathbf{s} = [p_x, p_y]^T$ consists of coordinates of the agent position, and control $\mathbf{a} = [v_x, v_y]^T$ is the velocity.

For the double integrators, their dynamics is given as

$$\dot{p}_x = v_x \tag{2.19a}$$

$$\dot{p}_y = v_y \tag{2.19b}$$

$$\dot{v}_x = a_x \tag{2.19c}$$

$$\dot{v}_y = a_y \tag{2.19d}$$

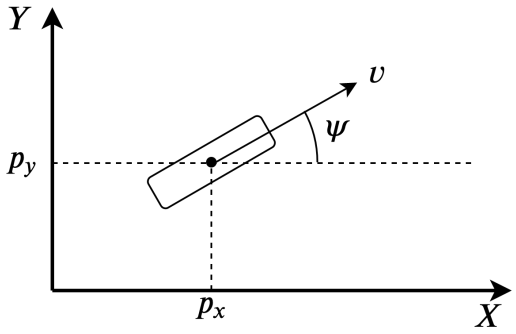


Figure 2.4: Unicycle Model

where state $\mathbf{s} = [p_x, p_y, v_x, v_y]^T$ consists of the position and the velocity, and its control $\mathbf{a} = [a_x, a_y]^T$ is acceleration. As illustrated in Fig. 2.4, the dynamics of a unicycle is defined as

$$\dot{p}_x = v\cos(\psi) \tag{2.20a}$$

$$\dot{p}_y = v\sin(\psi) \tag{2.20b}$$

$$\dot{\psi} = w \tag{2.20c}$$

$$\dot{v} = a \tag{2.20d}$$

where state $\mathbf{s} = [p_x, p_y, \psi, v]^T$ consists of position p_x and p_y, inertial heading ψ and speed v. The control $\mathbf{a} = [w, a]^T$ consists of yaw rate w and acceleration a.

The dynamics of kinematic bicycle models is illustrated in Fig 2.5, it is defined as

$$\dot{p}_x = v\cos(\psi + \beta) \tag{2.21a}$$

$$\dot{p}_y = v\sin(\psi + \beta) \tag{2.21b}$$

$$\dot{\psi} = \frac{v}{l_r}\sin(\beta) \tag{2.21c}$$

$$\dot{v} = a \tag{2.21d}$$

$$\beta = \tan^{-1}\left(\frac{l_r}{l_f + l_r}\tan\delta_f\right) \tag{2.21e}$$

where state $\mathbf{s} = [p_x, p_y, \psi, v]^T$ consists of position p_x and p_y, inertial heading ψ and speed v. We use β to represent the angle between velocity and the longitudinal axis of the robot, and use l_f, l_r to represent the distance from the centroid to the front and rear axles respectively. The control $\mathbf{a} = [\delta_f, a]^T$ consists of the front steering angle δ_f and acceleration a.

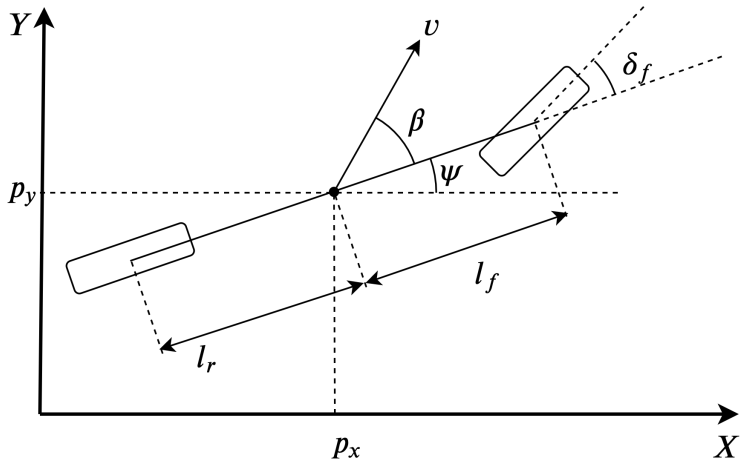


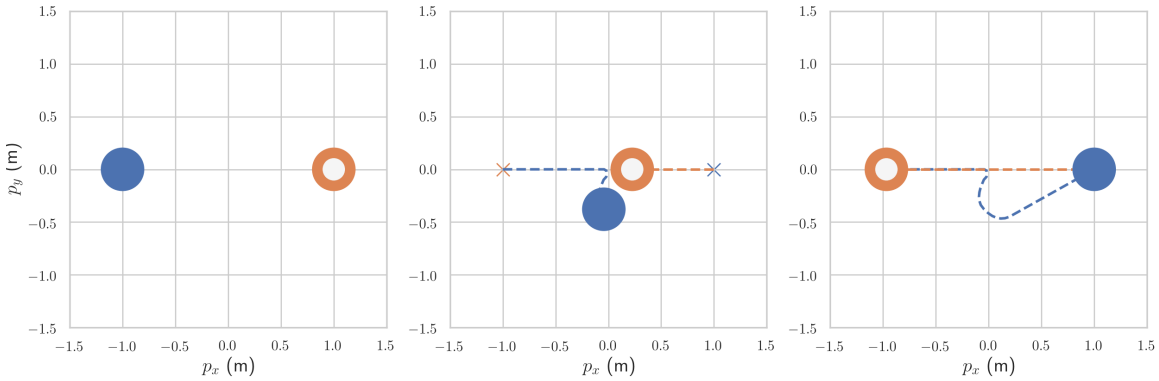
Figure 2.5: Kinematic Bicycle Model

To get the corresponding discrete time models, we apply a constant control $\mathbf{a}_i$ within a time step $\Delta t = 0.04\ s$ for each agent i. The physical control space $\mathscr{A}^i$ for each agent i is a polyhedron. As for the planning module for each agent, we use a simple PD controller that drives the corresponding agent to a goal position with zero velocity. The dangerous set $\widetilde{\mathscr{O}}^{i,j}$ for each pair of agents i and j is a polyhedron externally tangent to the collision set $\mathscr{O}^{i,j}$.

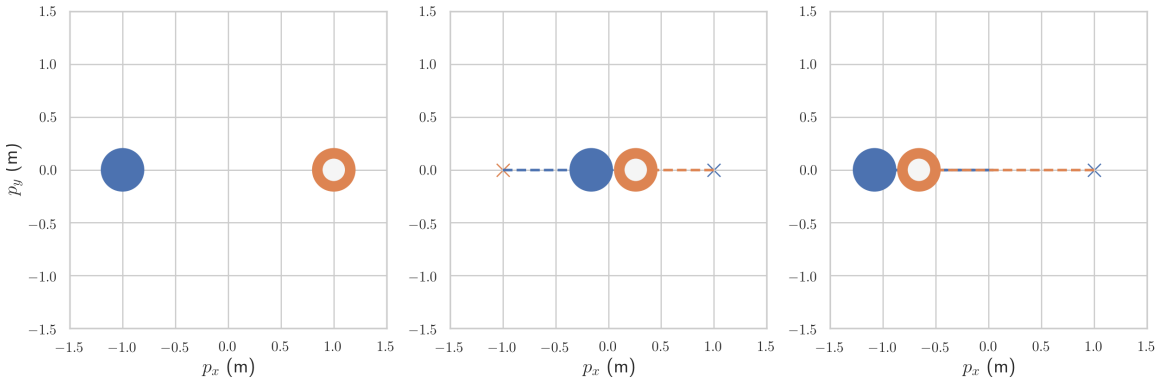
2.6.2 Simulation Results

We first test our approach in simple scenarios consisted of heterogeneous linear systems. As shown in Fig 2.6, we compare the proposed approach with ORCA in a simulation where one single integrator and one double integrator swap their positions. To apply ORCA algorithm to the double integrator, we approximate it by a single integrator and introduce a PD controller to track velocities from ORCA. Though this simulation is relatively simple, the planning modules are designed to drive two agents heading toward each other, and deadlock could easily occur. As illustrated in Fig 2.6a, the proposed approach avoids collision successfully, and both agents reach their goal positions. On the contrary, as shown in Fig 2.6b, deadlock occurs when ORCA is applied. Though there is no collision, the double integrator pushes the single integrator away from its goal position.

Then we demonstrate our approach with more complex tasks. As shown in 2.7, we include a simulation that contains two single integrators and two double integrators. The goal position of each agent is symmetric with the initial position about the origin. The generated trajectories are smooth and collision-free, and all agents reach their goal positions.



(a) Trajectories generated by the proposed approach.



(b) Trajectories generated by the ORCA.

Figure 2.6: A simulation where one single integrator and one double integrator swapping positions via the proposed approach. The single integrator is denoted by a solid circle and the double integrator is denoted by a ring. Their goal positions are denoted by cross markers with the same colors as the agents themselves. The generated trajectories are marked by dashed lines with corresponding colors.

We also test the proposed approach with more complex tasks that includes nonlinear dynamics: unicycle models and kinematic bicycle models. As shown in Fig. 2.8 and 2.9, the unicycles and bicycles are denoted by circles with black and white arrows inside, respectively. The directions of arrows represent the initial headings

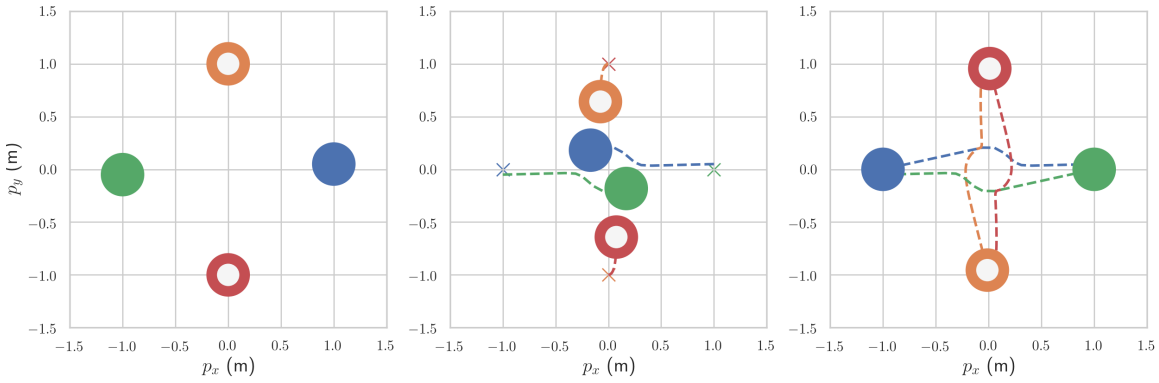


Figure 2.7: A simulation that contains two single integrators and two double integrators.

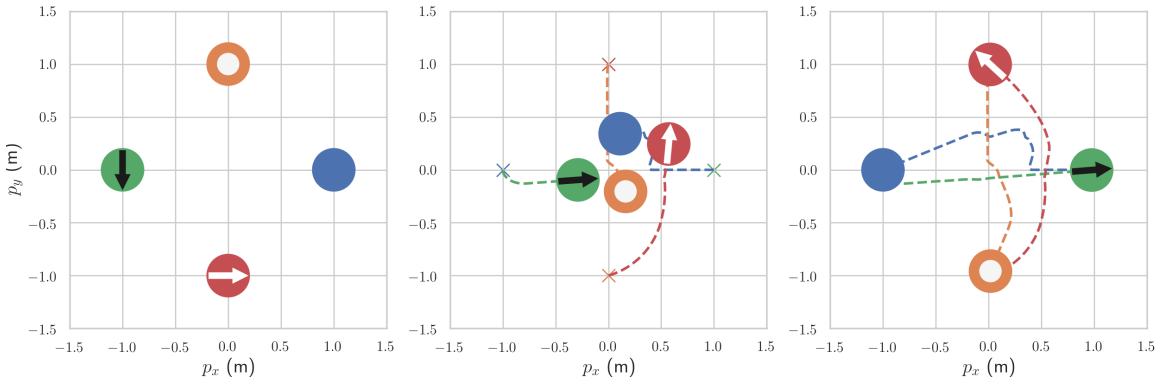


Figure 2.8: A simulation that contains four agents: a single integrator (solid circle), a double integrator (ring), a unicycle (circle with black arrow), and a bicycle (circle with white arrow).

of agents. The simulation illustrated by Fig. 2.8 includes four agents with different dynamics, and the simulation illustrated by Fig. 2.9 contains eight agents with four types of dynamics. The goal position of each agent is symmetric with respect to

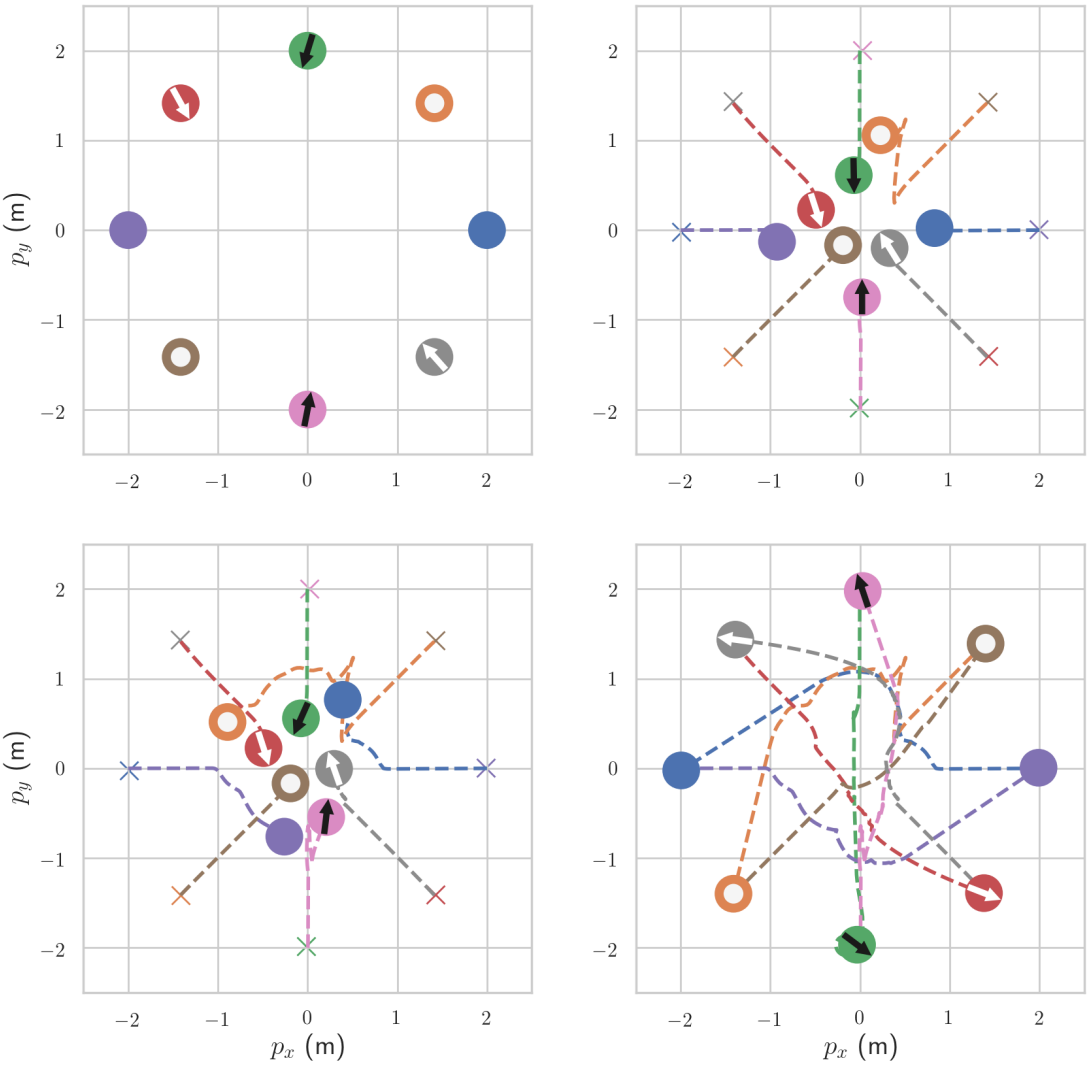


Figure 2.9: A simulation that contains eight agents with four types of dynamics.

the initial position about the origin. Though such symmetric settings could easily lead to collisions and congestion, the proposed algorithm generates safe and smooth trajectories that drive all agents to their goal positions.

Furthermore, we demonstrate our problem in a scenario that include an uncontrolled agent. As shown in Fig. 2.10, the black circle denotes an uncontrolled single

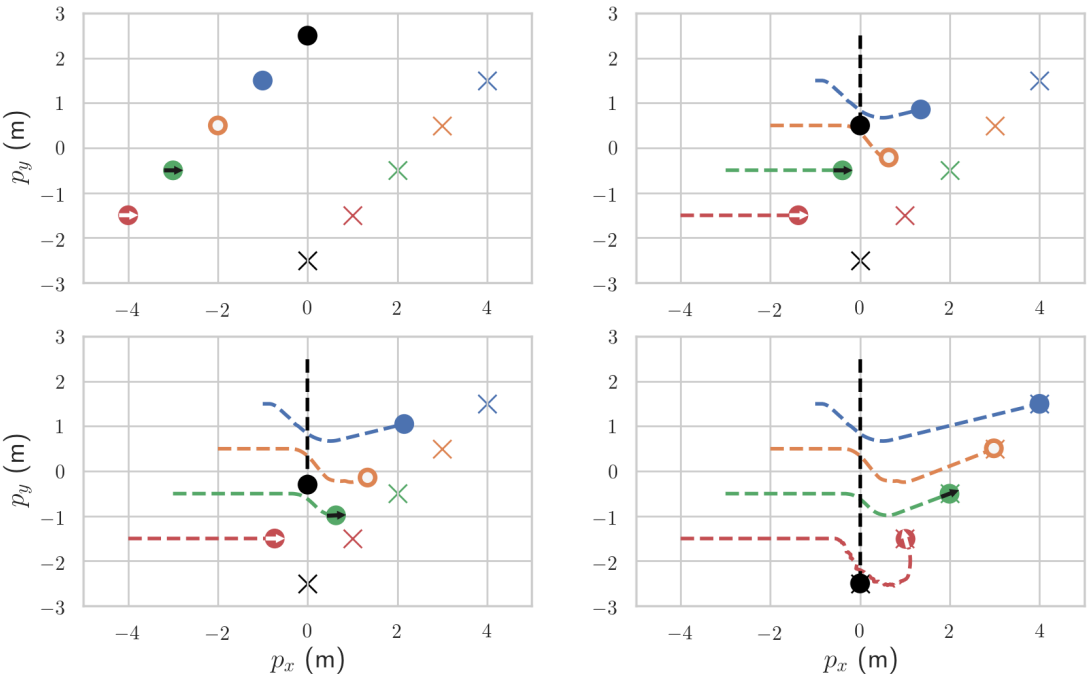


Figure 2.10: A simulation that contains an uncontrolled agent. The black circle represents an uncontrolled agent that moves toward its goal position directly.

integrator that moves toward its goal position directly. Each other agent need to interact with the uncontrolled one to avoid collisions. It is clear from the figure that all other agents bypass the uncontrolled one without collisions and reach their goals successfully with smooth trajectories.

2.7 Conclusions

This chapter studies the decentralized collision avoidance algorithm for heterogeneous systems. We propose a decentralized multi-agent collision avoidance algorithm for heterogeneous systems. The algorithm is based on the concept of the extended control obstacles (ECO), which is defined as a set of constant joint control that could

cause collisions for a two agents system. The introduction of ECO provides pairwise linear control constraints that guarantee safety strictly for heterogeneous linear systems, which are state-dependent. The collision avoidance problem for each agent is formulated as a convex optimization. The linear constraints can be easily obtained via one convex optimization or a series of convex optimization. The proposed approach can be easily extended and handle realistic scenarios that include nonlinear agents, uncontrolled agents, and obstacles. The performance of the proposed approach is demonstrated with realistic simulations.

Chapter 3: Multi-agent Collision Avoidance for General Nonlinear Agents via Reinforcement Learning

3.1 Introduction

In this chapter, we study the decentralized multi-agent collision avoidance for general nonlinear systems. One promising direction is to apply RL algorithms for collision avoidance, since many popular model-free RL algorithms have been developed [40, 41, 42, 43]. But there are several challenges to apply RL for decentralized multi-agent collision avoidance. First, the environment is non-stationary for a decentralized problem. Since other agents are considered as part of the environment and their control policy would evolve in the process of training, the environment is changing from the perspective of each single agent. Thus, such RL-based collision avoidance algorithms are hard to converge in the training process. Most existing RL-based algorithms are restricted to discrete action spaces, and they assume the velocities of each agent can be directly controlled. Second, the number of agents involved in the planning period may be dynamically changing, which makes it hard to represent the state for RL algorithms. Third, no safety guarantee is provided by RL-based approaches. They only consider safety via the design of reward functions, and there is no theoretic safety guarantee for collision avoidance.

To address the aforementioned issues, we propose a fast multi-agent collision avoidance algorithm for general nonlinear agents with continuous action space. We consider realistic scenarios where each agent observes only the positions and velocities of nearby agents. We first decompose the multi-agent scenario and solve a two agents collision avoidance problem using reinforcement learning (RL). Since there are only one other agent from the perspective of each agent in the training process, the evolvement of environments is much slower. Thus the RL algorithm can converge in a faster way, and can be applied to more complicated dynamics. Since there is just two agents in the training process, the varying number of agents is no longer an issue. When extending the trained policy to a multi-agent problem, safety is ensured by introducing linear constraints from the optimal reciprocal collision avoidance (ORCA). The overall collision avoidance action could be found through a simple convex optimization with state-dependent control constraints in real-time. Realistic simulations based on nonlinear bicycle agent models are performed with various challenging scenarios, indicating a competitive performance of the proposed method in avoiding collisions, congestion and deadlock with smooth trajectories. The main contributions of this work are summarized below:

- **General Nonlinear Agents:** Unlike other RL-based approaches that assume velocities of agents could be directly controlled, or other reciprocal collision avoidance algorithms that need specific models and feedback controllers, our approach works for general nonlinear systems with continuous action space.

- **Improved Safety:** Different from other RL-based approaches which only consider safety in reward function design, our approach incorporates safety bounds

systematically by introducing linear constraints obtained from ORCA. In Section 3.4.3, we demonstrate the safety of our algorithm with complex and challenging tasks.

- **Lightweight Structure:** Through decomposing the problem into a two agents collision avoidance problem, we significantly simplify the RL training task. The learning process of our approach is much faster and the neural network is much smaller when compared with other RL-based approaches.

3.2 Problem Formulation

We consider a discrete time heterogeneous multi-agent system consisting of M agents, indexed by $\mathcal{M} \triangleq \{1, \ldots, M\}$. It is assumed that each agent could be approximated as a disc or sphere with the same radius r. Each agent $i \in \mathcal{M}$ is modeled by the following nonlinear dynamics

$$\mathbf{s}_{t+1}^i = f(\mathbf{s}_t^i, \mathbf{a}_t^i), \tag{3.1a}$$

$$\mathbf{y}_t^i = h(\mathbf{s}_t^i). \tag{3.1b}$$

Here $\mathbf{s}_t^i \in \mathscr{S} \subseteq \mathbb{R}^n$ and $\mathbf{a}_t^i \in \mathscr{A} \subseteq \mathbb{R}^m$ represent the state, control of agent i at time $t \geq 0$, where $\mathscr{S}$ and $\mathscr{A}$ represent convex constraints in state space and action space, respectively. The output of the system is $\mathbf{y}_t^i = (\mathbf{p}_t^i, \mathbf{v}_t^i)$, which consists of the position $\mathbf{p}_t^i \in \mathbb{R}^d$ and velocity $\mathbf{v}_t^i \in \mathbb{R}^d$ of agent i, where $d = 2$ or 3 typically. The function $f(\cdot, \cdot) : \mathscr{S} \times \mathscr{A} \to \mathscr{S}$ is the state transition function and $g(\cdot) : \mathscr{S} \to \mathbb{R}^{2d}$ is the output function. For example, if the state $\mathbf{s}_t^i$ comprises the position $\mathbf{p}_t^i$ and the velocity $\mathbf{v}_t^i$ and some other information, then another agent $j \neq i$ observes only the output of the agent i, i.e., the position and the velocity. Given M total agents, for agent i, it

could observe output of all other agents at each time t:

$$\mathbf{y}_t^{-i} \triangleq \left(\mathbf{y}_t^1, \mathbf{y}_t^2, \ldots, \mathbf{y}_t^{i-1}, \mathbf{y}_t^{i+1}, \ldots, \mathbf{y}_t^M\right).$$

It is assumed that each agent has a goal state $\tilde{\mathbf{s}}^i$, which is the state they would like to reach. Each agent knows its own state and goal state, and can observe output from other agents. All the information agent i has at time t could be represented as observation $\mathbf{o}_t^i$ which is defined as

$$\mathbf{o}_t^i \triangleq \left(\mathbf{s}_t^i, \tilde{\mathbf{s}}^i, \mathbf{y}_t^{-i}\right).$$

Given the observation, we would like to compute a control policy π that maps the observation $\mathbf{o}_t^i$ to its action $\mathbf{a}_t^i$ for all agents. We parameterize the policy using variable $\theta \in \mathbb{R}^n$, that is

$$\mathbf{a}_t^i = \pi(\mathbf{o}_t^i, \theta) \in \mathscr{A}. \tag{3.2}$$

Given the policy $\pi(\cdot, \theta)$, each agent should reach their goal state in the shortest time without collisions. Then the collision avoidance problem could be formulated as a multi-agent sequential decision making problem, given by

$$\min_{\theta} \ \frac{1}{M} \sum_{i=1}^{M} \sum_{t=0}^{\infty} \gamma^t R(\mathbf{o}_t^i) \tag{3.3a}$$

$$\text{s.t. } \mathbf{s}_{t+1}^i = f(\mathbf{s}_t^i, \mathbf{a}_t^i) \ \forall i \in \mathcal{M}, \ \forall t \in \mathbb{N} \tag{3.3b}$$

$$(\mathbf{p}_t^i, \mathbf{v}_t^i) = h(\mathbf{s}_t^i) \tag{3.3c}$$

$$\mathbf{a}_t^i = \pi(\mathbf{o}_t^i, \theta) \in \mathscr{A} \tag{3.3d}$$

$$\|\mathbf{p}_t^i - \mathbf{p}_t^j\| > 2r, \ \forall i, j \in \mathcal{M}, \ i \neq j, \ \forall t \in \mathbb{N} \tag{3.3e}$$

$$\mathbf{s}_0^i = \mathbf{s}_{initial}^i, \ \forall i \in \mathcal{M}, \tag{3.3f}$$

where $R(\cdot)$ is a scalar reward function, and $\gamma \in (0, 1]$ is the discount factor. The reward function $R(\cdot)$ awards the agent for approaching the goal state and penalizes for collisions with other agents, it will be further explained in Section 3.3.1. Equation (3.3b) and (3.3c) are due to nonlinear dynamics. Equation (3.3d) means that all agents should follow the same policy, since all agents share the same dynamics and are equivalent in this multi-agent collision avoidance problem. Inequalities (3.3e) encode collision avoidance conditions. Each agent i has a given initial state $\mathbf{s}^i_{initial}$, which is expressed as the equality constraint (3.3f).

It is rather challenging to directly solve (3.3) in a centralized way. The optimization is highly non-convex due to nonlinear equality constraints (3.3b) and (3.3c), and non-convex inequality constraints (3.3e). Besides, each agent only has partial information of surrounding agents. Instead of solving (3.3) directly, we handle the problem by first solving a two agents collision avoidance problem via RL, and then transfer the multi-agent collision avoidance problem into a simple convex optimization problem with state-dependent safety constraints from ORCA.

3.3 Approach

This section presents an algorithm to solve the multi-agent collision avoidance problem (3.3) which combines RL and ORCA. To reduce online computation and make the training fast, we first solve a two agents collision avoidance problem using deep RL. Many existing RL-based collision avoidance algorithms assume that agents can directly control their velocities and require discretization of the action space. Thus, they are only applicable to simple dynamics and can lead to unnatural trajectories. Different from those studies, we formulate RL with continuous action space

for general nonlinear agents. Our RL-based approach is faster than other RL-based algorithms with a lightweight policy neural network. Given the trained policy, we apply it to multi-agent problems by combing pairwise actions introduced by different nearby agents. To avoid collisions, ORCA is introduced and we selects action by solving a simple convex optimization problem with linear safety constraints. The framework of our algorithm is shown in Fig. 3.1.

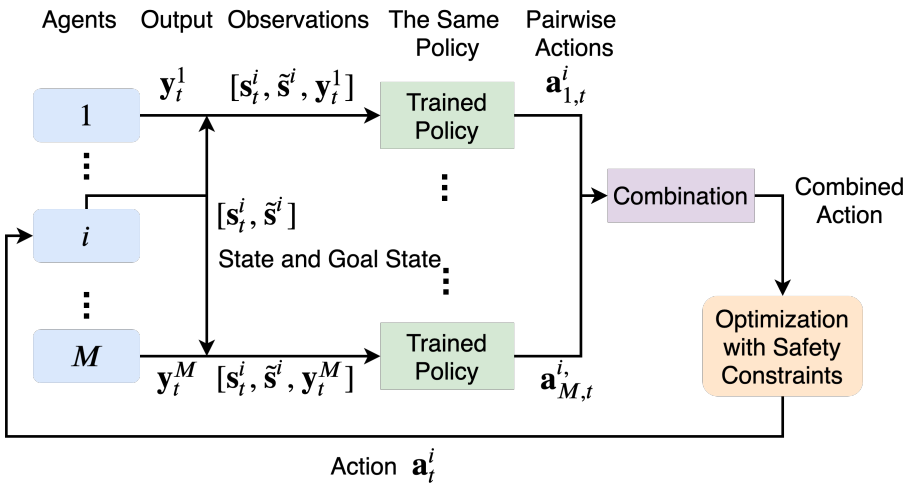


Figure 3.1: The Algorithm Framework for RL-based Multi-agent Collision Avoidance Algorithm

3.3.1 Two Agents Collision Avoidance via Reinforcement Learning

For general policy-based RL algorithm, the policy is usually represented as a neural network, with the state as input and the action or parameters of the action as output. For the two agents collision avoidance problem with agents i and j, both of them have their observations. With slight abuse of notation, we denote the observations of agent

i and j by $\mathbf{o}_{j,t}^i$ and $\mathbf{o}_{i,t}^j$, i.e.

$$\mathbf{o}_{j,t}^i = \left(\mathbf{s}_t^i, \tilde{\mathbf{s}}^i, \mathbf{y}_t^j\right),$$

$$\mathbf{o}_{i,t}^j = \left(\mathbf{s}_t^j, \tilde{s}^j, \mathbf{y}_t^i\right).$$

State space in standard RL is now the observation space. We use a deterministic policy, where the input of the neural network is the observation and output is the action, that is $\mathbf{a}_{j,t}^i = \bar{\pi}(\mathbf{o}_{j,t}^i, \bar{\theta})$. Here we use $\bar{\pi}(\cdot,\cdot)$ to denote the policy neural network and $\bar{\theta}$ to denote parameters of the network. Since there are only two agents, the input dimension of the neural network is relatively small and the training can be done efficiently.

The objective for each agent is to achieve the goal state in the shortest time while avoiding collision with each other. The reward function is designed as follows:

$$R(\mathbf{o}_{j,t}^i) = (r_g)_t^i + (r_p)_t^i + (r_c)_t^i \tag{3.4}$$

For agent i at time t, the reward function is designed as the sum of three terms. The first term $(r_g)_t^i$ is designed to encourage the agent for reaching the goal state,

$$(r_g)_t^i = \begin{cases} r_{arrival} & \text{if } \left\|\mathbf{s}_{t+1}^i - \tilde{\mathbf{s}}^i\right\|_2 < d_{arrival} \\ 0 & \text{otherwise} \end{cases}, \tag{3.5}$$

where $d_{arrival}$ represents a small, positive threshold to determine whether agent i has reached the goal state $\tilde{\mathbf{s}}^i$, and $r_{arrival}$ is a positive reward. The second term $(r_p)_t^i$ is designed to award the agent approaching the goal state.

$$(r_p)_t^i = -\frac{\left\|\mathbf{s}_{t+1}^i - \tilde{\mathbf{s}}^i\right\|_2}{\left\|\mathbf{s}_0^i - \tilde{\mathbf{s}}^i\right\|_2}. \tag{3.6}$$

As the agent gets closer to the goal state, the reward becomes larger. The last term $(r_c)_t^i$ is designed to penalize collisions with other agents,

$$(r_c)_t^i = \begin{cases} r_{collision} & \text{if } \left\|\mathbf{p}_{t+1}^i - \mathbf{p}_{t+1}^j\right\|_2 < 2r,\ \forall j \neq i \\ 0 & \text{otherwise} \end{cases}, \tag{3.7}$$

where $r_{collision}$ is a negative penalty for collisions.

Since all agents are homogeneous and cooperative, it is natural to assume that both agents follow the same policy. Therefore, when updating the neural network, the difference of parameters $d\bar{\theta}$ comes from the observation, action and reward trajectories of both agents. Though different from standard single-agent RL, this modification can be applied to nearly any deep RL framework. In our simulation, we use Evolution Strategy (ES) as our training method.

3.3.2 Multi-agent Collision Avoidance with Improved Safety

For the multi-agent collision avoidance problem, each agent has more than one neighbor agent. Given the trained policy $\bar{\pi}(\cdot, \bar{\theta})$ from two agents collision avoidance problem, each nearby agent j of agent i would introduce an action $\mathbf{a}_{j,t}^i$ for collision avoidance. Multiple actions are combined into one action. Then we introduce state-dependent linear constraints from the ORCA and handle the multi-agent collision avoidance problem by solving a simple convex optimization.

For one agent i, each neighbor agent j results in an action from the trained policy, $\mathbf{a}_{j,t}^i \triangleq \bar{\pi}(\mathbf{o}_{j,t}^i, \bar{\theta})$, $j \neq i$ and $i, j = 1, \ldots, M$. Intuitively, the actions introduced by closer neighbors and approaching neighbors are more important. Therefore, we combine all actions with distance and velocity-based weights:

$$\mathbf{a}_t^{i,comb} = \frac{\sum_{j=1, j\neq i}^{M} \mathbf{w}_{j,t}^i \mathbf{a}_{j,t}^i}{\sum_{j=1, j\neq i}^{M} \mathbf{w}_{j,t}^i}, \tag{3.8}$$

$$\mathbf{w}_{j,t}^i = \frac{e^{-\alpha d_{j,t}^i}}{d_{j,t}^i}. \tag{3.9}$$

Here $\mathbf{w}_{j,t}^i$ represents the weight determined by the pseudo-distance $d_{j,t}^i$, and α is a scaling parameter. Our design of weight $\mathbf{w}_{j,t}^i$ is inspired by the artificial potential

function proposed by [33]. The weight $\mathbf{w}_{j,t}^{i}$ becomes infinite when $d_{j,t}^{i} = 0$. The pseudo-distance $d_{j,t}^{i}$ is determined by the relative position of agent j to agent i and velocity of agent j. If agent i is behind the neighbor agent j, $d_{j,t}^{i}$ is the Euclidean distance, otherwise it is defined as adjusted distance that scaled along the velocity of agent j:

$$d_{j,t}^{i} = \begin{cases} \left\|\mathbf{p}_{i,t}^{j}\right\|_2 & \text{if } (\mathbf{p}_{i,t}^{j})^T \mathbf{v}_t^j > 0 \\ \sqrt{\frac{\left\|\mathbf{p}_{i,t}^{j} \times \mathbf{v}_t^j\right\|_2^2 + \left\|\gamma_{j,t} \cdot (\mathbf{p}_{i,t}^{j})^T \mathbf{v}_t^j\right\|_2^2}{\left\|\mathbf{v}_t^j\right\|_2^2}} & \text{if } (\mathbf{p}_{i,t}^{j})^T \mathbf{v}_t^j \leq 0 \end{cases}, \tag{3.10}$$

$$\gamma_{j,t} = e^{-\beta \left\|\mathbf{v}_t^j\right\|_2}. \tag{3.11}$$

Here $\mathbf{p}_{i,t}^{j}$ represents the relative position of agent j to i, and $\gamma_{j,t}$ represents the scaling factor that defined by parameter β and $\mathbf{v}_t^j$, i.e. the velocity of agent j. Such a design of weights makes agent i to give priority to agent j that is approaching, and the weight increases as agent j speeds up. This weighted combination approach is consistent with human intuition. As human drivers, we would pay more attention to cars in the vicinity and approaching. Given the combined action $\mathbf{a}_t^{i,comb}$, it is possible that the agent i could still collide with nearby agents in complex multi-agent scenarios. To guarantee safety, we introduce the well-known ORCA constraints and address the multi-agent collision avoidance problem by solving a simple convex optimization.

Before introducing the convex optimization, we briefly review some key concepts in the optimal reciprocal collision avoidance (ORCA) algorithm. The ORCA algorithm is a distributed collision avoidance algorithm for continuous time system with single integrators dynamics. Similarly to our assumption, ORCA assumes that each agent i could be approximated by a disc with different radius r_i. Each agent takes action independently by observing the velocities $\mathbf{v}_t^j$, positions $\mathbf{p}_t^j$ and radius r_j of other agents. Since ORCA is designed for agents with continuous time domain, with a

little abuse of notation, here we use $\mathbf{v}_t^i$ and $\mathbf{p}_t^i$ to represent velocity and position of agent i at continuous time t. We first consider a two agents collision avoidance problem with a continuous time horizon T. For two agents i and j at time t, ORCA returns a pair of linear constraints for velocities of the two agents independently:

$$(\mathbf{a}_{orca,j}^i)^T \mathbf{v}_{t+\tau}^i + b_{orca,j}^i \geq 0, \quad \tau \in [0, T], \tag{3.12a}$$

$$(\mathbf{a}_{orca,i}^j)^T \mathbf{v}_{t+\tau}^j + b_{orca,i}^j \geq 0, \quad \tau \in [0, T]. \tag{3.12b}$$

Here $\mathbf{a}_{orca,j}^i$, $\mathbf{a}_{orca,i}^j \in \mathbb{R}^d$ and $b_{orca,j}^i$, $b_{orca,i}^j \in \mathbb{R}$ are determined by the current positions $\mathbf{p}_t^i$, $\mathbf{p}_t^j$, velocities $\mathbf{v}_t^i$, $\mathbf{v}_t^j$ and radius r_i, r_j, which are known to both agents. If both constraints are satisfied, there would be no collision between agents i and j within time horizon T. When ORCA is extended to the multi-agent problem, there would be multiple linear constraints for each agent i,

$$A_{orca}^i \mathbf{v}_{t+\tau}^i + \mathbf{b}_{orca}^i \geq 0, \ \tau \in [0, T], \tag{3.13}$$

where $A_{orca}^i \in \mathbb{R}^{K \times d}$ and $\mathbf{b}_{orca}^i \in \mathbb{R}^K$, K is the number of nearby agents to be considered. Note that K might be smaller than $M - 1$, which means we do not need to consider all moving agents for collision avoidance of agent i. Given the maximal speed $v^{i,max}$ of each agent $i \in \mathcal{M}$, if the distance between agent i and j is greater than $(v^{i,max} + v^{j,max})T$, they would not collide within time horizon T. With properly selected K, inequality (3.13) provides a safety guarantee of each agent i in velocity space.

To avoid collisions for general nonlinear agents, we introduce the ORCA constraints (3.13) and transform them from velocity space to action space. Different from ORCA, our approach is designed for discrete time systems. We first adapt (3.13) for discrete time systems by letting $T \triangleq N \triangle t$ and use t to denote discrete time. Here N

is a positive integer representing discrete time horizon, and $\triangle t$ is the discretization time step. The ORCA constraints could be written as

$$A^i_{orca}\mathbf{v}^i_{t+d} + \mathbf{b}^i_{orca} \geq 0,\ d = 1, \ldots, N. \tag{3.14}$$

In practice, we update collision avoidance action $\mathbf{a}^i_t$ at each time step t, thus it suffices to consider the constraints (3.14) for only one step, leading to

$$A^i_{orca}\mathbf{v}^i_{t+1} + \mathbf{b}^i_{orca} \geq 0. \tag{3.15}$$

Here A^i_{orca} and $\mathbf{b}^i_{orca}$ is determined by positions, velocities (and shared radius r) of agent i and its closest K neighbor agents at time t. Safety is guaranteed if (3.15) holds at each time step. To control the agent i for collision avoidance, we need to convert the velocity constraints to state-dependent action constraints. We assume the relationship between the current action $\mathbf{a}^i_t$ and the velocity at next time step $\mathbf{v}^i_{t+1}$ is linear or could be approximated by a linear mapping. For a general nonlinear system, we could linearize the system around the current state and last action. This relationship is approximated by

$$\mathbf{v}^i_{t+1} \approx A^i_v\mathbf{a}^i_t + \mathbf{b}^i_v. \tag{3.16}$$

By combing (3.15) and (3.16), we could obtain a safety constraints for the action $\mathbf{a}^i_t$,

$$A^i_{orca}(A^i_v\mathbf{a}^i_t + \mathbf{b}^i_v) + \mathbf{b}^i_{orca} \geq 0. \tag{3.17}$$

If inequality (3.17) holds at each time step and the relationship between action $\mathbf{a}^i_t$ and the velocity $\mathbf{v}^i_{t+1}$ is exactly linear, it is strictly guaranteed that the agent i would never collide with any other agents [5]. Rigorously speaking, the linearization for nonlinear system can introduce small errors. However, in practice this can be worked around by slightly enlarging the radius r in our approach.

The combined action from RL (3.8) provides preferred action that would drive the agent to reach the goal state, and the state-dependent constraints from (3.17) provides safety constraints for the agent. To reach the goal while avoiding collisions, agent i selects the action that is the closest to $\mathbf{a}_t^{i,comb}$ while satisfying safety constraints (3.17) and physical constraints:

$$\mathbf{a}_t^i = \underset{\mathbf{a}\in\mathbb{R}^m}{\arg\min} \left\| \mathbf{a} - \mathbf{a}_t^{i,comb} \right\|_2^2 \tag{3.18a}$$

$$\text{s.t. } A_{orca}^i (A_v^i \mathbf{a} + \mathbf{b}_v^i) + \mathbf{b}_{orca}^i \geq 0 \tag{3.18b}$$

$$\mathbf{a} \in \mathscr{A}. \tag{3.18c}$$

Here (3.18a) is a quadratic cost function, (3.18b) are linear safety constraints from ORCA, and (3.18c) is the convex physical constraints. The optimization problem (3.18) is a simple convex optimization and can be solved efficiently.

To sum up, given the trained policy $\bar{\pi}(\cdot, \bar{\theta})$ from the two agents collision avoidance problem, we propose a decentralized multi-agent collision avoidance algorithm by solving a simple convex optimization with state-dependent safety constraints from ORCA. If the relationship between action and velocity is linear, the safety constraints provide a strict safety guarantee. The overall algorithm is summarized in Algorithm 4. In the next section, we demonstrate our algorithm via different challenging simulations.

3.4 Simulations and Results

We illustrate the performance of our proposed method through several multi-agent interactive scenarios.

Algorithm 4 Reciprocal Collision Avoidance for General Nonlinear Agents using Reinforcement Learning

Input: The trained policy $\bar{\pi}(\cdot, \bar{\theta})$ from the two agents collision avoidance problem, the initial state $\mathbf{s}^i_{initial}$, and the goal state of each agent $\tilde{\mathbf{s}}^i$, $i = 1, \ldots, M$
Initialization: $\mathbf{s}^i_0 \leftarrow \mathbf{s}^i_{initial}$, $t \leftarrow 0$
while not all agents reach goal states **do**
    for $i = 1, \ldots, M$ **do**
        for $j = 1, \ldots, M$ and $j \neq i$ **do**
            $\mathbf{a}^i_{j,t} \leftarrow \bar{\pi}(\mathbf{o}^i_{j,t}, \bar{\theta})$
        end for
        Combine actions, $\mathbf{a}^{i,comb}_t$ via (3.8)
        Select action $\mathbf{a}^i_t$ by solving optimization (3.18)
        Apply $\mathbf{a}^i_t$ to agent i
    end for
    $t \leftarrow t + 1$
end while

3.4.1 Simulation Setup

Throughout this section, we adopt the nonlinear kinematic bicycle model [65] operating in a two-dimensional Euclidean space. The state transition function is given as (2.21).

Unlike other RL-based approaches [19, 10, 20, 66, 23] that assume that velocity of each agent could be directly controlled, our approach is directly applicable to the above nonlinear model with acceleration a and steering angle δ_f as control. As far as we know, this is the first RL-based multi-agent collision avoidance algorithm that directly works for kinematic bicycle model. As for ORCA-based approach, [18] is designed for bicycle model by fixing a tracking controller and transferring the control to velocity space. But like many other ORCA-based approaches, it requires a high-level planning module to provide preferred velocity, which would greatly influence the performance.

To reduce redundancy and simplify training, we use a local coordinate for each agent with x axis pointing towards the inertial heading of the vehicle in simulation. The observation of agent i at time t is represented by concatenating the goal position, velocity of agent i, positions and velocities of nearby agents in the local coordinate. The dynamics (2.21) is discretized in time and all simulations operate with discretization time step of 0.05 s (20 Hz).

3.4.2 Policy Training with RL

The policy for two-agent collision avoidance is trained following the described procedure in Section 3.3.1. We use a two-layer neural network with 16 neurons at each layer. The network takes the observation of size 8 and predicts the control action of size 2. Compared with other RL-based approaches [23, 24, 19, 10, 20, 44], this is a significantly lightweight design of network architecture with a total of only 450 training parameters. On an eight-core i7-7700K CPU 4.20GHz machine, the empirically optimal policy is obtained within 20 minutes of wall-clock time training. The policy is then combined with the ORCA constraints to handle multi-agent interactions. It is worth noting that all simulation results in 3.4.3 comes from the same trained policy.

3.4.3 Simulation Results

As shown in Fig. 3.2, we first test the proposed method in simple scenarios with up to eight agents. The trajectory of each agent is represented by circles with the same color, and the color gradually fades as the agent moves. The goal position of each agent is symmetric with the initial position about the origin. It is worth emphasizing that the three-agent interactive scene is inspired by a well-know deadlock scenario of the classic ORCA method [67]. The symmetric setup of scenarios with four and

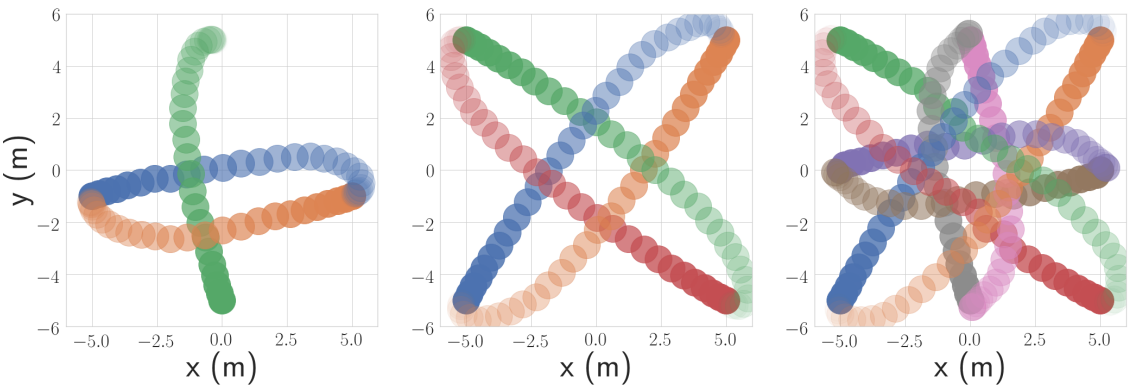


Figure 3.2: Trajectories of three, four and eight agents

eight agents are also easy to result in conflicts of actions. Despite the difficulty, our proposed approach successfully generates a set of smooth trajectories reaching the target position for all the three scenarios without encountering any collision, congestion or deadlock.

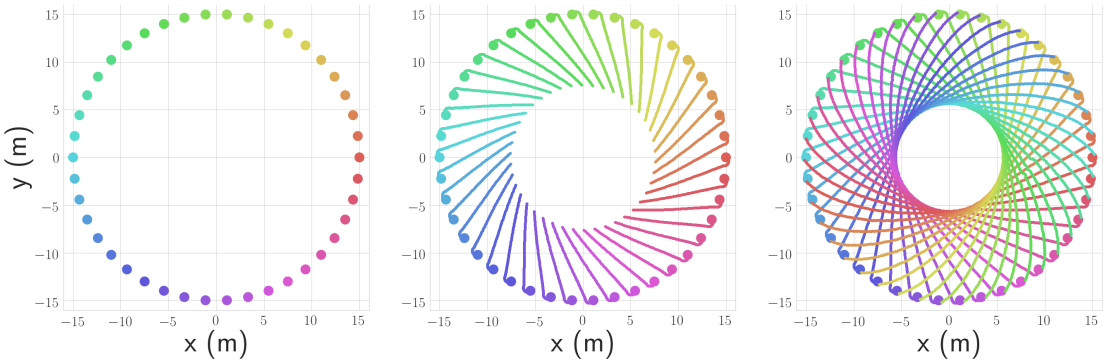


Figure 3.3: Trajectories of 42 agents. The left figure: the initial position of each agent, different agents are marked by dots with different color. The middle figure: the trajectory in the middle of the process. The right figure: complete trajectories.

We now demonstrate our approach with a more complex task. As shown in Fig. 3.3, we have 42 agents evenly distributed on a circle, with initial velocities pointing to the adjacent neighbor of each agent clockwise. The goal position of each agent is symmetric with the initial position about the origin. Trajectories of different agents are marked by different colors. It is clear from the figure that a set of safe and smooth trajectories are generated with each agent first turning right to have the heading more aligned with the target position, then passing the origin from the left side with ORCA-constrained actions for collision avoidance.

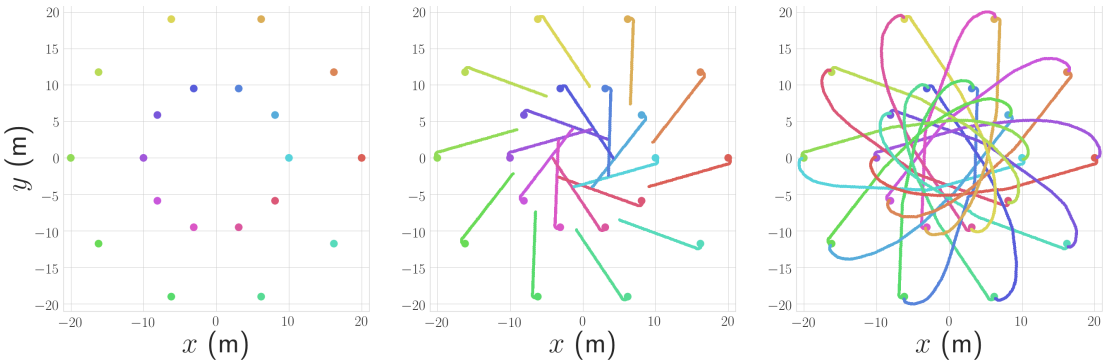


Figure 3.4: Trajectories of 20 agents. The left figure: the initial position of each agent is denoted by solid circle with different colors. The middle figure: the trajectory in the middle of the process. The right figure: complete trajectories.

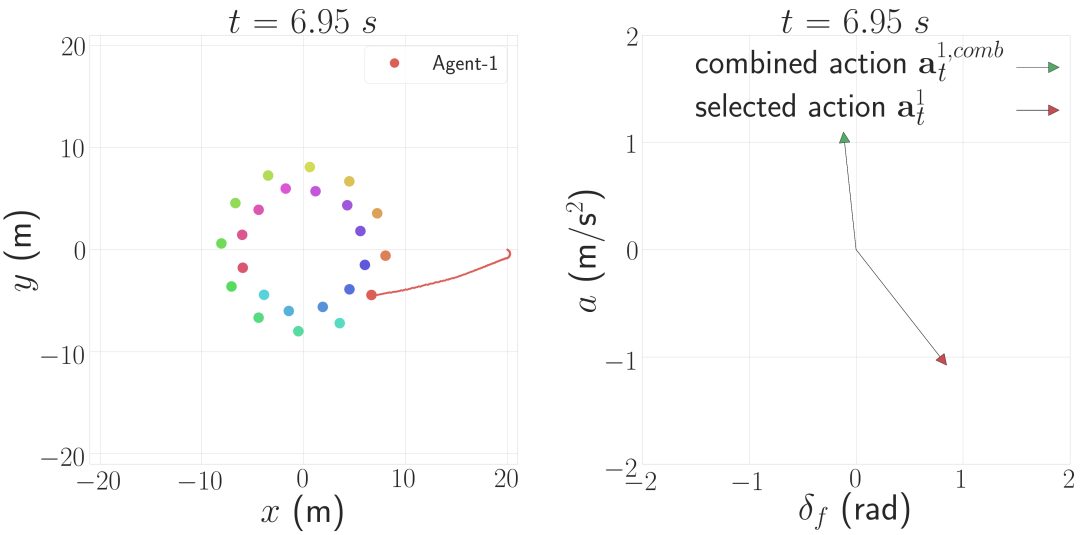
The next example is to demonstrate the capability of the proposed method in a customized task that requires more sophisticated interactions with surrounding agents. As shown in Fig. 3.4, initial positions are marked in the left figure as dots with different colors. We have 20 agents that are evenly distributed on two co-centric circles. For agents located on the large circle, the goal positions are located on the small circle that aligns with the start position through the center, and vice versa.

Such a task is more difficult than previous ones. In order to reach the goal position, each agent is expected to have conflicts with agents initialized from both circles. However, as shown in Fig. 3.4, trajectories generated by our approach is smooth and safe.

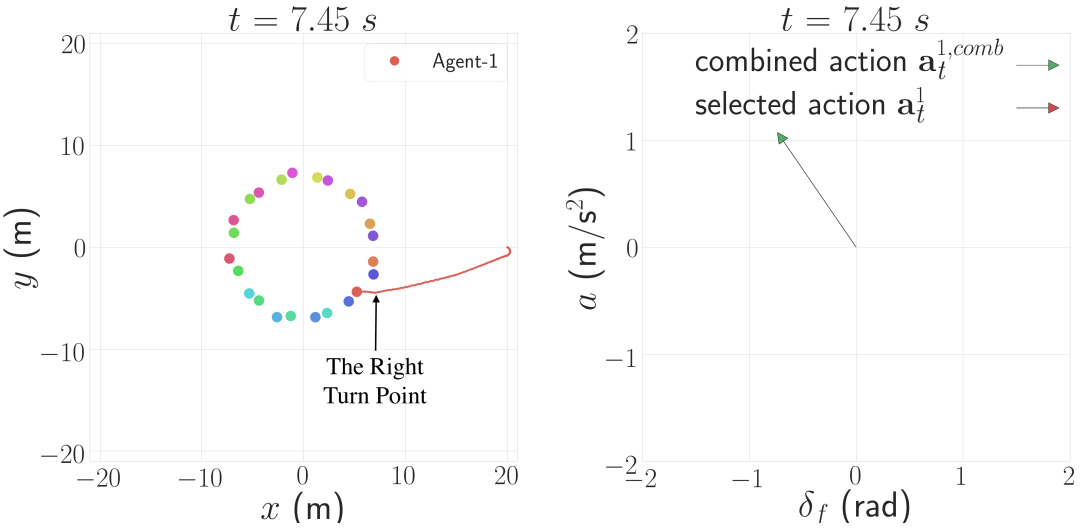
Furthermore, recall the discussion in Section 3.3.2 where the combination of actions from trained policy could also drive the controlled agent to the unsafe state. From Fig. 3.5, we select one agent and further illustrates *how the ORCA constraints alter the selected actions to enhance safety.* As shown in Fig. (3.5a), when $t = 6.95$ s, the combined action of agent 1 would drive the agent forward to its left. However, this would cause a collision with the agent (marked by dark blue) to its front left. To avoid collision, the ORCA constraints revise the agent to a right turn with deceleration. We can observe an obvious twist to the right from the position trajectory in the left figure of Fig.(3.5b), which is marked by a black arrow. The example clearly indicates the importance of having ORCA constraints in the proposed approach to avoid collisions.

3.5 Conclusions

In this chapter, we propose a decentralized collision avoidance algorithm for general nonlinear systems by combing the RL and ORCA. The proposed method consists of two stages: first train a two-agent collision avoidance policy using RL; then compute the overall multi-agent collision avoidance action by properly combining multiple avoidance actions while respecting state-dependent safety constraints induced by ORCA. Since the multi-agent collision avoidance problem is first decomposed into two agents collision avoidance problems, the overall training process is much faster than



(a) Positions, trajectory, actions when $t = 6.95\ s$



(b) Positions, trajectory, actions when $t = 7.45\ s$

Figure 3.5: Illustration of the importance of ORCA constraints through a 24 agents example. The left column shows positions of all agents and trajectories of agent 1. The right column shows combined actions and selected actions of agent 1.

other RL-based algorithms. The introduction of ORCA significantly improves system safety. We demonstrate the proposed method through several collision avoidance problems with kinematic bicycle agent models.

Chapter 4: Fitted Value Iteration in Continuous Markov Decision Processes with State-Dependent Action Sets

4.1 Introduction

In both Chapter 2 and 3, the collision avoidance requirement is incorporated as state-dependent action constraints. In Chapter 3, RL algorithms are introduced to solve a two agents collision avoidance problem. One natural extension of our previous work is to apply RL algorithms to MDPs with state-dependent action constraints. In this chapter, we study RL algorithms for MDPs with continuous state, continuous action and state-dependent action constraints, and extends our work to monotone MDPs.

Approximate dynamic programming (ADP) and Reinforcement Learning (RL) algorithms have become powerful tools to solve MDPs with varying degree of approximations. Many RL algorithms [40, 68, 69, 70] have been developed for general MDPs. In these "fitted" algorithms, the value functions are evaluated using certain empirical means and function approximators are used to store the value functions [60, 64, 71, 72, 61]. The proof of convergence of fitted algorithms follow from delicate arguments under a variety of sufficient conditions. All of these algorithms can be

viewed within the mathematical framework of iterated random operators over complete separable metric spaces [73, 62, 74, 75]. Recently, the notion of probabilistic fixed point of iterated random operators was introduced in [76] in the context of empirical value iteration of finite MDPs. This was later extended to very general settings in [62]. Besides, by exploiting some structural information about the MDP, one can accelerate the convergence of ADP algorithms, as has been shown in [77, 78]. In this chapter, we investigate one such ADP algorithm, called fitted value iteration and fitted Q-value iteration.

In this chapter, we leverage the main result of [62] to show that the fitted value iteration and fitted Q-value iteration converges under fairly general conditions on the MDPs with a universal function approximating class [79]. We further consider the case where the optimal value function is monotone, which is a common class of MDPs naturally arising in many fields such as economics, finance, and operations research, among many others; see, for example, [80, Chapter 4.7], [81, Chapter 18], [77, 82], and the references therein.

Our work is inspired by the Monotone-ADP algorithm for finite MDPs in [77], where the authors proposed a monotone projection operator that preserves the monotonicity of the value function. We extend their framework to continuous-state continuous-action MDPs here. The key idea is to use monotone neural networks [83] to approximate the value functions at every step of the algorithm. The convergence of this new algorithm follows immediately from our main result.

This work significantly generalizes some of the earlier work in a non-trivial manner by leveraging the main result of [62]. The key contributions are:

- **State-Dependent Action Constraints:** In most fitted value iteration problems [60, 61, 62, 63], the authors assume that all actions can be taken at all states. This is incorrect in many resource constrained problems – for instance, one cannot draw more energy from a battery than the state-of-charge of the battery. Thus, we consider in this work that the admissible action set at every state depends on that state.

- **Lipschitz Continuity:** Due to the admissible action set being state-dependent, establishing the Lipschitz continuity of value function is non-trivial. We use [84] to identify sufficient conditions under which the value function is Lipschitz.

- **Relaxed Assumptions of MDPs:** In [60, 63, 64], the authors make certain absolute continuity assumptions on the transition kernel of the MDPs. We believe that this is a strong assumption, since it rules out any deterministic transitions in the MDPs. We relax this restriction under the mild assumption of having a universal function approximating class (that is, this class is dense in the space of continuous and bounded functions under supremum norm).

- **Monotone MDPs:** As a byproduct of the main result, we extend the convergence result to a sufficiently general class of monotone MDPs.

4.2 Problem Formulation

Let $(\Omega, \mathscr{F}, \mathbb{P})$ be a standard probability space. We consider an infinite horizon MDP with the state transition function

$$S_{t+1} = f(S_t, A_t, W_t) \quad t = 0, 1, \ldots, \tag{4.1}$$

where $S_t \in \mathscr{S} \subseteq \mathbb{R}^d$, $A_t \in \Gamma(S_t) \subseteq \mathscr{A} \subseteq \mathbb{R}^p$ and $W_t \in \mathscr{W} \subseteq \mathbb{R}^l$ represents the state, action, and exogenous noise at time t, respectively. It is assumed that $\{W_t\}_{t=0}^{\infty}$ are independent and identically distributed. Here $\Gamma : \mathscr{S} \to 2^{\mathscr{A}}$ is a correspondence that represents state-dependent action constraints, where $2^{\mathscr{A}}$ stands for the power set of $\mathscr{A}$. Let $\mathscr{D} \triangleq \{(s,a) \in \mathscr{S} \times \mathscr{A} | a \in \Gamma(s)\}$ be defined as the joint space of state and action, and let $c(\cdot,\cdot) : \mathscr{D} \to \mathbb{R}$ denote the one-stage cost function. We assume that $\mathscr{D} \subset \mathscr{S} \times \mathscr{A}$ is a compact set.

A feasible policy π is defined as a measurable function $\pi : \mathscr{S} \to \mathscr{A}$ such that $\pi(s) \in \Gamma(s)$ for all $s \in \mathscr{S}$. We use Π to denote the set of all feasible policies. Given a discount factor $\gamma \in (0,1)$, our goal is to minimize the expectation of the discounted accumulated cost by solving the following optimization:

$$v^*(s) = \inf_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(S_t, \pi(S_t)) \middle| S_0 = s\right], \tag{4.2}$$

for all $s \in \mathscr{S}$, and v^* is called the optimal value function. It is equivalent to solve the following optimization:

$$q^*(s,a) = \inf_{\pi \in \Pi} \mathbb{E}\left[c(S_0, A_0) + \sum_{t=1}^{\infty} \gamma^t c(S_t, \pi(S_t)) \middle| S_0 = s, A_0 = a\right], \tag{4.3}$$

for all $(s,a) \in \mathscr{D}$, and q^* is called the optimal Q function.

Under mild conditions [85], the optimal value function v^* and the optimal Q function q^* can be obtained by iteratively applying the Bellman operators H_v and H_q, respectively, which are defined as

$$[H_v(v)](s) \triangleq \inf_{a \in \Gamma(s)} \Big[c(s,a) + \gamma \mathbb{E}\left[v(f(s,a,W)\right]\Big],$$
$$[H_q(q)](s,a) \triangleq c(s,a) + \gamma \mathbb{E}\left[\inf_{a' \in \Gamma(f(s,a,W))} q(f(s,a,W), a')\right],$$

where W has the same distribution as W_t. The iterative processes

$$v_{k+1} = H_v(v_k),$$

$$q_{k+1} = H_q(q_k)$$

are called the *value iteration* and *Q-value iteration* algorithm, respectively. Here k is the index of iteration.

4.2.1 Empirical Bellman Operators

In high dimensional settings, computing the expectations in Bellman operators H_v and H_q is computationally challenging. Thus, one often approximates the Bellman operators by empirical Bellman operators $\hat{H}_v^{k,n} : \mathcal{C}_b(\mathscr{S}) \to \mathcal{C}_b(\mathscr{S})$ and $\hat{H}_q^{k,n} : \mathcal{C}_b(\mathscr{D}) \to \mathcal{C}_b(\mathscr{D})$, respectively. Here n denotes the number of noise samples per iteration. Let $\{W_{k,i}\}_{i=1}^n$ be independent samples of the noise at the iteration k. The empirical Bellman operators $\hat{H}_v^{k,n}$ and $\hat{H}_q^{k,n}$ are defined as

$$[\hat{H}_v^{k,n}(v)](s) \triangleq \inf_{a \in \Gamma(s)} \Big[c(s,a) + \gamma \frac{1}{n} \sum_{i=1}^{n} v(f(s,a,W_{k,i})) \Big],$$

$$[\hat{H}_q^{k,n}(q)](s,a) \triangleq c(s,a) + \gamma \frac{1}{n} \sum_{i=1}^{n} \inf_{a' \in \Gamma(f(s,a,W_{k,i}))} q(f(s,a,W_{k,i}), a').$$

4.2.2 Fitted Value Iteration and Q-Value Iteration

For continuous state space $\mathscr{S}$ or action space $\mathscr{A}$, function approximators are necessary to store or represent the value functions or Q functions. Common function approximating classes include non-parametric regression models [79], such as nearest neighbor or kernel-based function approximators, and parametric regression models [86], such as neural networks, support vector machines, or reproducing kernel Hilbert spaces. In the aforementioned references, these function approximating classes have

been shown to be *universal*, that is, these classes are dense in the space of continuous and bounded functions under the supremum norm.

Let $\hat{M}_v^{k,m,l} : \mathcal{C}_b(\mathscr{S}) \to \mathcal{C}_b(\mathscr{S})$ and $\hat{M}_q^{k,m,l} : \mathcal{C}_b(\mathscr{D}) \to \mathcal{C}_b(\mathscr{D})$ denote the projection operators, which take the output of empirical Bellman operators as input and outputs a function in the chosen function approximating class. These operators are called projection operators [60, 64, 63].

We construct the $\hat{M}_v^{k,m,l}$ as follows: let $\mathcal{F}_l \subseteq \mathcal{C}_b(\mathscr{S})$ denote the function class of the model, where l is the number of data points or parameters to define functions in this function class; for example, l could be the number of parameters that define a neural network. For a function $v \in \mathcal{C}_b(\mathscr{S})$, the algorithm samples m uniformly distributed states $\left\{s^{k,i}\right\}_{i=1}^m$. Then, the projection operator $\hat{M}_v^{k,m,l}$ is defined as

$$\hat{M}_v^{k,m,l}(v) = \arg\min_{f \in \mathcal{F}_l} L\left(v, f \middle| \left\{s^{k,i}\right\}_{i=1}^m\right). \tag{4.4}$$

Here, $L\left(\cdot, \cdot \middle| \left\{s^{k,i}\right\}_{i=1}^m\right) : \mathcal{C}_b(\mathscr{S}) \times \mathcal{F}_l \to \mathbb{R}$ is a loss function. One common example is the mean squared error (MSE), which is defined as

$$L\left(v, f \middle| \left\{s^{k,i}\right\}_{i=1}^m\right) = \frac{1}{m} \sum_{i=1}^m \left(v(s^{k,i}) - f(s^{k,i})\right)^2.$$

In a similar fashion, we can define the projection operator $\hat{M}_q^{k,m,l}$ for projecting the Q function by replacing the samples of states with samples of state-action pairs and taking $\mathcal{F}_l$ to be a set of functions on $\mathscr{D}$.

At each iteration k, we update the function approximator by combining the empirical Bellman operators and projection operators, which are denoted by random operators $\hat{T}_v^{k,n,l} : \mathcal{C}_b(\mathscr{S}) \to \mathcal{C}_b(\mathscr{S})$ and $\hat{T}_q^{k,n,l} : \mathcal{C}_b(\mathscr{D}) \to \mathcal{C}_b(\mathscr{D})$ as follows

$$\hat{T}_v^{k,n,l} \triangleq \hat{M}_v^{k,n_2(n),l} \circ \hat{H}_v^{k,n_1(n)}, \tag{4.5}$$

$$\hat{T}_q^{k,n,l} \triangleq \hat{M}_q^{k,n_2(n),l} \circ \hat{H}_q^{k,n_1(n)}, \tag{4.6}$$

where $n_2(n)$ an $n_1(n)$ are monotone functions of n. The fitted value iteration and Q-value iteration algorithms are defined, respectively, as

$$\hat{v}^{k+1,n,l} = \hat{T}_v^{k,n,l}(\hat{v}^{k,n,l}), \tag{4.7}$$

$$\hat{q}^{k+1,n,l} = \hat{T}_q^{k,n,l}(\hat{q}^{k,n,l}), \tag{4.8}$$

where $k \in \mathbb{N}$ denotes the iteration index and $\hat{v}^{0,n,l}$ or $\hat{q}^{0,n,l}$ can be picked arbitrarily in $\mathcal{C}_b(\mathscr{S})$ or $\mathcal{C}_b(\mathscr{D})$. It is easy to observe that: (a) The operators $\hat{T}_v^{k,n,l}$ and $\hat{T}_q^{k,n,l}$ are independent and identically distributed operators; (b) $\hat{M}_v^{k,m,l}$ satisfies: for any $v_1, v_2 \in \mathcal{C}_b(\mathscr{S})$,

$$\begin{aligned}\left\|\hat{M}^{k,m,l}(v_1) - \hat{M}^{k,m,l}(v_2)\right\|_\infty \leq &\left\|\hat{M}^{k,m,l}(v_1) - v_1\right\|_\infty + \|v_1 - v_2\|_\infty \\ &+ \left\|\hat{M}^{k,m,l}(v_2) - v_2\right\|_\infty.\end{aligned}$$

Assume that $\cup_{l=1}^{\infty}\mathcal{F}_l$ is universal. Then, for any $\zeta > 0$, there exists $m_0, l_0 > 0$ such that

$$\left\|\hat{M}^{k,m,l}(v) - v\right\|_\infty < \zeta$$

for all $l \geq l_0$ and $m \geq m_0$ with high probability. Thus, we conclude that with high probability,

$$\left\|\hat{M}^{k,m,l}(v_1) - \hat{M}^{k,m,l}(v_2)\right\|_\infty \leq \|v_1 - v_2\|_\infty + 2\zeta.$$

Thus, $\hat{M}_v^{k,m,l}$ is a non-expansive operator with certain error for sufficiently large m, l if $\cup_{l=1}^{\infty}\mathcal{F}_l$ is universal.

4.3 Main Results

In this section, we present the convergence result for the proposed fitted value iteration and fitted Q-value iteration algorithms in (4.8). We first identify some

sufficient conditions on the MDP under which the optimal value function v^* and the optimal Q function q^* are Lipschitz continuous. Then, we show that the proposed algorithms converges to the optimal solution in an appropriate sense when some reasonable conditions about the projection operators are met.

4.3.1 Lipschitz MDP

We first discuss the conditions about the MDP under which the optimal value function and optimal Q function are Lipschitz. Given any metrics ρ_S and ρ_A for $\mathscr{S}$ and $\mathscr{A}$, respectively, we define a metric ρ_D over $\mathscr{D}$ as

$$\rho_D\left((s,a),(s',a')\right) \triangleq \rho_S(s,s') + \rho_A(a,a').$$

For any two closed sets $\mathscr{A}_1$, $\mathscr{A}_2 \subseteq \mathscr{A}$, we define the *Hausdorff metric* ρ_H as

$$\rho_H(\mathscr{A}_1,\mathscr{A}_2) \triangleq \max\left[\sup_{a_1\in\mathscr{A}_1}\inf_{a_2\in\mathscr{A}_2}\rho_A(a_1,a_2), \sup_{a_2\in\mathscr{A}_2}\inf_{a_1\in\mathscr{A}_1}\rho_A(a_1,a_2)\right].$$

We place the following assumptions on the MDP.

Assumption 2. *The following holds:*

(i) The state space $\mathscr{S}$ is compact, and the correspondence $\Gamma : \mathscr{S} \to 2^{\mathscr{A}}$ is nonempty, compact-valued, and there exists $L_D \geq 0$, such that for all $s, s' \in \mathscr{S}$, we have

$$\rho_H\left(\Gamma(s),\Gamma(s')\right) \leq L_D\rho_S(s,s').$$

(ii) For every $w \in \mathscr{W}$, the state transition function $f(\cdot,\cdot,w)$ is Lipschitz continuous in $(s,a) \in \mathscr{D}$ with Lipschitz coefficient $L_f(w)$ and $L_P \triangleq \int L_f(w)\mathbb{P}\{dw\} < \infty$.

(iii) The cost function $c : \mathscr{D} \to \mathbb{R}$ is Lipschitz continuous with Lipschitz coefficient L_c.

(iv) L_D *in i and* L_P *in ii satisfies* $L_P(1 + L_D) < 1/\gamma$.

Note that Assumption 2 (i) implies the correspondence Γ is Lipschitz under Hausdorff metric ρ_H. With the above notations and assumption, we have the following theorem for Lipschitz property.

Theorem 1. *If Assumption 2 holds, then*

(i) The Bellman operators $H_v : \mathcal{C}_b(\mathscr{S}) \to \mathcal{C}_b(\mathscr{S})$ *and* $H_q : \mathcal{C}_b(\mathscr{D}) \to \mathcal{C}_b(\mathscr{D})$ *are contraction operators with coefficient* γ *and fixed points* v^* *and* q^**, respectively.*

(ii) v^* *and* q^* *are Lipschitz continuous function with Lipschitz coefficient* $L_{v^*} \leq \frac{L_c(1+L_D)}{1-\gamma L_P(1+L_D)}$, $L_{q^*} \leq \frac{L_c}{1-\gamma L_P(1+L_D)}$, *respectively.*

The proof for Theorem 1 is given in Section 4.4.2. Note here that for the result of Theorem 1 (i) to hold, the Lipschitz continuity of the correspondence $\Gamma(\cdot)$, transition function $f(\cdot, \cdot, w)$ and cost function $c(\cdot, \cdot)$ can be replaced by a weaker assumption that they are just continuous. The Lipschitz continuity established in Theorem 1 (ii) is needed to establish the other main results in the sequel.

4.3.2 Fitted Value Iteration and Q-Value Iteration

In order to achieve the convergence result of the proposed algorithms, we also need the following reasonable assumptions on the projection operators.

Assumption 3. *The projection operator* $\hat{M}_v^{k,m,l} : \mathcal{C}_b(\mathscr{S}) \to \mathcal{C}_b(\mathscr{S})$ *satisfies the followings two conditions*

(i) $\hat{M}_v^{k,m,l}$ *is non-expansive, i.e. for all* $v_1, v_2 \in \mathcal{C}_b(\mathscr{S})$*, we have*

$$\left\| \hat{M}_v^{k,m,l}(v_1) - \hat{M}_v^{k,m,l}(v_2) \right\|_\infty \leq \|v_1 - v_2\|_\infty + \hat{\zeta}^{k,m,l},$$

where $\hat{\zeta}^{k,m,l} \leq \bar{\zeta} < \infty$ *almost surely and* $\hat{\zeta}^{k,m,l} \to 0$ *as* $m, l \to \infty$. $\bar{\zeta} > 0$ *is a constant number.*

(ii) For any $\epsilon > 0$ *and* $\delta > 0$*, there exists* M_1 *that may depends on* v^* *such that*

$$\mathbb{P}\left\{\left\|\hat{M}_v^{k,m,l}(v^*) - v^*\right\|_\infty > \epsilon\right\} < \delta \text{ for all } m \geq M_1.$$

When the class of function approximators is dense in the space of optimal value functions and Q functions, the above assumption can be easily satisfied. Then, we have the following theorem where the convergence of the fitted value iteration algorithm is established.

Theorem 2. *If Assumptions 2 and 3 holds, then* v^* *is a probabilistic fixed point of* $\hat{T}_v^{k,n,l}$*, i.e. for any initial* $v_0 \in \mathcal{C}_b(\mathscr{S})$ *and* $\kappa > 0$*, we have*

$$\limsup_{l\to\infty} \limsup_{n\to\infty} \limsup_{k\to\infty} \mathbb{P}\left\{\left\|\hat{v}^{k,n,l} - v^*\right\|_\infty > \kappa\right\} = 0.$$

where $\hat{v}^{0,n,l} = v_0$ *and* $\hat{v}^{k+1,n,l} = \hat{T}_v^{k,n,l}(\hat{v}^{k,n,l})$ *for* $k \geq 0$.

Similarly, for the Q function and projection operator $\hat{M}_q^{k,m,l}$, we have the following assumption and theorem.

Assumption 4. *The projection operator* $\hat{M}_q^{k,m,l} : \mathcal{C}_b(\mathscr{D}) \to \mathcal{C}_b(\mathscr{D})$ *satisfies the followings two conditions*

(i) $\hat{M}_q^{k,m,l}$ *is non-expansive, i.e. for all* $q_1, q_2 \in \mathcal{C}_b(\mathscr{D})$*, we have*

$$\left\|\hat{M}_q^{k,m,l}(q_1) - \hat{M}_q^{k,m,l}(q_2)\right\|_\infty \leq \|q_1 - q_2\|_\infty + \hat{\zeta}^{k,m,l},$$

where $\hat{\zeta}^{k,m,l} \leq \bar{\zeta} < \infty$ *almost surely and* $\hat{\zeta}^{k,m,l} \to 0$ *as* $m, l \to \infty$.

(ii) For any $\epsilon > 0$ and $\delta > 0$, there exists M_2 that may depends on q^* such that

$$\mathbb{P}\left\{\left\|\hat{M}_q^{k,m,l}(q^*) - q^*\right\|_\infty > \epsilon\right\} < \delta \text{ for all } m \geq M_2.$$

Theorem 3. *If Assumptions 2 and 4 holds, then q^* is a probabilistic fixed point of $\hat{T}_q^{k,n,l}$, i.e. for every initial $q_0 \in \mathcal{C}_b(\mathscr{D})$ and $\kappa > 0$, we have*

$$\limsup_{l\to\infty} \limsup_{n\to\infty} \limsup_{k\to\infty} \mathbb{P}\left\{\left\|\hat{q}^{k,n,l} - q^*\right\|_\infty > \kappa\right\} = 0,$$

where $\hat{q}^{0,n,l} = q_0$ and $\hat{q}^{k+1,n,l} = \hat{T}_q^{k,n,l}(\hat{q}^{k,n,l})$ for $k \geq 0$.

4.4 Proofs of the Main Results

In this section, the main results of our chapter are proved. We first briefly recall some established results on the iterated random operators. Then, we establish a lemma for the empirical Bellman operator. Finally, we present the complete proofs of the main results.

4.4.1 Probabilistic Contraction Analysis of Iterated Random Operators

In [76], the authors considered problem of convergence of iterated random operators over Euclidean spaces with supremum norm, and derived some sufficient conditions so that the generated Markov chain converges to a certain point in probability. This framework was generalized to iterated random operators over complete metric spaces in [62], which we discuss next.

Assume that $T : \mathcal{X} \to \mathcal{X}$ is a contraction operator defined over a complete metric space $\mathcal{X}$ with a unique fixed point x^*. The metric space is endowed by metric ρ_X. In practice, T is usually approximated by a sequence of random operators $\{\hat{T}^{k,n}\}_{k=1}^{\infty}$

to reduce the computation complexity. It is assumed that each $\hat{T}^{k,n}$ is generated by n independent and identically distributed samples of one random variable. For all $x_1, x_2 \in \mathcal{X}$ and $n, k \in \mathbb{N}$, we assume that

$$\rho_X\left(\hat{T}^{k,n}(x_1), \hat{T}^{k,n}(x_2)\right) \leq \hat{\gamma}^{k,n}\rho_X\left(x_1, x_2\right) + \hat{\zeta}^{k,n},$$

where $\hat{\gamma}^{k,n}$ and $\hat{\zeta}^{k,n}$ are random variables taking values in the set $[0,1]$ and $[0,\infty)$, respectively. Then we can generate a Markov chain $\left\{\hat{x}^{k,n}\right\}_{k=1}^{\infty}$ by letting $\hat{x}^{k+1,n} = \hat{T}^{k,n}(\hat{x}^{k,n})$. The following result establishes the convergence in probability of $\hat{x}^{k,n}$ to the fixed point x^* as k and n goes to infinity.

Assumption 5. *The following holds:*

(i) $\mathcal{X}$ *is a complete metric space, and* T *is a contraction operator on* $\mathcal{X}$ *with a contraction coefficient* $\gamma < 1$ *and unique fixed point* x^*.

(ii) For each $n \in \mathbb{N}$, $\{\hat{T}^{k,n}\}_{k=1}^{\infty}$ *is a sequence of independent and identically distributed operators.*

(iii) For any $k \in \mathbb{N}$ *and* $\delta \in (0, 1-\gamma)$,

$$\lim_{n\to\infty} \mathbb{P}\left\{\hat{\gamma}^{k,n} \geq 1-\delta\right\} = 0$$

and $\hat{\gamma}^{k,n} \leq 1$ *almost surely for all* $n, k \in \mathbb{N}$.

(iv) For any $k \in \mathbb{N}$ *and* $\epsilon > 0$,

$$\lim_{n\to\infty} \mathbb{P}\left\{\rho_X\left(\hat{T}^{k,n}(x^*), T(x^*)\right) + \hat{\zeta}^{k,n} \geq \epsilon\right\} = 0.$$

(v) There exists $\bar{w} > 0$ *such that* $\rho_X\left(\hat{T}^{k,n}(x^*), T(x^*)\right) + \hat{\zeta}^{k,n} \leq \bar{w}$ *almost surely for all* $n, k \in \mathbb{N}$.

We have the following result from [62, Theorem 1].

Theorem 4. *If Assumption 5 holds, for all $\epsilon > 0$, we have*

$$\lim_{n\to\infty} \limsup_{k\to\infty} \mathbb{P}\left\{\rho(\hat{x}^{k,n}, x^*) > \epsilon\right\} = 0.$$

For the problem in this chapter, $\mathcal{C}_b(\mathscr{S})$ and $\mathcal{C}_b(\mathscr{D})$ corresponds to the metric space $\mathcal{X}$, $\hat{T}^{k,n}$ is replaced by $\hat{T}_v^{k,n,l}$ and $\hat{T}_q^{k,n,l}$, and x^* becomes v^* and q^*. In what follows, we prove that the random operators $\hat{T}_v^{k,n,l}$ and $\hat{T}_q^{k,n,l}$ defined in (4.5)-(4.6) satisfy the Assumption 5. Thus, our empirical fitted value iteration and Q-value iteration algorithms converges to the optimal value function or the optimal Q function of the MDP by Theorem 4.

4.4.2 Proof of Theorem 1

Proof of (i): the proof for H_v being a contraction operators over $\mathcal{C}_b(\mathscr{S})$ is given in [87, Theorem 9.6 and Exercise 9.7 b]. Since $\mathcal{C}_b(\mathscr{S})$ is a complete metric space, H_v has a unique fixed point in $\mathcal{C}_b(\mathscr{S})$. As shown by [85, Proposition 1.2.1], the unique fixed point of H_v is v^*. The corresponding proof for q^* is very similar to v^*, thus we omit the proof here.

Proof of (ii): the conclusion for v^* follows directly from [84, Theorem 4.1]. For q^*, we can prove it based on the relationship between v^* and q^*

$$q^*(s,a) = c(s,a) + \gamma \mathbb{E}\left[v^*(f(s,a,W))\right].$$

4.4.3 Proof of Theorem 2 and 3

Before proceeding to the proof of Theorem 2 and 3, we first provide the following lemma for the empirical Bellman operators $\hat{H}_v^{k,n}$ and $\hat{H}_q^{k,n}$.

Lemma 1. *If Assumption 2 holds, then*

(i) The empirical Bellman operators $\hat{H}_v^{k,n} : \mathcal{C}_b(\mathscr{S}) \to \mathcal{C}_b(\mathscr{S})$ *and* $\hat{H}_q^{k,n} : \mathcal{C}_b(\mathscr{D}) \to \mathcal{C}_b(\mathscr{D})$ *are contraction operators with contraction coefficient* γ.

(ii) For any $\epsilon > 0$, *we have the following holds:*

$$\lim_{n\to\infty} \mathbb{P}\left\{\left\|\hat{H}_v^{k,n}(v^*) - H_v(v^*)\right\|_\infty \geq \epsilon\right\} = 0,$$
$$\lim_{n\to\infty} \mathbb{P}\left\{\left\|\hat{H}_q^{k,n}(q^*) - H_q(q^*)\right\|_\infty \geq \epsilon\right\} = 0.$$

Proof. See Appendix A.1. □

Lemma 1, together with Assumption 3, implies Assumption 5 (i) - (iv) is satisfied. To see that Assumption 5 (v) holds, note that $\|\hat{T}_v^{k,n,l}(v^*) - v^*\|_\infty \leq 2\|v^*\|_\infty + \bar{\zeta} < \infty$. We now arrive at the conclusion using the result from Theorem 4, which completes the proof for Theorem 2.

The proof of Theorem 3 is identical to the proof for Theorem 2, thus we omit it here.

4.5 Extensions to Monotone MDP

In this section, we extend the proposed fitted value iteration algorithm to MDPs with monotone optimal value functions. For this class, we discuss some neural networks based function approximators that are universal within the class of monotone functions.

We use $\Theta(\sigma, K)$ to denote neural networks that consist of K hidden layers with σ as the activation function, linear output layer with one neuron, and all weights are

non-negative:

$$\Theta(\sigma, K) = \Big\{\theta : \mathscr{S} \to \mathbb{R} \Big| \theta(s) = Z^K \sigma(Z^{K-1}\sigma(\dots \\ \sigma(Z^0 s + z^0)\dots) + z^{K-1}) + z^K, 0 \preceq Z^k \in \mathbb{R}^{n_{k+1} \times n_k}, \\ z^k \in \mathbb{R}^{n_{k+1}}, \text{ for } k = 1, \dots, K, n_{K+1} = 1\Big\},$$

where $0 \preceq Z^k$ implies that every element of the matrix Z^k is non-negative. By properly selecting the activation function σ, we enforce the non-negative neural network to be monotone.

Lemma 2. *If σ is monotone, then every $\theta \in \Theta(\sigma, K)$ is a monotone function of $\mathscr{S}$.*

Proof. Each $\theta \in \Theta$ is a composition of linear functions and activation function σ. Since all weights are non-negative, the linear functions are monotone and the result follows. □

We now focus on MDPs where the optimal value function v^* or the optimal Q function $q^*(\cdot, a)$ with fixed $a \in \mathscr{A}$ is *monotone*. A function $v \in \mathcal{B}(\mathscr{S})$ is *monotone* if $s, s' \in \mathscr{S}$ and $s \preceq s' \implies v(s) \leq v(s')$. Here we use $\preceq$ to denote the usual componentwise inequality relationship over the state space $\mathscr{S}$.

We next identify some sufficient conditions under which v^* and $q^*(\cdot, a)$ are monotone functions.

Assumption 6. *The following holds*

(i) The correspondence Γ is monotone in the sense that for all $s, s' \in \mathscr{S}$ such that $s \preceq s'$, we have $\Gamma(s) \subseteq \Gamma(s')$.

(ii) For each $a \in \mathscr{A}$, the cost function $c(\cdot, a)$ is monotone, i.e. for all $s, s' \in \mathscr{S}$ such that $s \preceq s'$, $a \in \Gamma(s)$ and $a \in \Gamma(s')$, we have $c(s, a) \leq c(s', a)$.

(iii) For each $a \in \mathscr{A}$ and $w \in \mathscr{W}$, the transition function $f(\cdot, a, w)$ is monotone, i.e., for all $s, s' \in \mathscr{S}$ satisfying $s \preceq s'$, we have $f(s, a, w) \preceq f(s', a, w)$ for all $a \in \Gamma(s)$, $a \in \Gamma(s')$ and $w \in \mathscr{W}$.

Note here we assume the correspondence Γ, the cost function c, and the transition function f are monotone with respect to state s. Then we have the following result.

Theorem 5. *Under Assumption 2 and 6, the optimal value function v^* is monotone, and $q^*(\cdot, a)$ is monotone for each $a \in \mathscr{A}$.*

Proof. See Appendix A.2. □

Having shown that the optimal value or the Q functions are monotone, we now state a remarkable result from [83, Theorem 3.1] that identifies conditions when a class of neural networks is dense in the set of monotone functions.

Theorem 6. *For every continuous monotone function $v : \mathscr{S} \subseteq \mathbb{R}^d \to \mathbb{R}$ with a compact domain $\mathscr{S}$, and for every $\epsilon > 0$, if σ is sigmoid, there exists $\theta \in \Theta(\sigma, d)$ such that $\|\theta - v\|_\infty < \epsilon$.*

It was empirically observed in [77] in the context of finite monotone MDP that if the projection operator is designed in a way that the value functions retain monotonicity throughout the simulation, then the convergence rate is superior. We can provide an informal justification for the convergence property. Recall the definition of the projection operator from (4.4). Let $\mathcal{F}_l$ be the set of neural networks with σ as the activation function and l sufficiently large. Thus, $\Theta(\sigma, d) \subset \mathcal{F}_l$. Therefore, by restricting the domain of the optimization in (4.4) to a smaller set, we automatically regularize the projection operation. In the process, we do not lose the universal function approximation ability, as the optimal value function v^* is close to some function

in $\Theta(\sigma, d)$. This can improve the convergence rate of the fitted value iteration. We will conduct a more rigorous analysis of the convergence rate in our future work.

4.6 Conclusions

In this chapter, we establish convergence of the fitted value iteration and Q-value iteration algorithm for MDPs with state-dependent action sets. The key idea is to formulate these algorithms as iterated random operators acting on certain complete metric spaces. This allows us to derive the asymptotic consistency of the two approximate dynamic programming algorithms using the main result from [62]. Our result significantly generalizes a similar result established in [60] in the context of continuous-state finite-action MDP where all actions are admissible at every state.

Chapter 5: Conclusions and Future Work

This dissertation studies multi-agent collision avoidance algorithms and RL algorithms for MDPs with state-dependent action constraints. Two decentralized multi-agent collision avoidance algorithms are proposed to handle different scenarios. The first algorithm focuses on collision avoidance for heterogeneous multi-agent systems, while the second one focuses on general nonlinear systems. The collision avoidance constraints in both algorithms are formulated as state-dependent action constraints, and RL algorithms are introduced in the second approach to handle collision avoidance. Inspired by this, we study RL algorithms for continuous MDPs with state-dependent action constraints. We establish the convergence of fitted value iteration and fitted Q-value iteration, and extend the result to monotone MDPs.

In the first part of the dissertation, we propose a decentralized collision avoidance algorithm for heterogeneous multi-agent systems based on the concept of the ECO. The introduction of the ECO provides an efficient way to compute state-dependent safety constraints for any pair of heterogeneous agents. Given an ECO, a pair of linear safety constraints can be obtained via convex optimizations in real-time, which provides a strict safety guarantee for linear systems. The overall collision avoidance problem for each agent is formulated as a convex optimization that can be solved efficiently. The proposed algorithm is a unified framework that can handle complex

scenarios with nonlinear agents, uncontrolled agents, and obstacles. The performance of this approach is demonstrated via realistic simulations. One potential future research direction is to improve the persistent feasibility of the proposed algorithm. Like most decentralized collision avoidance algorithms, our approach could be infeasible in very dense scenarios. One promising direction is to combine our work with reachability analysis and construct the dangerous set as a control-invariant set. Another critical and practical future work is to apply our algorithms to real robots.

In the second part of the dissertation, a fast RL-based collision avoidance algorithm is proposed for general nonlinear agents with continuous action space, where each agent only observes positions and velocities of nearby agents. We first decompose the multi-agent scenario and solve a two agents collision avoidance problem via RL. Through the decomposition, we significantly simplify the RL training task. The neural network is much smaller, and the training process is much faster compared to other RL-based approaches. Unlike most RL-based approaches that rely on the discretization of action spaces and assume velocities can be directly controlled, our approach is applicable to general nonlinear agents. When extended to the multi-agent problem, state-dependent action constraints from ORCA are introduced, and the collision avoidance is formulated as a simple convex optimization. The state-dependent action constraints provide improved safety performance compared to other RL-based approaches. Realistic simulations based on nonlinear bicycle models are performed with various challenging scenarios, indicating a competitive performance of the proposed method in avoiding collisions, congestion, and deadlock with smooth trajectories. In the future, we would like to apply the proposed approach to systems with more complex dynamics and validate our algorithm with real robots.

In the third part of the dissertation, we focus on RL for continuous-state continuous-action MDPs with state-dependent action constraints. This is inspired by our previous two works, which convert the collision avoidance requirement into state-dependent action constraints. We establish the convergence of fitted value iteration and fitted Q-value iteration algorithms. We also extend the algorithms and the convergence result to the case of monotone MDPs. One potential future research direction is to extend our work to monotone and convex MDPs. If we could identify a universal function approximating class specifically designed for monotone and convex MDPs, we can easily migrate our algorithms and convergence results. Another important future work is to validate the convergence of proposed algorithms and show an accelerated convergence rate for monotone MDPs via simulations.

Appendix A: Proofs

A.1 Proof of Lemma 1

The proof of part (i) is very similar to Theorem 1 (i), thus we omit it here. To prove part (ii), we need to introduce some notations. Let $g_{s,a}(\cdot) : \mathscr{W} \to \mathbb{R}$ be defined as $g_{s,a}(w) \triangleq v^*(f(s,a,w))$ where $(s,a) \in \mathscr{D}$. Let $\mathcal{G}$ be a collection of such functions as $\mathcal{G} \triangleq \bigcup_{(s,a)\in\mathscr{D}} g_{s,a}(\cdot)$. Note that we have

$$\begin{aligned}
&\left\|\hat{H}_v^{k,n}(v^*) - H_v(v^*)\right\|_\infty \\
=&\sup_{s\in\mathscr{S}}\left|\inf_{a\in\Gamma(s)}\left[c(s,a) + \frac{\gamma}{n}\sum_{i=1}^{n} v^*\left(f(s,a,W_{k,i})\right)\right]\right. \\
&\left. - \inf_{a\in\Gamma(s)}\left[c(s,a) + \gamma\int_{w\in\mathscr{W}} v^*\left(f(s,a,w)\right)\mathbb{P}\left\{dw\right\}\right]\right| \\
\leq&\gamma\sup_{(s,a)\in\mathscr{D}}\left|\frac{1}{n}\sum_{i=1}^{n} g_{s,a}(W_{k,i}) - \int_{w\in\mathscr{W}} g_{s,a}(w)\mathbb{P}\left\{dw\right\}\right| \\
=&\gamma\sup_{g\in\mathcal{G}}\left|\frac{1}{n}\sum_{i=1}^{n} g(W_{k,i}) - \int_{w\in\mathscr{W}} g(w)\mathbb{P}\left\{dw\right\}\right|.
\end{aligned}$$

Then we have the following inequality for any $\epsilon > 0$:

$$\begin{aligned}
\mathbb{P}\left\{\left\|\hat{H}_v^{k,n}(v^*) - H_v(v^*)\right\|_\infty \geq \epsilon\right\} \leq \\
\mathbb{P}\left\{\sup_{g\in\mathcal{G}}\left|\frac{1}{n}\sum_{i=1}^{n} g(W_{k,i}) - \int_{w\in\mathscr{W}} g(w)\mathbb{P}\left\{dw\right\}\right| \geq \frac{\epsilon}{\gamma}\right\}. \quad \text{(A.1)}
\end{aligned}$$

To show the right side of (A.1) converges to 0 as n goes to infinity, we show that the bracketing number of $\mathcal{G}$ is finite for each $\epsilon > 0$. Since $v^*(\cdot)$ and $f(\cdot, \cdot, w)$ are Lipschitz continuous function for every $w \in \mathscr{W}$, for all $(s_1, a_1), (s_2, a_2) \in \mathscr{D}$, we have

$$|g_{s_1,a_1}(w) - g_{s_2,a_2}(w)| \leq L_{v^*} L_f(w) \rho_D \left((s_1, a_1), (s_2, a_2)\right).$$

According to [88, Theorem 2.7.11], the bracketing number of $\mathcal{G}$ is bounded by the covering number of $\mathscr{D}$. Since $\mathscr{S}$ is compact and $\Gamma(\cdot)$ is Lipschitz continuous, $\mathscr{D}$ is bounded. Thus, the bracketing number of $\mathcal{G}$ is bounded for each $\epsilon > 0$. Then by [88, Theorem 2.4.1], we conclude that the right side of (A.1) converges to 0 as n goest to infinity. For $\hat{H}_q^{k,n}$, the proof is identical, so we omit it here.

A.2 Proof of Theorem 5

Let $v_0 \in \mathcal{C}_b(\mathscr{S})$ be a monotone function. Construct a sequence of function $\{v_k\}_{k=0}^{\infty}$ by letting $v_{k+1} \triangleq H_v(v_k)$. By Theorem 1, we have $v^* = \lim_{k\to\infty} v_k$.

Assume that $v_k \in \mathcal{C}_b(\mathscr{S})$ is monotone. Then, we have

$$\begin{aligned} v_{k+1}(s) &= [H_v(v_k)](s) \\ &= \inf_{a \in \Gamma(s)} \Big[c(s,a) + \gamma \mathbb{E}\left[v_k(f(s,a,W)\right] \Big]. \end{aligned}$$

Since v_k is monotone and $f(s, a, W)$ is monotone with respect to s, $\mathbb{E}\left[v_k(f(s,a,W)\right]$ is also monotone with respect to s, so is $c(s,a) + \gamma \mathbb{E}\left[v_k(f(s,a,W)\right]$. For every $s, s' \in \mathscr{S}$ and $s \preceq s'$, we have

$$\begin{aligned} c(s,a) + \gamma \mathbb{E}\left[v_k(f(s,a,W)\right] \\ \leq c(s',a) + \gamma \mathbb{E}\left[v_k(f(s',a,W)\right]. \end{aligned}$$

Recall by Assumption 6 that $\Gamma(s) \subseteq \Gamma(s')$. This yields

$$\begin{aligned}
&\min_{a \in \Gamma(s)} c(s,a) + \gamma \mathbb{E}\left[v_k(f(s,a,W)\right] \\
&\qquad \leq \min_{a \in \Gamma(s)} c(s',a) + \gamma \mathbb{E}\left[v_k(f(s',a,W)\right] \\
&\qquad \leq \min_{a \in \Gamma(s')} c(s',a) + \gamma \mathbb{E}\left[v_k(f(s',a,W)\right].
\end{aligned}$$

This implies that v_{k+1} is monotone. This implies that if $v_0 \in \mathcal{C}_b(\mathscr{S})$ is monotone, v^* is also monotone by the principle of mathematical induction. Since v^* is monotone, it is easy to establish that $q^*(\cdot, a)$ is also monotone.

Bibliography

[1] G. Sanchez and J.-C. Latombe, "Using a prm planner to compare centralized and decoupled planning for multi-robot systems," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2, pp. 2112–2119, IEEE, 2002.

[2] P. Švestka and M. H. Overmars, "Coordinated path planning for multiple robots," *Robotics and autonomous systems*, vol. 23, no. 3, pp. 125–152, 1998.

[3] G. M. Hoffmann and C. J. Tomlin, "Decentralized cooperative collision avoidance for acceleration constrained vehicles," in *2008 47th IEEE Conference on Decision and Control*, pp. 4357–4363, IEEE, 2008.

[4] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, pp. 1928–1935, IEEE, 2008.

[5] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*, pp. 3–19, Springer, 2011.

[6] L. Wang, A. Ames, and M. Egerstedt, "Safety barrier certificates for heterogeneous multi-robot systems," in *2016 American Control Conference (ACC)*, pp. 5213–5218, IEEE, 2016.

[7] L. Wang, A. D. Ames, and M. Egerstedt, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 2659–2664, IEEE, 2016.

[8] Y. Chen, H. Peng, and J. Grizzle, "Obstacle avoidance for low-speed autonomous vehicles with barrier function," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 194–206, 2017.

[9] D. V. Dimarogonas, S. G. Loizou, K. J. Kyriakopoulos, and M. M. Zavlanos, "A feedback stabilization and collision avoidance scheme for multiple independent non-point agents," *Automatica*, vol. 42, no. 2, pp. 229–243, 2006.

[10] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1343–1350, IEEE, 2017.

[11] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered Voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[12] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015.

[13] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3475–3482, IEEE, 2011.

[14] M. Rufli, J. Alonso-Mora, and R. Siegwart, "Reciprocal collision avoidance with motion continuity constraints," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 899–912, 2013.

[15] D. Bareiss and J. Van den Berg, "Reciprocal collision avoidance for robots with linear dynamics using LQR-obstacles," in *2013 IEEE International Conference on Robotics and Automation*, pp. 3847–3853, IEEE, 2013.

[16] H. Cheng, Q. Zhu, Z. Liu, T. Xu, and L. Lin, "Decentralized navigation of multiple agents based on ORCA and model predictive control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3446–3451, IEEE, 2017.

[17] J. Alonso-Mora, P. Beardsley, and R. Siegwart, "Cooperative collision avoidance for nonholonomic robots," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 404–420, 2018.

[18] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," in *2012 IEEE International Conference on Robotics and Automation*, pp. 360–366, IEEE, 2012.

[19] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 285–292, IEEE, 2017.

[20] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3052–3059, IEEE, 2018.

[21] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *arXiv preprint arXiv:1910.11689*, 2019.

[22] S. H. Semnani, H. Liu, M. Everett, A. de Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3221–3226, 2020.

[23] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6252–6259, IEEE, 2018.

[24] T. Fan, P. Long, W. Liu, and J. Pan, "Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios," *arXiv preprint arXiv:1808.03841*, 2018.

[25] Q. Tan, T. Fan, J. Pan, and D. Manocha, "Deepmnavigate: Deep reinforced multi-robot navigation unifying local & global collision avoidance," *arXiv preprint arXiv:1910.09441*, 2019.

[26] M. Chen, J. C. Shih, and C. J. Tomlin, "Multi-vehicle collision avoidance via Hamilton-Jacobi reachability and mixed integer programming," in *55th Conference on Decision and Control (CDC)*, pp. 1695–1700, IEEE, 2016.

[27] 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 *2015 European Control Conference (ECC)*, pp. 3304–3309, IEEE, 2015.

[28] S. Bansal, M. Chen, J. F. Fisac, and C. J. Tomlin, "Safe sequential path planning of multi-vehicle systems under presence of disturbances and imperfect information," in *American Control Conference*, 2017.

[29] M. Chen, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Robust sequential trajectory planning under disturbances and adversarial intruder," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 4, pp. 1566–1582, 2018.

[30] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015.

[31] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[32] D. V. Dimarogonas, S. G. Loizou, K. J. Kyriakopoulos, and M. M. Zavlanos, "Decentralized feedback stabilization and collision avoidance of multiple agents," *NTUA, http://users. ntua. gr/ddimar/TechRep0401. pdf, Tech. Report*, 2004.

[33] M. T. Wolf and J. W. Burdick, "Artificial potential functions for highway driving with collision avoidance," in *2008 IEEE International Conference on Robotics and Automation*, pp. 3731–3736, IEEE, 2008.

[34] M. Wang and M. Schwager, "Distributed collision avoidance of multiple robots with probabilistic buffered voronoi cells," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 169–175, IEEE, 2019.

[35] H. Zhu and J. Alonso-Mora, "B-uavc: Buffered uncertainty-aware voronoi cells for probabilistic multi-robot collision avoidance," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 162–168, IEEE, 2019.

[36] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[37] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[38] J. E. Godoy, I. Karamouzas, S. J. Guy, and M. Gini, "Implicit coordination in crowded multi-agent navigation," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[39] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed autonomous robotic systems*, pp. 203–216, Springer, 2013.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[41] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[42] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[43] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[44] J. E. Godoy, I. Karamouzas, S. J. Guy, and M. Gini, "Adaptive learning for multi-agent navigation," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1577–1585, 2015.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[46] I. M. Mitchell, "Comparing forward and backward reachability as tools for safety analysis," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 428–443, Springer, 2007.

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

[48] A. Dhinakaran, M. Chen, G. Chou, J. C. Shih, and C. J. Tomlin, "A hybrid framework for multi-vehicle collision avoidance," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2979–2984, IEEE, 2017.

[49] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, pp. 3420–3431, IEEE, 2019.

[50] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *Departmental Papers (ESE)*, p. 323, 1992.

[51] E. G. Hernández-Martínez, E. Aranda-Bricaire, F. Alkhateeb, E. Maghayreh, and I. Doush, *Convergence and collision avoidance in formation control: A survey of the artificial potential functions approach*. INTECH Open Access Publisher Rijeka, Croatia, 2011.

[52] D. M. Stipanović, P. F. Hokayem, M. W. Spong, and D. D. Šiljak, "Cooperative avoidance control for multiagent systems," 2007.

[53] H. G. Tanner and A. Boddu, "Multiagent navigation functions revisited," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1346–1359, 2012.

[54] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in applied mathematics*, vol. 11, p. 412, 1990.

[55] S. W. Ekanayake and P. N. Pathirana, "Formations of robotic swarm: An artificial force based approach," *International journal of advanced robotic systems*, vol. 6, no. 1, p. 7, 2009.

[56] F. Fahimi, C. Nataraj, and H. Ashrafiuon, "Real-time obstacle avoidance for multiple mobile robots," *Robotica*, vol. 27, no. 2, pp. 189–198, 2009.

[57] S. A. Bortoff, "Path planning for uavs," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 1, pp. 364–368, IEEE, 2000.

[58] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[59] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[60] R. Munos and C. Szepesvári, "Finite-time bounds for fitted value iteration," *Journal of Machine Learning Research*, vol. 9, pp. 815–857, 2008.

[61] H. Sharma and R. Jain, "An approximately optimal relative value learning algorithm for averaged MDPs with continuous states and actions," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 734–740, IEEE, 2019.

[62] A. Gupta, R. Jain, and P. Glynn, "Probabilistic contraction analysis of iterated random operators," *arXiv preprint arXiv:1804.01195*, 2018.

[63] H. Sharma, M. Jafarnia-Jahromi, and R. Jain, "Approximate relative value learning for average-reward continuous state MDPs," in *Proceedings UAI*, 2019.

[64] D. Shah and Q. Xie, "Q-learning with nearest neighbors," in *Advances in Neural Information Processing Systems*, pp. 3111–3121, 2018.

[65] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[66] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *The International Journal of Robotics Research*, p. 0278364920916531, 2020.

[67] J. Godoy, T. Chen, S. J. Guy, I. Karamouzas, and M. Gini, "ALAN: Adaptive learning for multi-agent navigation," *Autonomous Robots*, vol. 42, no. 8, pp. 1543–1562, 2018.

[68] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[69] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, 2015.

[70] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[71] H. Sharma, R. Jain, and A. Gupta, "An empirical relative value learning algorithm for non-parametric MDPs with continuous state space," in *2019 18th European Control Conference (ECC)*, pp. 1368–1373, IEEE, 2019.

[72] W. B. Haskell, R. Jain, H. Sharma, and P. Yu, "A universal empirical dynamic programming algorithm for continuous state MDPs," *IEEE Transactions on Automatic Control*, vol. 65, no. 1, pp. 115–129, 2019.

[73] P. Diaconis and D. Freedman, "Iterated random functions," *SIAM review*, vol. 41, no. 1, pp. 45–76, 1999.

[74] A. Gupta and W. B. Haskell, "Convergence of recursive stochastic algorithms using Wasserstein divergence," *arXiv preprint arXiv:2003.11403*, 2020. Submitted to SIAM Journal on Mathematics of Data Science.

[75] L. E. Dubins and D. A. Freedman, "Invariant probabilities for certain Markov processes," *The Annals of Mathematical Statistics*, vol. 37, no. 4, pp. 837–848, 1966.

[76] W. B. Haskell, R. Jain, and D. Kalathil, "Empirical dynamic programming," *Mathematics of Operations Research*, vol. 41, no. 2, pp. 402–429, 2016.

[77] D. R. Jiang and W. B. Powell, "An approximate dynamic programming algorithm for monotone value functions," *Operations Research*, vol. 63, no. 6, pp. 1489–1511, 2015.

[78] H. Li, H. Chen, and W. Zhang, "On model-free reinforcement learning for switched linear systems: A subspace clustering approach," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 123–130, IEEE, 2018.

[79] L. Györfi, *Principles of nonparametric learning*, vol. 434. Springer, 2002.

[80] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. USA: John Wiley amp; Sons, Inc., 1st ed., 1994.

[81] K. Hinderer, U. Rieder, and M. Stieglitz, *Monotonicity of the Value Functions*, pp. 87–104. Cham: Springer International Publishing, 2016.

[82] K. Papadaki and W. B. Powell, "Monotonicity in multidimensional markov decision processes for the batch dispatch problem," *Operations Research Letters*, vol. 35, 3 2007.

[83] H. Daniels and M. Velikova, "Monotone and partially monotone neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 6, pp. 906–917, 2010.

[84] K. Hinderer, "Lipschitz continuity of value functions in Markovian decision processes," *Mathematical Methods of Operations Research*, vol. 62, no. 1, pp. 3–22, 2005.

[85] D. P. Bertsekas, "Dynamic programming and optimal control 4th edition, volume ii," *Belmont, MA: Athena Scientific*, 2012.

[86] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.

[87] N. L. Stokey and R. E. Lucas Jr., *Recursive methods in economic dynamics*. Harvard University Press, Cambridge, MA, 1989.

[88] A. W. v. d. Vaart and J. A. Wellner, *Weak convergence and empirical processes: with applications to statistics*. New York: Springer, 1996. OCLC: 45749647.